IDNANI, ASHOK U

NUMERICALLY STABLE DUAL PROJECTION METHODS FOR SOLVING POSITIVE DEFINITE QUADRATIC PROGRAMS

City University of New York PH.D. 1980

NUMERICALLY STABLE DUAL PROJECTION METHODS FOR SOLVING POSITIVE DEFINITE QUADRATIC PROGRAMS

by

ASHOK U. IDNANI

A dissertation submitted to the Graduate Faculty in Computer Science, School of Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1980

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

4/17/80
date

Donald Goldfarb
Chairman of Examining Committee
Professor Donald Goldfarb

4/23/80
date

F. E. Thau
Executive Officer
Dean Frederick Thau

Professor M. Anshel

Professor G. Bloom

Dr. M.D. Grigoriadis

Professor G. Lidor

Professor W. Sit

Professor P. Sterbenz

Supervisory Committee

The City University of New York

Abstract

Numerically stable dual projection methods for solving positive definite Quadratic Programs

by

Ashok U. Idnani

Advisor: **Professor Donald Goldfarb**

Powell's efficient algorithm for solving general non-linear programs (NLP) with non-linear constraints which is based upon methods due to Han and Biggs, requires solving a positive definite Quadratic Program (QP) at each iteration. An efficient dual projection (DP) method for solving such a QP is presented. The DP method for QP starts at an optimal, but, in general, infeasible point and seeks feasibility while maintaining optimality. It always takes steps in the direction of the normal of a violated constraint projected onto the currently active constraint manifold. Its finite termination is proved.

Another method which we call the primal-dual projection (P-DP) method is also developed. It incorporates ideas used in our DP method and resorts to solving subproblems by using the primal projection method. It has the advantage of being able to start from an arbitrary point.

Though the DP method is related to Van de Panne and Whinston's dual tableau method for QP, the projection method takes fewer operations per iteration even under the most unfavourable conditions.

An efficient numerically stable implementation using the orthogonal and triangular factors particularly suited for the positive definite QP is presented. Tests on the use of this numerically stable DP method in Powell's algorithm on six published NLP problems produced a 45 to 70 percent reduction in the computational effort over that required by Fletcher's QP algorithm. Rigorous testing on randomly generated problems with as many as 81 variables and 243

constraints of six different primal projection methods, and the DP method show that the DP method is far more efficient. In most cases, the feasible point routine for the primal methods required more computational effort than obtaining the solution using the DP method. When started from the unconstrained minimum, the P-DP method performed slightly worse than the DP method. Finally, a method based on our results is suggested for initiating the dual simplex method for Linear Programs (LP).

Acknowledgements

In the spirit of true gratitude, I would like to acknowledge the many people that have aided me in completing this thesis.

My life and career would not be what it is if it were not for my having met and worked with **Professor Donald Goldfarb.** He inspired me and will always be remembered for leading me to a space from which I could think clearly. If it were not for the patience and understanding that he has shown me despite his extremely busy schedule, I could not have completed this work. In addition, his high level of intuitiveness and support resulted in this thesis and for all of these I will be forever grateful.

I would especially like to express my gratitude to **Professors M. Anshel, G. Bloom, G. Lidor, W. Sit, P. Sterbenz and Dr. M. D. Grigoriadis** of IBM for their careful reading of the earlier drafts of this document and their valuable comments and criticism that led to considerable improvement to the present form.

I thank the University research committee on computing for providing me with necessary funding to conduct experimental runs on the City University's IBM 370/168 computer.

I also wish to acknowlede my thanks to Bell Laboratories Inc. for their partial support in completing my thesis. I would also like to extend my thanks to the Unix Word Processing Center at Bell Labs, in particular, to **Mrs. T. Howard** for the superb job done to produce the final copy of this document.

Finally, I am grateful to my parents and to my family members for their cooperation and patience throughout my education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

symbol	meaning
a	vector of coefficients for linear term in objective function
b	r.h.s. of constraints
e_j	j-th column of identity matrix
f	quadratic objective function
g	gradient of the objective function
k	total number of inequality constraints
m	number of variables
n	a constraint normal
n^+	normal of a violated constraint to be added to the basis
n^-	normal of a constraint to be removed from the basis
n^{-*}	row corresponding to the constraint being deleted in N^*.
p	usually, index of a violated constraint with normal n^+
q	number of active linearly independent constraints
r	infeasibility multipliers IM, $r = N^* n^+$.
s	vector of slacks
t	step length in z direction
t_1	max. step length in z direction with nonnegative LM
t_2	step length in z direction when a violated constraint becomes active
u	Lagrange multipliers, LM, $u = N^* g$
u^+	$N^{+*} g$
u^-	$N^{-*} g$
x	solution variables
z	step direction, H n^+
A	set of indices of q linearly independent active constraints
A^+	set AU {p} ,p$\notin$A

A^-	a proper subset of A with one fewer element than A.
B	$L^{-1}N$
B^+	$L^{-1}N^+$
B^-	$L^{-1}N^-$
C	constraint matrix
D_j	defining hyperplane corresponding to the j-th constraint
G	constant symmetric positive definite Hessian matrix of f.
H	non-Euclidean projection operator with normals N, $H = G^{-1} - G^{-1}NN^*$
H^+	H defined with N^+
H^-	H defined with N^-
I	identity matrix
J	$Q^T L^{-1}$, m $\times$ m matrix
K	set of all k constraints.
L	Cholesky factor of $G = LL^T$, lower triangular matrix
M	linear manifold formed by the intersection of constraints in A
M^+, M^-	M defined with A^+, A^- respectively
N	mxq basis matrix of normals in set A.
N^+	N defined with A^+.
N^-	N defined with A^-
N^*	qxm matrix of generalized pseudo-inverse of N, $(N^T G^{-1} N)^{-1} N^T G^{-1}$
N^{+*}	N^* with N^+
N^{-*}	N^* with N^-
P^j_{j+1}	Givens matrix causing rotation in (j - j+1) plane
Q	orthogonal factor of $B = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$
R	upper triangular factor of B, size $q \times q$
Y	Product of Givens matrices
ϕ	Null set

LIST OF ABBREVIATIONS

D	-	Dual Method
DP	-	Dual Projection
D.P.1	-	Primal-Dual with P.1 Primal
F	-	Feasible Point Routine
FP	-	Fletcher's Feasible Point Routine for QP
FQ	-	Fletcher's Quadratic Programming Routine
IM	-	Infeasibility Multiplier(s)
K-T	-	Kuhn-Tucker
LM	-	Lagrange Multiplier(s)
LP	-	Linear Programming
NLP	-	Non Linear Programming
P	-	Primal Method
POP	-	Post Office Parcel Problem
P.1	-	Primal Projection Method (Fletcher's)
P.1.1	-	Same as P.1 with Steepest Edge Option
P.2	-	Primal Projection Method (Goldfarb's)
P.2.1	-	Same as P.2 with Steepest Edge Option
P.2'	-	Two Phased P.2
P.2'.1	-	Two Phased P.2.1
P-DP	-	Primal-Dual Projection
QP	-	Quadratic Programming
QPP	-	Quadratic Programming Problems
SDS	-	Satisfied Dual System
SP	-	Subproblem
SSP	-	Solved Subproblem

VDS	-	Violated Dual System
VLDS	-	Violated Linearly Dependent Dual System
VW	-	Van de Panne - Whinston

1. INTRODUCTION.

Quadratic Programming (QP) is becoming prominent as a method useful for solving general Non-linear Programming (NLP) problems with non-linear constraints (see Powell (29,30,31)). Algorithms to solve the Quadratic Programming Problems (QPP) are abundant in the recent literature. A parametric approach is given in Grigoriadis and Ritter (22), Ritter (32). Methods to solve the general QPP are given in Fletcher (12), Bunch and Kaufman (5) and Murray (28). Goncalves(21) describes a primal-dual method.

Basically all the above methods and various other methods can be classified into two categories. Methods in the first category use ideas similar to those used in Rosen's (33) gradient projection method (see Fletcher (12), Goldfarb (16)). Those in the other category use simplex-like tableau. The methods of Beale (2), Dantzig(7)-Wolfe(40) and Van de Panne - Whinston (38) fall into this category. A good description of these and other methods can be found in Boot (4) and Van de Panne (39). The projection methods are more efficient because they only store and update the necessary operators rather than the whole tableaux as in the simplex-like methods. There are similarities between gradient projection methods, simplex methods for QPP and the simplex method for Linear Programming (LP) and these are dealt with in detail in the literature (see Rosen(33), Goldfarb(16), Van de Panne-Whinston(38)).

All of the above methods with the exception of parametric approach and Van de Panne-Whinston's (VW) dual tableau method require a feasible point to start the algorithm and remain feasible throughout. VW's dual tableau method requires a dual feasible, that is, a primal optimal point for for some subproblem of the original problem, to start. It was found in our computational experience that on the average, 20 to 60% of the total effort required to solve the QPP, involves finding a feasible point and the necessary operators. This suggests that significant savings can be achieved, if, while seeking feasibility, optimality conditions are also satisfied simultaneously. Dual algorithms are based upon this idea. A method to find a good (in the sense that it is close to the solution of the problem) starting point for QPP with no addi-

tional work was suggested by Idnani(26). We propose a method based upon this recommendation to solve the strictly convex QPP by a method which can be viewed as dual to the gradient projection methods. This can also be thought of as an extension of the dual simplex method for LP. Unlike the primal methods, the proposed dual projection method starts at an optimal, but, in general, an infeasible point. It remains optimal throughout while achieving feasibility. Steps are always taken in the direction of the projected normal, hence it can be viewed as constraint normal projection method. It is related to VW's dual tableau method.

The gradient projection methods and the simplex-like methods move in a direction such that the objective function improves while maintaining feasibility. Such a direction is given by projecting the gradient onto the manifold defined by the intersection of the active constraints. This leads to a further classification among the projection algorithms according to the dimension of the manifold in which the steps are taken. When a comparison of two similar gradient projection methods was done by Idnani(26), it was found that the algorithm that took steps on as high dimensional manifolds as possible was more efficient and took fewer steps. Some recent computational results of Lenard (27) support these findings. Since our proposed dual projection method starts from unconstrained point, it is expected to take steps on high dimensional manifolds and appears to be efficient.

In section 2, we define the problem, present the notation and recursion formulae for updating the operators. In addition, we present some important properties of the operators necessary to develop the algorithm. In section 3, we develop a dual and a primal-dual projection method and prove their finite termination. An efficient numerically stable implementation of the algorithms using orthogonal factors and their updating whenever a basis change takes place is shown in section 4. Section 5 shows the relationship of the dual projection algorithm to VW's dual tableau method which is based on duality theory. In section 6, we show how this dual method can be used as a subroutine in Powell's code for solving general NLP problems. We also present the results of our runs with Powell's NLP method on six published NLP problems, comparing the dual projection method with one of the primal methods, namely, Fletcher's. In

section 7, we present the steepest face or edge considerations for dropping constraints in the primal methods and a somewhat comparable rule which we call the greatest increase choice for adding a constraint in the dual methods. The performance and comparison of six primal projection methods, dual and primal-dual projection methods, along with the experimental design of our testing procedure is described in section 8. Finally, some conclusions and areas for future research along with a suggestion for dual simplex method for LP and phase 1 of primal simplex method for LP which is based on our results is made.

2. PRELIMINARIES.

Problem Definition.

The general convex QPP in m variables and k constraints is defined as

$$\textit{minimize} \quad f(x) = a^T x + 1/2 x^T G x \tag{2.1}$$

$$\textit{subject to} \quad C^T x - b = s \geq 0 \tag{2.2}$$

where x and a are m vectors,

G is an mxm symmetric positive definite matrix,

C is an mxk constraint matrix,

b is a k vector

and s is the k vector of non-negative slacks.

The matrix C can be written as $(n_1\ n_2 \cdots n_k)$ where n_j is the m vector normal of the j-th constraint in (2.2).

Notation.

We will use matrix notation. Lower case letters x,y,... etc. will denote column vectors, scalars or elements of a set. Upper case letters C,G,... etc., will denote matrices or a set of indices of the k constraints in (2.2). Superscript T denotes transpose, so that a^T is a row vector. A vector relation applies to each component of the vector, (i.e.) $x \geq 0$, indicates that each component of x is non-negative. The identity matrix will be denoted by I and the j-th column of I will be represented as e_j. Sometimes we will use subscripted I to show the identity matrix of the size of the subscript. Normally, the components of a column vector will be indicated by subscripts. Matrix, vector and scalar zero will be represented as 0. The null set will be denoted by ϕ. G^{-1} will represent the inverse of the matrix G. We will use $\bar{A}$ to indicate A after an update.

We will use K to indicate the set of indices of all the k constraints in (2.2). We will denote the set of indices corresponding to linearly independent constraints that are currently active as

A. A constraint is said to be active if it is satisfied as an equality, i.e., the slack corresponding to that constraint is 0. Clearly, the set A is a subset of K. The matrix of normals of constraints whose indices are in A will be represented as N(A) or N and the b vector as b(A) and the s vector corresponding to the set A will be denoted as s(A ,x). We will use K\A to indicate the set of indices in K but not in A. A^+ will denote the set (A U{p}), where p is an element in K\A. Similarly, A^- will represent a proper subset of A, whenever A is not null and will contain one fewer element than A. To be consistent, N^+ and $N(A^+)$ represent the set of normals in A^+ and N^- and $N(A^-)$ represent the set of normals in A^-. n^+ will be used to indicate the column vector in C, corresponding to p in the definition of A^+. Likewise, n^- will be used to indicate the column vector to be deleted from N to get N^-. Usually, the number of elements in A will be q, $q \leq m$.

Operators and Factorization.

The matrix G as defined in (2.1) can be decomposed into the product of two more convenient matrices as follows:

$$G = LL^T \tag{2.3}$$

where L is a lower triangular square matrix of size m. Define

$$B = L^{-1}N \tag{2.4}$$

where B is an mxq matrix. The orthogonal factors of B can be written as

$$B = Q \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{2.5}$$

where Q is an mxm orthogonal matrix and R is an upper triangular matrix of dimension qxq.

The k inequality constraints in (2.2) represent k closed half-spaces in m dimensional Euclidean space E^m. The hyperplane corresponding to a strict equality in (2.2) is a "defining hyperplane" for the associated half-space. Let D_j represent such a defining hyperplane corresponding to the j-th constraint in (2.2). (i.e.)

$$D_j = \{ x \mid n_j^T x - b_j = s_j = 0 \} \tag{2.6}$$

Let the set of indices of q linearly independent constraints be A. Then we will use M(A) or M to indicate the m-q dimensional manifold, formed by the intersection of these q linearly independent defining hyperplanes. A set of hyperplanes are linearly independent if the corresponding constraint normals are linearly independent. When A is the null set, M(A) is the full space E^m and when A has cardinality m, (i.e. q=m), M(A) is a point with zero degrees of freedom.

We will frequently encounter the projection of a vector y in the m dimensional space E^m onto the linear manifold M. This projection can be accomplished by using the projection operator,

$$H(N(A)) = H(N) = H = G^{-1} - G^{-1}N(N^T G^{-1} N)^{-1} N^T G^{-1} \tag{2.7}$$

where H is an mxm positive semi-definite matrix of rank m-q. If G $\neq$ I, H is a non-orthogonal projection operator. We will use M^+ and M^- for $M(A^+)$ and $M(A^-)$ respectively. We will use a like notation for H, (i.e.) H^+ and H^-.

Another operator we will use is the generalized pseudo-inverse of N defined as

$$N^* = (N^T G^{-1} N)^{-1} N^T G^{-1} \tag{2.8}$$

where N^* is a qxm matrix of rank q. We will use a notation similar to N for N^{+*} and N^{-*}. We will use n^{-*} to denote the row corresponding to the index of the deleted constraint in N^*.

The two operators H and N^* can be written in terms of the factors of B. Applying (2.3) through (2.5) in (2.7) we can write

$$\begin{aligned} H &= L^{-T}(I - B(B^T B)^{-1} B^T) L^{-1} \\ &= L^{-T}(I - Q \begin{bmatrix} R \\ 0 \end{bmatrix} (R^T R)^{-1} (R^T \; 0) \; Q^T) L^{-1} \end{aligned} \tag{2.9}$$

Partitioning Q as $(Q_1 \; Q_2)$ where Q_1 is of size mxq and using the orthogonality of Q, we can reduce (2.9) to

$$H = L^{-T} Q_2 Q_2^T L^{-1} \tag{2.10}$$

Similarly N^* can be simplified as follows.

$$\begin{aligned} N^* &= (B^T B)^{-1} B^T L^{-1} \\ &= (R^T R)^{-1} (R^T \; 0)\; Q^T L^{-1} \\ &= R^{-1} Q_1^T \; L^{-1} \end{aligned} \tag{2.11}$$

The following well known properties will be used frequently and their proofs can be found in Goldfarb (15,16).

$$Hw = 0 \; iff \; w = N \; v. \tag{2.12}$$

$$w^T H w = 0 \; iff \; w = N \; v. \tag{2.13}$$

Using definitions (2.7) and (2.8) and algebraic manipulations, it is easy to see that

$$H \; G \; H = H \tag{2.14}$$

$$H \; H^+ = H^+ \tag{2.15}$$

$$N^* \; G \; H = 0 \tag{2.16}$$

In the algorithm presented in the next section, we will add a new constraint to or delete a constraint from the current basis. The number of operations (multiplications and divisions) required to calculate the two operators from scratch is prohibitively large and would make our algorithms impractical. However, updating of these operators can be achieved by using the following recursion formula which cost only $0(m^2)$ operations (see Fletcher(12)). In our actual implementation we store only the matrices $Q^T L^{-1}$ and R in addition to the original data, updating them whenever there is a change of basis. The required recursions are given in section 4.

Adding a constraint with normal n^+.

$$H^+ = H - zz^T/(z^T n^+) \tag{2.17}$$

$$N^{+*} = \begin{bmatrix} N^* \\ 0 \end{bmatrix} + \begin{bmatrix} -N^* n^+ \\ 1 \end{bmatrix} z^T \; / \; (z^T n^+) \tag{2.18}$$

where $z = H\ n^{+}$.

Deleting a constraint.

$$H^{-} = H + \frac{n^{-*T} n^{-*}}{n^{-*} G\ n^{-*T}} \tag{2.19}$$

$$N^{-*} = N^{*} - \frac{N^{*} G\ n^{-*T} n^{-*}}{n^{-*} G\ n^{-*T}} \tag{2.20}$$

where n^{-*} is the row corresponding to the constraint being deleted in N^{*}. Of course, the row corresponding to the dropped constraint is zero in N^{-*}. Clearly, H = G^{-1} when A is the null set and H = 0 when q = m. Also $N^{*} = N^{-1}$ in the latter case.

Multipliers.

For a given linearly independent set of normals N = N(A), we will define a q-vector v of multipliers corresponding to the m vector y as

$$v = N^{*} y \tag{2.21}$$

If N w = y, then v = w since $N^{*}N$ = I, but (2.21) does not, in general imply that N v = y.

Lagrange Multipliers.(LM)

The gradient of the objective function (2.1) can be written as

$$g(x) = \nabla_x f(x) = G\ x + a \tag{2.22}$$

The vector of multipliers corresponding to y = g(x) in (2.21) are called the Lagrange Multipliers (LM) and are represented by vector

$$u(x) = N^{*} g(x) \tag{2.23}$$

We will denote $N^{+*} g(x)$ as $u^{+}(x)$ and $N^{-*} g(x)$ as $u^{-}(x)$. Other names used for LM are shadow prices and dual variables.

Lemma 1.

$$u(x) = N^* g(x) \text{ is independent of } x \text{ on a given manifold } M.$$

Proof:

Let y and z be two points on M. Then,

$$u(y) - u(z) = N^* G\ (y-z)$$
$$= (N^T G^{-1} N)^{-1}\ N^T (y-z)$$

Since y and z are on M, $N^T(y-z) = 0$ and therefore $u(y) = u(z)$●

Infeasibility Multipliers. (IM)

Infeasibility multipliers (IM) corresponding to a particular constraint with normal n are defined by replacing y in (2.21) by the normal n. We denote this vector by r.

$$r = N^* n \tag{2.24}$$

As we will see in the next section, the IM indicate whether or not a feasible point exists. By the above definition, r is independent of x on a given manifold M.

3. ALGORITHM AND FINITENESS.

Unconstrained minimum

The unconstrained minimum of the objective function (2.1) can be obtained by using classical techniques. In particular, by setting the derivative of f with respect to x to zero, $\nabla_x f(x) = G\ x + a = 0$ and solving for x, we get the unconstrained minimum

$$x^0 = -G^{-1}a \tag{3.1}$$

Minimum on a manifold.

If, at the unconstrained minimum x^0, all the constraints (2.2) are satisfied, (i.e.) $s(\ K\ ,x^0) \geq 0$, then x^0 solves the problem. Otherwise, the solution to the problem has to lie on some manifold of q linearly indepent defining hyperplanes and the indices of these q constraints are a subset of K and $1 \leq q \leq$ min.(k,m). Our problem then, is to find this manifold and the minimum of f(x) on it. (See figure 1 on page 25.)

The necessary condition for x^* to be the minimum on M is that the gradient at x^* lies in the subspace orthogonal to M, that is,

$$H\ g(x^*) = 0 \tag{3.2}$$

This implies that

$$g(x^*) = Gx^* + a = Nu \tag{3.3}$$

Basic approach

The following definitions of subproblem(SP) and solved subproblem (SSP) are essential for explaining the basic dual and primal-dual approaches developed in this section.

Subproblem (SP)

It is a QPP with the same objective function as (2.1) with only a subset of the original constraints (2.2). It can be thought of as a set of constraints for all practical purposes, (i.e), SP $\subseteq$ K.

Solved Subproblem (SSP)

Let the solution of a Subproblem SP lie on a manifold of q linearly independent active constraints. Then SSP is this set of q constraints. Clearly, SSP $\subseteq$ SP.

Dual approach

Now the basic dual approach can be explained as follows:

Assume that some SSP and the solution point on the manifold corresponding to it are available.

1) Pick a violated constraint, if any, say p-th.

2a) Define SP = (SSP U {p}).

2b) Solve the SP.

2c) If no solution, terminate because the feasible region is empty; otherwise, define an SSP.

Repeat 1 and 2 until solution. One can use the empty set and the unconstrained minimum as a possible start. Steps 2b and 2c are developed later.

Primal-Dual approach

The primal-dual approach can be explained in the following five steps.

0) *Starting Point.*

Define an SP, some subproblem of the original problem and go to step 2.

1) *Define SP.*

If an SSP exists, then SP = (SSP U {p}), where p is the index of a violated constraint, if any. Otherwise, define some SP to be solved.

2) *Feasibility.*

Apply dual projection method to find a feasible point for the SP. If none can be found, terminate.

3) *Optimality.*

If the feasible point solves the subproblem SP, then define SSP and go to step 1, otherwise go to step 4.

4) *Solve subproblem SP.*

Apply primal projection method to solve SP and define SSP. Go to step 1.

Step direction

In order to develop the necessary tools for the dual algorithm, first, we will show the form of the step from the minimum on a manifold M to the minimum on a submanifold M^+ of M.

Lemma 2.

Let x and x^+ be optimum points on manifolds M and $M^+ \subset M$, respectively. Then, we can find x^+ from x in one step, z, and z is of the form $G^{-1}N^+y$, where y is a (q+1) vector, that is,

$$x^+ = x + G^{-1}N^+y \tag{3.4}$$

Proof:

Since $M^+ \subset M$, $g(x)$ is orthogonal to M^+. Therefore, $g(x^+) - g(x)$ is orthogonal to M^+. This implies that

$$g(x^+) - g(x) = N^+y$$

$$G(x^+ - x) = N^+y$$

which results in (3.4)●

The above holds as long as $M^+ \subset M$, therefore, $A^+ \backslash A$ need not be only one element. One can apply the above lemma even when A is the null set. The following lemma gives another representation of the optimum point x^+ in (3.4).

Lemma 3.

Let x and x^+ be the optimum points on manifolds M and $M^+ \subset M$. Then, we can find x^+ from x in one step, z, (i.e.)

$$x^+ = x + z \tag{3.5}$$

where
$$z = -N^{+*T} s(A^+, x). \tag{3.6}$$

Moreover, x^+ is on M^+ even if x is an arbitrary point in E^m.

Proof:

From (3.4) it follows that

$$s(A^+, x^+) = 0$$
$$= s(A^+, x) + N^{+T} G^{-1} N^+ y$$

Consequently (3.4) can be written as

$$x^+ = x - G^{-1} N^+ (N^{+T} G^{-1} N^+)^{-1} s(A^+, x)$$

which is just (3.6).

When x is an arbitrary point in E^m, (3.5) and (3.6) yield a point which lies on M^+ since it follows from (3.5) and (3.6) that $s(A^+, x^+) = 0$●

Theorem 1.

The minimum of (2.1) on manifold M lies at

$$x = N^{*T} b(A) - H a \tag{3.7a}$$

and the LMs of active constraints are given by

$$u(x) = (N^T G^{-1} N)^{-1} b(A) + N^* a \tag{3.7b}$$

Proof:

Applying Lemma 3 with A replaced by null set and A^+ by A, we get

$$x = -G^{-1} a - N^{*T} s(A, -G^{-1} a)$$
$$= -G^{-1} a - G^{-1} N (N^T G^{-1} N)^{-1} (-N^T G^{-1} a - b(A))$$
$$= N^{*T} b(A) - H a$$

which is (3.7a). (3.7b) can be shown by using the definition of LM and the fact that

$N^* G\ H = 0$●

One could enumerate subsets of k constraints and evaluate x, f and s at the minimum point for every subset. Theil-Van de Panne (37) suggest an intelligent way of picking the subsets to be evaluated. It involves enormous amount of work and is impractical for large problems.

Before we present the development of the algorithm, let us stress one well known condition for the step direction z to satisfy. In order to yield an improvement in s, that is, $s(A,\bar{x}) \geq s(A, x)$, where $\bar{x} = x + z$, the direction z must satisfy

$$N^T z \geq 0. \tag{3.8}$$

This is the consequence of the fact that $s(A,\bar{x}) - s(A,x) = N^T z$ and hence the condition (3.8).

Kuhn-Tucker conditions for optimum point.

Let the manifold M be determined uniquely by a set of active constraint indices in A, (i.e), M is not contained in the m-1 dimensional subspace corresponding to any of the defining hyperplanes whose indices are in K\A. Then, the Kuhn-Tucker (K-T) necessary conditions for a point $x^* \in M$ to be the global minimum of the problem (2.1), (2.2) are (see Goldfarb (15,16)),

Optimality:

$$H\ g(x^*) = 0 \tag{3.9}$$

$$N^* g(x^*) = u(x^*) \geq 0 \tag{3.10}$$

Feasibility:

$$s(K\backslash A, x^*) \geq 0 \tag{3.11}$$

$$s(A, x^*) = 0 \tag{3.12}$$

The optimality conditions imply that the gradient at x^* can be written as nonnegative linear combination of the normals N.

We need to define a *v*iolated *d*ual *s*ystem (VDS) in order to make the development of the

algorithm simple. A VDS with respect to the set AU{p} and a point x which we denote as E(x,A,p), is one that meets the following criteria.

1) p is the index of a constraint in K\A with normal n^+, such that $|Hn^+| \neq 0$, (i.e.) the set of constraints whose indices are in $A^+ = A \cup \{p\}$, are linearly independent. (3.13)

2) $H^+ g(x) = 0.$ (3.14)

3) $s(A, x) = 0.$ (3.15)

4) $u^+(x) = N^{+*} g(x) \geq 0.$ (3.16)

5) $s(\{p\}, x) < 0.$ (3.17)

We will define a *s*atisfied *d*ual *s*ystem (SDS) with respect to the set AU{p} and a point x as one which satisfies (3.13) - (3.16) and $s(\{p\},x) = 0$. Such an SDS will be denoted as S(x,A,p). Now we will present a Lemma on the rate of change of LM when we take a step from a VDS in the projected normal direction.

Lemma 4.

Let E(x,A,p) be a VDS. Let $u^+(x)$ be the vector of q+1 LMs. Let the step direction z be,

$$z = H\, n^+. \tag{3.18}$$

Let $\bar{x}$ be a point such that

$$\bar{x} = x + t\, z, \tag{3.19}$$

where t is a positive scalar representing the step length. The LM along the direction z is given by

$$u^+(\bar{x}) = \begin{Bmatrix} u_j^+(x) - t\ e_j^T N^* n^+ \ , \ j = 1,2...,q \\ u_{q+1}^+(x) + t \end{Bmatrix} \tag{3.20}$$

Proof

$$u^+(\bar{x}) = N^{+*} g(x) + t\ N^{+*} G\ z$$

$$= N^{+*} g(x) + t\ N^{+*} G\ H\ n^+ \tag{3.21}$$

Using recursion formula (2.18) for N^{+*} and properties (2.14) and (2.16) in (3.21) gives (3.20)●

By applying Lemmas 2, 3 and 4 to a VDS we now show how a step in the direction of the projected normal results in a VDS or an SDS depending upon the step length.

Theorem 2.

Let E(x,A,p) be a VDS. If the point $\bar{x}$ is defined by

$$\bar{x} = x + t\ z, \tag{3.22}$$

where

$$z = H\ n^+, \tag{3.23}$$

$$t = minimum\ (t_1\ ,\ t_2) \geq 0 \tag{3.24}$$

$$t_1 = min. \left\{ \underset{\substack{r_j > 0 \\ j=1,\ldots,q}}{min.} \left\{ \frac{u_j^+(x)}{r_j} \right\},\ \alpha \right\} \tag{3.25}$$

$$t_2 = -s(\{p\},x) \,/\, (z^T n^+) \tag{3.26}$$

and

$$r = N^* n^+$$

then

$$s(\{p\},\bar{x}) \geq s(\{p\},x) \tag{3.27}$$

and

$$f(\bar{x}) - f(x) = t\ z^T n^+\ (1/2\ t + u_{q+1}^+(x)) \geq 0. \tag{3.28}$$

Moreover, if t = t_1, we have a VDS, E($\bar{x}$,A,p), if $t = t_2$ we have an SDS, $S(\bar{x},A,p)$.

Proof:

From (2.18), (3.23) and (3.26) it follows that

$$-N^{+*T}s(A^{+},x) = (-s(\{p\},x)/(z^{T}n^{+}))\ z\ = t_2\, z \tag{3.29}$$

since E(x,A,p) is a VDS. Therefore, if t $= t_2 < t_1$, it follows from Lemma 3 that $s(A^{+},\bar{x}) = 0$. Also

$$H^{+}g(\bar{x}) = H^{+}g(x) - H^{+}G\ N^{+*T}s(A^{+},x) = 0 \tag{3.30}$$

since $H^{+}G\ N^{+*T} = 0$ by (2.16) and $u^{+}(\bar{x}) > 0$ by Lemma 4. Since

$$\bar{x}\ =\ \bar{x}(t)\ =\ \bar{x}(t_2),$$

we have an SDS of the form $S(\bar{x},A,p)$.

If t $= t_1 < t_2$, then, we have a VDS of the form $E(\bar{x},A,p)$ by the above and by linearity of $S(A^{+},\bar{x}(t))$, $H^{+}g(\bar{x}(t))$ and $u^{+}(\bar{x}(t))$ with respect to the step length t.

The change in function value can be calculated by using Taylor's theorem as

$$f(\bar{x}) - f(x) = t\ z^{T}g(x) + 1/2\ t^{2}\, z^{T}G\ z \tag{3.31}$$

where

$$z^{T}G\ z = n^{+T}H\ G\ H\ n^{+} = n^{+T}H\ n^{+} = z^{T}n^{+} > 0 \tag{3.32}$$

and

$$z^{T}g(x) = n^{+T}H\ g(x) \tag{3.33}$$

From the recursion formulae (2.17), (2.18) and the fact that $H^{+}g(x) = 0$,

$$u^{+}_{q+1}(x) = \frac{n^{+T}H\ g(x)}{n^{+T}H\ n^{+}} \tag{3.34}$$

and

$$H\ g(x) = H\ n^{+}\ u^{+}_{q+1}(x) \tag{3.35}$$

$$n^{+T}H\ g(x) = n^{+T}\ H\ n^{+}\ u^{+}_{q+1}\ (x)$$

$$= z^{T}n^{+}\ u^{+}_{q+1}(x) \tag{3.36}$$

Substituting (3.36) into (3.33) and (3.31), and (3.32) into (3.31) yields (3.28). Since all elements of (3.28) are nonnegative, $f(\bar{x}) \geq f(x)$. Moreover, as long as $t > 0$, $f(\bar{x}) > f(x)$●

The above theorem suggests that if one starts with a VDS, one can move to another VDS or an SDS with improvement in f and s. At an SDS one can update N^* and H by adding the constraint and check if another VDS can be obtained by picking a violated constraint. In that case, one can reapply the above theorem. If there is no violated constraint, we can terminate. If the violated constraint is linearly dependent, then, theorem 2 breaks down. This situation is discussed in theorem 3. However, if a VDS is arrived after the step, then the constraint corresponding to the minimum in t_1 can be dropped from the basis and A, N^* and H updated. We only need to show that this new point with respect to the new set of constraints forms a VDS by showing that the projection of the gradient on the new manifold is zero. This can be seen from (2.19) since the LM of the constraint dropped is zero. An SDS is guaranteed to be obtained in no more than q'+1 applications of theorem 2 where q' is the number of active constraints at the most recent SDS.

When A is a full set, (i.e. q = m) or when $|H\ n^+| = 0$, we no longer have a VDS and hence theorem 2 is not applicable. At this juncture, one could drop a constraint to form a VDS and apply theorem 2. The condition to be met for the choice of the constraint to be dropped can be found from the requirement for VDS and Lemma 4 and is shown in the next theorem.

Dropping criteria

We will define a violated linearly dependent system VLDS, with respect to the set AU {p}, denoted as V(x,A,p) the one that meets the following criteria.

1) p is the index of a constraint in K\A with normal n^+ such that

   $|H\ n^+| = 0.$

2) $|H\ g(x)| = 0$

3) $s(A, x) = 0$

4) $u(x) = N^{*}g(x)$ ≥ 0

5) $s(\{p\},x)$ < 0

Theorem 3.

Let V(x,A,p) be a VLDS. Compute

$$t_1 = min. \left\{ \infty, \underset{j=1,\ldots,q}{\underset{r_j > 0,}{min.}} \left\{ \frac{u_j(x)}{r_j} \right\} \right\} \tag{3.37}$$

Then,

a) if $t_1 = \infty$, then there is no allowable direction z such that $s(\{p\}),x) < s(\{p\},\bar{x})$ and $s(A,\bar{x}) \geq s(A,x) = 0$, where $\bar{x} = x + z$.

b) if $t_1 < \infty$, then the constraint corresponding to the minimum in (3.37) can be dropped from the set A to give A^{-} and $E(x,A^{-},p)$ is a VDS

Proof:

Since V(x,A,p) is a VLDS, we have $|H\ n^{+}| = 0$ which implies

$$N\ r = n^{+} \tag{3.38}$$

and

$$r = N^{*}n^{+}$$

Now there are two possibilities.

Case 1 $r \leq 0$.

From (3.38) we can write

$$n^{+T}z = r^{T}N^{T}z \tag{3.39}$$

where z is the step direction. By (3.8), $s(A,\bar{x}) \geq s(A,x)$ only if $N^{T}z \geq 0$ and $s(\{p\}, \bar{x}) > s(\{p\},x)$ only if $n^{+T}z > 0$. If r <= 0, then, it follows from (3.39) that no allowable direction exists such that $s(A,\bar{x}) \geq s(A,x)$ and $s(\{p\},\bar{x}) > s(\{p\},x)$.

Case 2 At least one component of r (say r_q) is positive.

Now we will show the condition required to drop a constraint so that $E(x,A^-,p)$ is a VDS. Consider dropping a constraint whose component in r is positive, say q-th. Since V(x,A,p) is a VLDS, $|H\ g(x)| = 0$ and $|H\ n^+| = 0$. Let $A^-U\{q\} = A$. Consequently by (2.19) and the definition of u(x) and r,

$$H^-g(x) = \frac{n^{-*T}u_q(x)}{n^{-*}G\ n^{-*T}} \tag{3.40}$$

$$H^-n^+ = \frac{n^{-*T}r_q}{n^{-*}G\ n^{-*T}} \tag{3.41}$$

$$\frac{n^{+T}H^-g(x)}{n^{+T}H^-n^+} = \frac{u_q(x)}{r_q} \tag{3.42}$$

Let H^+ and N^{+*} denote the operators H and N^* with respect to the set $A^+ = A^-U\{p\}$ for the rest of this theorem. Applying the recursion (2.17) we find

$$H^+g(x) = H^-g(x) - \frac{H^-n^+n^{+T}H^-g(x)}{n^{+T}H^-n^+} \tag{3.43}$$

By substituting (3.40) - (3.42) in (3.43) we find that $H^+g(x) = 0$. Also from (3.41), $|H^-n^+| > 0$.

The other condition required for $E(x,A^-,p)$ to be a VDS is that $N^{+*}g(x) \geq 0$. Using the recursion formulae (2.20) and (2.18) we can write

$$N^{-*}g(x) = N^*g(x) - \frac{N^*G\ n^{-*T}u_q(x)}{n^{-*}G\ n^{-*T}} \tag{3.44}$$

$$N^{-*}n^+ = N^*n^+ - \frac{N^*G\ n^{-*T}r_q}{n^{-*}G\ n^{-*T}} \tag{3.45}$$

and using (3.42)

$$N^{+*}g(x) = \begin{bmatrix} N^{-*}g(x) \\ 0 \end{bmatrix} + \begin{bmatrix} N^{-*}n^{+} \\ 1 \end{bmatrix} \frac{u_q(x)}{r_q} \tag{3.46}$$

Substituting (3.44), (3.45) in (3.46) we find

$$N^{+*}g(x) = \begin{Bmatrix} u(x) - r\,\dfrac{u_q(x)}{r_q} \\ \dfrac{u_q(x)}{r_q} \end{Bmatrix} \tag{3.47}$$

The requirement of nonnegativity of $N^{+*}g(x)$ suggests that the constraint to be dropped be the one corresponding to

$$\underset{\substack{r_j > 0, \\ j=1,\ldots,q}}{min.} \left\{ \frac{u_j(x)}{r_j} \right\}$$

The Theorems 2 and 3 provide a constructive procedure which results in improvement of the function value and reduction of infeasibility of a violated constraint, ultimately reaching an SDS. Note that in case 2 of theorem 3, x is not changed; only the set A is. Successive applications of the above theorems as necessary, starting from a VDS will converge to the global solution of the problem. Now we have the necessary tools to present the algorithm and show its finiteness.

Dual Algorithm:

Step 1. Compute G^{-1}, the unconstrained minimum, $x = -G^{-1}a$ and f = $-1/2a^{T}x$. Set $H = G^{-1}$ and $A = \{\phi\}$, $q = 0$, the LM, u = 0.

Step 2. Compute s(K\A,x). Find $s(\{p\},x)$ = min. (s(K\A,x)). If $s(\{p\},x) \geq 0$, then terminate; x solves the problem; otherwise, let the normal corresponding to p be n^{+} and set $u^{+} = \begin{bmatrix} u \\ 0 \end{bmatrix}$.

Step 3. Compute the step direction z = H n^{+}.

If q > 0, compute the IM, r = $N^{*}n^{+}$. If |z| >0, go to step 4.

Compute t_1 as in (3.37). If $t_1 = \infty$, then there is no feasible point that solves (2.2); terminate. Otherwise, drop the constraint corresponding to the minimum in (3.37), update u as in (3.47), A, H, N,* set q = q-1 and repeat this step.

Step 4. Compute t_1 as in (3.25), t_2 as in (3.26) and t = min.(t_1,t_2). Take step $\bar{x}$ = x + t z and update $\bar{f}$ as in (3.28) and $u^+(\bar{x})$ as in (3.20). Calculate s({p},$\bar{x}$). Set x ← $\bar{x}$ and f ← $\bar{f}$. If t = t_2, then the p-th constraint is added, (i.e.), set A ← AU{p}, update H, N^* and q. Set u ← u^+. Go to step 2. It $t = t_1$, drop the constraint corresponding to the minimum value of t_1 from A, set A ← A^-, update H, N^* and q. Go to step 3●

Finiteness of Dual Algorithm

Theorem 4.

The given dual algorithm will produce the optimum point in ' 'te number of steps or terminate when there is no feasible point.

Proof:

It is clear from the algorithm that when step 2 is executed, x solves some subproblem of the original problem due to the fact that the optimality conditions with respect to that subproblem are satisfied. The subproblem may be thought of just the q active constraints. Every time a step is taken, the function value increases and we must return to step 2 in finite number of steps or terminate with no feasible point. This suggests that we will never solve the same subproblem again. Since there are a finite number of subproblems, the algorithm must terminate with the global solution, if there is one, in a finite number of steps or else indicate that the feasible region is empty●

One advantage of the dual algorithm as described is that the gradient of the objective function and its projection are never computed. The LM can be updated every time a basis change takes place. This results in a significant savings in the amount of work per iteration. Starting

from the unconstrained optimum does not require any additional work than most primal projection methods because the operators H and N^* have to be calculated for the first time (in feasible point routine) which directly or indirectly use G^{-1} or when it is a full basis, N^{-1}. Now we will present an example and geometric interpretation of the method. In order to illustrate the various parts of the algorithm, we will not require that the most violated constraint be the one to be added.

Example:

Consider the following problem.

$$\text{minimize } f(x) = (6\ 0)\begin{pmatrix}x_1\\x_2\end{pmatrix} + 1/2\ (x_1\ x_2)\begin{pmatrix}4 & -2\\-2 & 4\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}$$

subject to

1) x_1	≥ 0	normal $(1,0)^T$
2) x_2	≥ 0	normal $(0,1)^T$
3) $x_1 + x_2$	≥ 2	normal $(1,1)^T$
4) $-2x_1 - x_2$	≥ -4	normal $(-2,-1)^T$

Solution:

$$G^{-1} = \begin{pmatrix}2/6 & 1/6\\1/6 & 2/6\end{pmatrix}$$

The unconstrained minimum occurs at $x = (-2\ -1)^T$ with function value -6. The slack vector s is $(-2\ -1\ -5\ 9)^T$ and $H = G^{-1}$, set $q = 0$.

Iteration 1.

Among the first three violated constraints, let us pick the second constraint to be added, (i.e.) $p = 2$, thus making $E(x,\{\phi\},2)$ a VDS. We can now take a full step as follows.

$$z^T = (1/6\ 2/6),\ z^T n^+ = 1/3,\ t_2 = 3,\ t = t_2.$$

$$x^T = (-3/2\ 0), u = 3 \text{ and } f = -9/2.$$

Adding constraint 2 to the basis and updating H, N^*, A and q, we obtain

$$H = \begin{pmatrix} 1/4 & 0 \\ 0 & 0 \end{pmatrix}, N^* = (1/2\ 1), A = \{2\} \text{ and } q = 1.$$

Iteration 2.

The s vector at this new x is $(-3/2\ 0\ -7/2\ 7)^T$. Let us choose constraint 3 among the violated constraints to be added, (i.e.) p = 3. This makes E(x,{2},3) a VDS.

We can now calculate

$z^T = (1/4\ 0)$, $z^T n^+ = 1/4$, $r = 3/2$, $t_1 = 2$, $t_2 = 14$ and $t = t_1$. This is a partial step. After the step,

$$x^T = (-1\ 0),\ u = (0\ 2)^T \text{ and } f = -4.$$

Therefore, we can drop constraint 2 from the basis and update H, N^*, A and q.

$$H = \begin{pmatrix} 2/6 & 1/6 \\ 1/6 & 2/6 \end{pmatrix},\ A = \{\phi\} \text{ and } q = 0.$$

The slack corresponding to the constraint 3 after the step is -3.

Iteration 3.

E(x,{ϕ},3) is a VDS. We can take a full step as follows:

$$z^T = (1/2\ 1/2),\ z^T n^+ = 1,\ t_2 = 3 \text{ and } t = t_2.$$

$$x^T = (1/2\ 3/2)\ \ u = 5 \text{ and } f = 13/2.$$

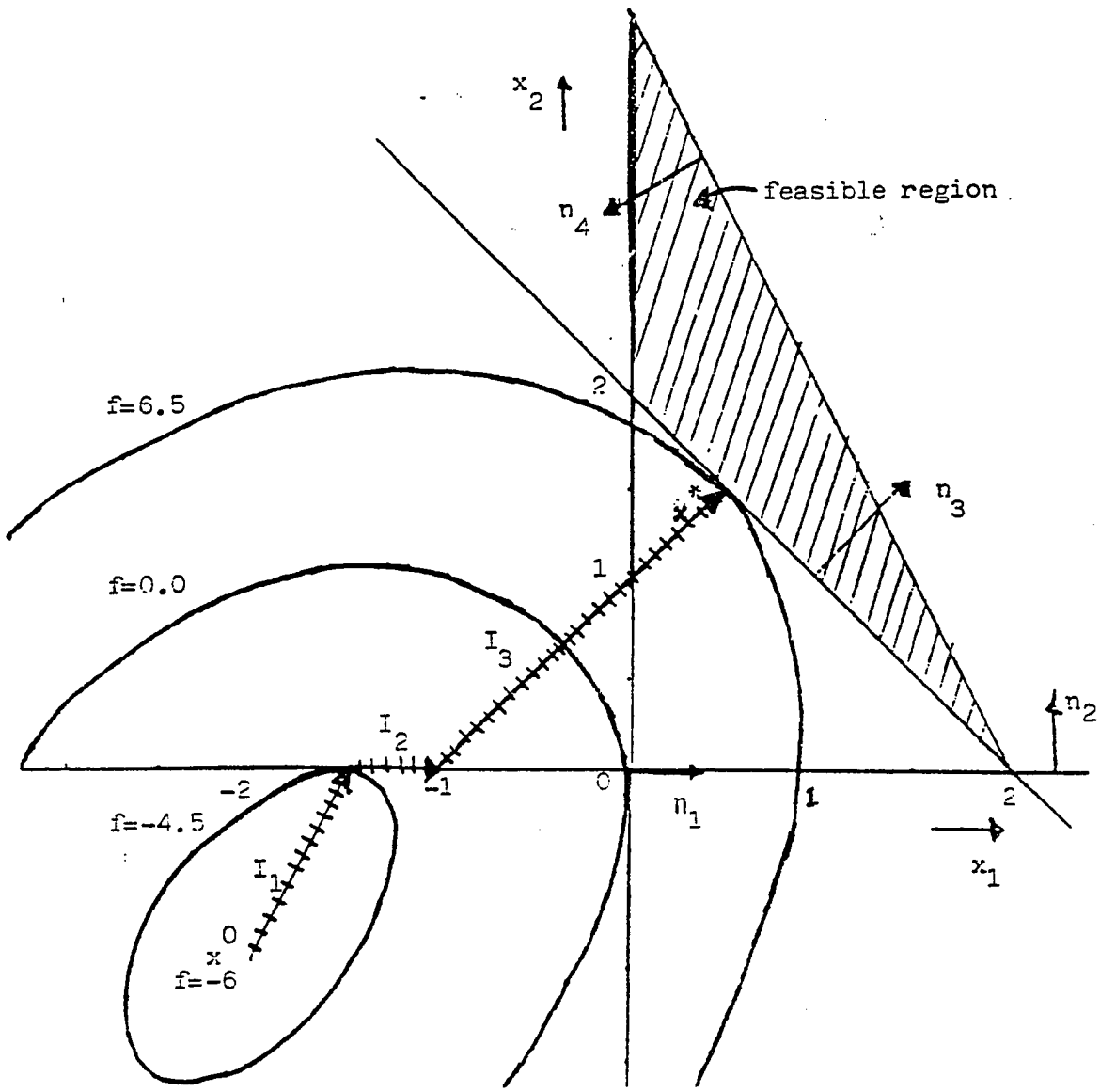
Adding constraint 3 to the basis and updating H, N^*, A and q we obtain

$$H = \begin{pmatrix} 1/12 & -1/12 \\ -1/12 & 1/12 \end{pmatrix},\ N^* = (1/2\ 1/2),\ A = \{3\} \text{ and } q = 1.$$

Iteration 4.

The s vector at this new x is $(1\ 3\ 0\ 3)^T$. Since $s \geq 0$, $x = \begin{pmatrix} 1/2 \\ 3/2 \end{pmatrix}$ solves the problem with function value 13/2.

See figure 1 on page 25 for geometric interpretation of these iterations.



⇉ A solution path

n_1, n_2, n_3, n_4 inward directed constraint normals

x^0 - unconstrained minimum

x^* - constrained minimum

x_1, x_2 variables, axes

I_1 - full step, iteration 1

I_2 - partial step, iteration 2

I_3 - full step, iteration 3

Figure 1.

Geometric interpretation of the example.

Primal-dual Method

In order to derive the primal-dual (P-D) method, we need to prove the following theorem on active constraints.

Theorem 5.

Let x solve a subproblem SP_1 with respect to q active linearly independent constraints whose indices are in set A. Let p be the index of a violated constraint in K\A, (i.e.) s({p},x) < 0. Then, $\bar{x}$, the solution to the new subproblem SP_2, comprising of the constraints whose indices are in AU{p}, must have the p-th constraint active, that is, s({p},$\bar{x}$) $= 0$ and f($\bar{x}$) $>$ f(x). (assuming SP_2 has a solution)

Proof:

Since s({p},x) < 0, the feasible region of SP_2 is a subset of the feasible region of SP_1. Moreover, since f takes the lowest value among all the feasible solutions of SP_1 at x, the function value at any of the feasible solutions of SP_2, say x', must satisfy $f(x') > f(x)$. This implies f($\bar{x}$) $>$ f(x). Furthermore, since the feasible region of SP_1 is convex, all the points on the line (x,$\bar{x}$) are in the feasible set of SP_1.

Let us assume that p is not active at $\bar{x}$, (i.e.) s({p},$\bar{x}$) > 0. Let x' be a point on the p-th constraint and on the line (x,$\bar{x}$). Then, x' can be written as a linear combination of the two extreme points x and $\bar{x}$ and $x' \neq x \neq \bar{x}$. Since the function f is strictly convex, we have

$$f(x) < f(x') < f(\bar{x}).$$

Moreover, x' is a feasible point for SP_2 because it is in the feasible set of SP_1 and satisfies the p-th constraint. Then, x' must solve SP_2 and not $\bar{x}$, hence our assumption that p is not active at the solution is invalid●

The dual projection algorithm presented earlier solves a sequence of subproblems of the original problem. The primal methods due to Fletcher (12) and Goldfarb (16) are capable of solving a QP problem provided a feasible point to that problem is available. We can combine the

two methods as follows.

Let us assume that the operators H and N^* w.r.t. some active set A are available. Then, starting from an arbitrary point and applying Lemma 3, we can get to the manifold M. Now, any primal method can be invoked to solve this subproblem of q constraints in A, ignoring the other k-q constraints. Defining a new subproblem as the currently solved subproblem and a violated constraint, if any, one can take a full dual step to arrive at a feasible point for the new subproblem or apply theorem 3 and take a full dual step if possible. (A subproblem may be of only the active constraints at the solution.) If all the LMs are of correct sign and the projected gradient is zero, we can repeat this dual step. Otherwise, we switch to the primal algorithm to solve this subproblem before returning to the dual.

Primal-Dual Algorithm

Step 1. Let x solve a subproblem with q active linearly independent constraints in set A. H, N^* and u are available. (apply Lemma 3 and/or primal method if needed)

Step 2. Same as step 2 of dual projection method.

Step 3. Same as step 3 of dual projection method.

Step 4. Compute t = t_2 = -s({p},x)/($z^T n^+$), $\bar{x}$ = x + tz, $\bar{f}$ as in (3.28) and $u^+(\bar{x})$ as in (3.20). Compute s({p},$\bar{x}$).

Set $x \leftarrow \bar{x}$, $f \leftarrow \bar{f}$, $u \leftarrow u^+$, $A \leftarrow AU\{p\}$, $q = q+1$. Update H and N^*. If u $\geq$ 0, then go to step 2. Otherwise, invoke primal to solve the subproblem whose indices are in A and go to step 2●

Finiteness of Primal-Dual Algorithm

Theorem 6.

The primal-dual algorithm converges to the global optimum, if any, in a finite number of steps or indicates that the feasible region is empty.

Proof:

At the start of the dual step, we have the solution of some subproblem of the original problem with the function value as f. The subproblem may be thought of as only the q active constraints in the set A. At this point, some constraint, say p, is violated. Consider the new subproblem as to be comprised of the constraints whose indices are in A^+. The solution to this new subproblem can be found in either one full dual step, or if it is not optimal, by a primal algorithm which is invoked. Suppose that the optimal value of the new subproblem be $\bar{f}$. Since the new subproblem is more constrained, $\bar{f} > f$ as shown in theorem 5. Since the objective function is monotonically increasing, we will never consider the same subproblem again at the beginning of a dual step. Moreover, since there are finite number of possible subproblems, the method must converge to the optimum point in a finite number of steps or must terminate with no solution in the dual step●

Linear Dependence

Unlike the primal algorithms, the dual method has a choice of the constraint to be added into the basis among the violated constraints. If possible, one could postpone adding a nearly linearly dependent constraint until a vertex is obtained, or apply the theorem 3 in order to avoid the problems of loss of accuracy in the projection or multiplier operators.

Equalities

The problem as presented in (2.1), (2.2) has only inequalities. In most practical problems, it is possible that some of the constraints may be equalities. This does not pose any additional problem to the dual methods. One could add the equality constraints first and never allow them to leave the basis. The LMs associated with these constraint equalities must be unrestricted in sign.

4. NUMERICALLY STABLE IMPLEMENTATION

It has been shown in the literature (see Bartels et al.(1)) that the methods that use the orthogonal factors described in section 2 are numerically more stable than the methods that use the projection operators H and N^*. We will now show a way to update the orthogonal factors of B introduced in section 2, whenever a basis change takes place. First, we will present Givens rotations and the structure of the product of Givens rotation matrices. Later, we will show how a matrix can be multiplied with the Givens matrices in an efficient manner.

Givens Rotations

Let us consider the problem of rotating a given m vector y into a multiple of the first column of the identity matrix, e_1, while maintaining its length. That is, determine a matrix P such that

$$P\,y = d\,e_1 \tag{4.1}$$

where d is a scalar equal to the length of the vector y, (i.e.),

$$d = ||y||_2 \tag{4.2}$$

and P is an mxm orthogonal matrix. This reduction can be achieved by using a sequence of Givens rotation matrices P_j^i which rotate vectors only with respect to the (i - j) th plane. A Givens matrix P_j^i is defined as an identity matrix, except that the i-th diagonal element is replaced by c, the j-th diagonal element replaced by -c, and the (i,j) and (j,i) elements replaced by h, where c and h are scalars such that $c^2 + h^2 = 1$. Note $P_j^i = (P_j^i)^T$. The scalars c and h can be determined from the fact that when P_j^i is applied to the vector y, it forms an m vector $\bar{y}$, which is the same as y, except that $\bar{y}_j = 0$ and $\bar{y}_i$ is replaced with $\pm(\,(y_i)^2 + (y_j)^2\,)^{1/2}$, so that $\bar{y}$ has the same Euclidean length as y. Daniel, Gragg et al (6) describe a way to compute the scalars c and h of a Givens matrix which avoids the artificial problems of overflow and underflow along with the determination of the sign of $\bar{y}_i$.

There are several sequences of Givens matrices which will achieve the reduction (4.1). For

example,

$$P_2^1\ P_3^2\ \cdots\ P_{m-1}^{m-2}\ P_m^{m-1}\ y = d\ e_1 \tag{4.3}$$

Due to the structure of P_{j+1}^j, we only need to store the two scalars c and h of that Givens matrix rather than the whole matrix. Moreover, the product of the Givens matrices in (4.3) is an upper Hessenberg matrix. The special structure of such product matrices is shown in Gill, Golub, et al (13). They also provide the necessary recurrence relations, so that, one could calculate the product of the Givens matrices in (4.3) from the coefficients. Later we present an alternate implementation of the multiplication of a vector or a matrix with the Givens matrices which is more efficient and avoids the zero divide problems. Now we will present a method that uses Givens matrices to update the factors Q and R, similar to the one discussed in Goldfarb(17).

Adding a constraint.

We will assume that we know the factors of B as defined in (2.5). That is, we have

$$B = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = L^{-1}N \tag{4.4}$$

Adding a constraint p, with normal n^+, to the active set A, we have the factorization

$$B^+ = (B\ b^+) = Q^+ \begin{bmatrix} R^+ \\ 0 \end{bmatrix} \tag{4.5}$$

where $b^+ = L^{-1}n^+$. We can write

$$Q^T B^+ = (Q^T B\ \ Q^T b^+) = \begin{bmatrix} R & Q^T L^{-1} n^+ \\ 0 & \end{bmatrix} \tag{4.6}$$

where $Q^T L^{-1} n^+$ is an m vector, necessary for calculation of $N^* n^+$ and H n^+ as shown in section 2. Let Y be the product of the Givens matrices required to make $Q^T L^{-1} n^+$ the q+1 st column in the upper triangular matrix R^+, by reducing the last m-q-1 elements to zero. The first q rows and columns of Y are identical to those of an identity matrix and the submatrix consisting of the rows and columns q+1 through m denoted as $\bar{Y}$ is upper Hessenberg. i.e.,

$$Y = \begin{bmatrix} I_q & 0 \\ 0 & \bar{Y} \end{bmatrix} \tag{4.7}$$

Premultiplying both sides of (4.6) by Y we obtain,

$$Y\ Q^T B^+ = Y \begin{bmatrix} R & Q^T L^{-1} n^+ \\ 0 & \end{bmatrix} = \begin{bmatrix} R^+ \\ 0 \end{bmatrix} \tag{4.8}$$

and

$$Y\ Q^T = Q^{+T} \tag{4.9}$$

We can easily verify (4.5) from (4.8) and (4.9).

One can store and update $J = Q^T L^{-1}$ instead of Q. This can be accomplished as

$$Y\ J = J^+ \tag{4.10}$$

and

$$Y \begin{bmatrix} R & Jn^+ \\ 0 & \end{bmatrix} = \begin{bmatrix} R^+ \\ 0 \end{bmatrix} \tag{4.11}$$

We can verify that $J^+N^+ = \begin{bmatrix} R^+ \\ 0 \end{bmatrix}$. Updating of J results in fewer operations in the calculation of $Q^T L^{-1} n^+$ and the computation of the projected normal and the multipliers. Therefore, in our computer program we store J and R rather than Q and R.

Deleting p-th constraint, $1 \leq p \leq q$.

Removing the p-th column of N to form N^- is equivalent to removing the p-th column from R in (4.4). This results in a matrix with a subdiagonal in columns p through q-1. We will denote this matrix of R with p-th column removed as $\hat{R}$. The subdiagonals of size q-p of $\hat{R}$ can be reduced to zero by multiplying $\hat{R}$ by appropriate Givens rotation matrices. This multiplication will only affect the rows p through q-1 of $\hat{R}$. Let Y be the product of the Givens matrices required to produce the desired reduction. It can be shown that Y has the form

$$Y = \begin{bmatrix} I_{p-1} & 0 & 0 \\ 0 & \bar{Y} & 0 \\ 0 & 0 & I_{m-q} \end{bmatrix} \tag{4.12}$$

where $\bar{Y}$ is a (q-p+1)x(q-p+1) lower Hessenberg matrix. We can write

$$Y\,Q^T L^{-1} N^- = Y \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} = \begin{pmatrix} R^- \\ 0 \end{pmatrix} \tag{4.13}$$

and

$$Y\,Q^T = (Q^-)^T \tag{4.14}$$

Again, we can verify that $(Q^-)^T B^- = \begin{pmatrix} R^- \\ 0 \end{pmatrix}$.

One can update $J = Q^T L^{-1}$ instead of Q^T in a like manner by replacing Q^T in (4.14) by J, $(Q^-)^T$ by J^- and $Q^T L^{-1}$ in (4.13) by J.

Multiplication with Givens Matrices

The updating of the factors is the most time consuming part of any projection algorithm and hence it is essential that these routines be very efficient. The upper Hessenberg matrix of the product of the Givens matrices shown in (4.3) can be obtained by using the recurrence relations shown in Gill, Golub, et al (13). These recurrences determine two vectors and any element of the matrix $\bar{Y}$ can be determined from these two vectors and/or the Givens matrix coefficients. This is convenient when one wants to know the product matrix explicitly. The recurrence relations however, break down when implemented in a computer program when one of the Givens matrix coefficient is zero, resulting in zero-divide problems, even though they are algebraically correct. We will present an efficient implementation of updating the factors, without calculating the product of Givens matrices. Since there is no division with possible zero in the denominator involved, it can be implemented in a computer program with no checking for zero-divide. To illustrate, let us consider the problem of reducing a vector of two elements, w_1 and w_2 into w and 0. The Givens matrix coefficients required to produce this reduction are c and h given by

$$w = (w_1^2 + w_2^2)^{1/2}$$

$$c = w_1/w$$

$$h = w_2/w.$$

Define $$y = w_2/(w_1 + w) \tag{4.15}$$

(If $w_1 = w_2 = 0$, then there is no reduction necessary.) The above has no zero-divide possibility. Assuming c, h and y are available, then consider the problem of determining the product of a two element vector $(v_1\ v_2)^T$ with the Givens matrix. This product can be accomplished in 3 multiplications rather than 4 and is shown in Gill, et al (13) as follows:

$$\begin{bmatrix} c & h \\ h & -c \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \end{bmatrix} \tag{4.16}$$

$$\bar{v}_1 = v_1 c + v_2 h \tag{4.17}$$

$$\bar{v}_2 = v_1 h - v_2 c$$

$$= (v_1 + \bar{v}_1) y - v_2 \tag{4.18}$$

The series of products as shown in (4.3) can be calculated from right to left in the above manner at the cost of 3(m-1) multiplications. The additional work required to calculate the y vector is (m-1) multiplications. Thus, one can update m column vectors in the above manner at the cost of 3m(m-1) + m-1, excluding the cost of calculating the Givens matrix coefficients, c and h. Another approach is described in Gill, Murray, et al (14) for finding the product with Givens matrices using formulae similar to the ones described here. However, rotations are always between j-th and m-th variables as opposed to j-th and j+1-st.

X-correction and Reinversion

In our implementation of the algorithms, whenever a constraint is added to the basis, a check is made to make sure that all the basis constraints are in fact active to some tolerance. A correction procedure may be needed to make sure x is on the desired manifold. This procedure is called as x-correction and is achieved by implementing lemma 3. This procedure will work as long as the errors in the factors are not significant. When they are significant, it may be necessary to recompute the factors from scratch. This is quite expensive and is called reinversion. No x-correction or reinversion was invoked for all problems tested with our numerically stable implementation using J and double precision arithmetic on IBM 370/168. This was true for all the primal and dual methods.

5. RELATIONSHIP TO OTHER METHODS AND LINEAR PROGRAMMING (LP)

Associated with every primal problem as stated in (2.1), (2.2) there exists a corresponding dual problem as shown in Dorn (9), with both problems possessing the same solution, if there is one. This is true for LP as well. There are two methods of solving the above problems and they are called primal and dual methods. It has been shown in the literature that for LP, the primal method on the primal problem and the dual method on the dual problem are equivalent. Similarly, for LP, the primal method on the dual problem and the dual method on the primal problem are equivalent. Van de Panne and Whinston (38) VW, have shown that the primal method (simplex-like) for QPP is a straightforward extension of the simplex method for LP. VW have also shown that applying the primal method on the dual problem of the original QP, is straightforward extension of the dual simplex method for LP. Goldfarb (16) shows that the Dantzig-Wolfe simplex-like method for QP takes the same path as one of the primal projection methods when started from same point basis. This primal projection method is also identical to Fletcher's (12) QP algorithm for positive definite case. The computational requirements are also shown in (16) for the tableau and projection methods. The simplex-like methods of Dantzig-Wolfe and VW are identical. We have solved the primal problem with a dual projection method. We will show that this approach takes the same path as the dual tableau method of VW (See Appendix C) which solves the dual problem. We will also present the computational requirements for both the methods.

Dual Problem

The dual problem of the problem (2.1), (2.2) can be written as shown in Dorn (9) as

$$\textit{maximize} \quad b^T u - 1/2\, x^T G\, x \tag{5.1}$$

$$\textit{subject to} \quad G\, x - C\, u = -a \tag{5.2}$$

$$u \geq 0 \tag{5.3}$$

The Lagrangean of the above problem can be written as

$$L(u,x,w) = b^T u - 1/2\, x^T G\, x - w^T(Gx - Cu + a) \tag{5.4}$$

where w is the LM of the constraints in (5.2). Setting the partial derivative of L(u,x,w) w.r.t. x, to zero, we find that

$$x = -w \tag{5.5}$$

The partial derivative of L(u,x,w) w.r.t. w when set to zero yields (5.2) and the partial derivative w.r.t. u gives, using (5.5),

$$b - C^T x = -s \tag{5.6}$$

Since u $\geq$ 0, the K-T optimality conditions for the above dual problem should satisfy (5.2), (5.3), (5.6) and (5.7).

$$s \geq 0 \tag{5.7}$$

In addition, the complementarity conditions (5.8) must also be satisfied.

$$u^T s = 0 \tag{5.8}$$

It is important to note that the above constraints for the dual problem are a restatement of the feasibility and optimality conditions of the primal problem. It can be shown (see Dorn(9)) that the function values are the same for the primal and dual problems at the solution.

Relationship to dual tableau method

We can write the conditions (5.2), (5.6) in tableau form as in T_0. The matrix C^T and the vectors s, u and b are partitioned into two parts, with the first one containing q rows and the second part containing the remaining k-q rows. They are indicated by superscripts for the vectors s, b and u, whereas the two parts of the matrix C are denoted by N_1 and N_2. Tableau T_0 is the setup tableau in VW. A basic solution to T_0 can be obtained by making x, s^1, and s^2 basic. That is, by defining B_1 to be the basic matrix, the basic feasible solution can be accomplished by premultiplying T_0 by B_1^{-1}. This basic solution is shown in $T_1 = B_1^{-1} T_0$.

variables	x	s^1	s^2	u^1	u^2	=	value
	G	0	0	$-N_1$	$-N_2$		-a
	N_1^T	-I	0	0	0		b^1
	N_2^T	0	-I	0	0		b^2

Tableau T_0.

$$B_1 = \begin{pmatrix} G & 0 & 0 \\ N_1^T & -I & 0 \\ N_2^T & 0 & -I \end{pmatrix} \tag{5.9}$$

$$B_1^{-1} = \begin{pmatrix} G^{-1} & 0 & 0 \\ N_1^T G^{-1} & -I & 0 \\ N_2^T G^{-1} & 0 & -I \end{pmatrix} \tag{5.10}$$

variable	x	s^1	s^2	u^1	u^2	=	value
x	I	0	0	$-G^{-1}N_1$	$-G^{-1}N_2$		$-G^{-1}a$
s^1	0	I	0	$-N_1^T G^{-1}N_1$	$-N_1^T G^{-1}N_2$		$-N_1^T G^{-1}a - b^1$
s^2	0	0	I	$-N_2^T G^{-1}N_1$	$-N_2^T G^{-1}N_2$		$-N_2^T G^{-1}a - b^2$

Tableau T_1.

variable	x	s^1	s^2	u^1	u^2	=	value
x	I	$-D^T$	0	0	$-EN_2$		$D^Tb^1 - Ea$
s^2	0	$-N_2^TD^T$	I	0	$-N_2^TEN_2$		$N_2^T(D^Tb^1-Ea)-b^2$
u^1	0	$-(N_1^TG^{-1}N_1)^{-1}$	0	I	DN_2		$(N_1^TG^{-1}N_1)^{-1}b^1+Da$

Tableau T_2.

Tableau method	Projection method
E	H
D	N^*
u^1	u
s^2	s(K\A,x)
x	x
s^1	s(A,x) = 0
u^2	LM of inactive = 0 constraints

Table 5.1

Relationship between the tableau and projection methods.

Tableau T_1 is the initial tableau (i) of VW and corresponds to the unconstrained minimum in our projection method. We can arrive at a standard tableau as defined by VW with x, s^2, u^1 basic and s^1, u^2 as nonbasic from tableau T_0 or T_1 by a sequence of simplex pivots or by premultiplying T_0 by the inverse of the basis matrix. We will call this tableau T_2. The inverse of this basis B_2 is shown in (5.11).

$$B_2^{-1} = \begin{bmatrix} E & D^T & 0 \\ N_2^T E & N_2^T D^T & -I \\ -D & (N_1^T G^{-1} N_1)^{-1} & 0 \end{bmatrix} \tag{5.11}$$

where $D = (N_1^T G^{-1} N_1)^{-1} N_1^T G^{-1}$, and $E = G^{-1} - G^{-1} N_1 D$. Tableau T_2 is the initial tableau (ii) in VW. In the basic solution corresponding to T_2, $s^1 = 0$, (i.e.) $N_1^T x = b^1$, therefore, $N_1 = N$, the normals of q linearly independent active constraints in our case. x corresponds to the minimum of the objective function on the manifold of the intersection of the q active constraints as shown in theorem 1. It is easy to see that the standard tableau in VW corresponds to minimum on a manifold and the relationships between the tableau method and our projection method are shown in table 5.1.

The rule 1 in the tableau method picks a new basic variable u corresponding to a negative s^2 to enter the basis. In the projection method, this corresponds to picking a violated constraint. Rule 2 selects the variable to leave the basis from the set of basic u variables, u^1 and the primal member of the basic pair corresponding to the entering variable according to

$$\begin{aligned} &min. \quad (d_j / w_j) \\ &j \epsilon \hat{J} \\ &w_j > 0 \end{aligned} \tag{5.12}$$

where d_j is the value of the basic variable,

w_j is the value corresponding to the entering variable column and

$\hat{J}$ is the set of rows associated with the basic u variables, u^1 and the primal member of the basic pair. In the projection method, this is equivalent to calculating the step length t. Since D is N^*, a column of DN_2 corresponds to the IM w.r.t. that normal in our case. Moreover, the element of $N_2^T E N_2$ corresponding to the most negative s variable and its basic pair is $n^{+T} H n^+$ in our notation. The ratios (5.12) correspond to (3.24), (3.25) and (3.26) in the projection method. Thus, both methods result in the same step length and the same step direction. If the leaving variable is the s variable corresponding to the entering u variable, then another standard tableau is obtained. This in our method is equivalent to taking a full step,

i.e., $t = t_2$ and adding the constraint into the basis.

If the leaving variable is a u variable, this corresponds to taking a partial step in our case, (i.e.) $t = t_1$ and updating the tableau corresponds to dropping the constraint from the basis. The resultant tableau is termed a non-standard tableau in VW. At some step, therefore, suppose that we have a non-standard tableau, with x, s^2, $u_1^1, \ldots, u_{q-1}^1$ and u_1^2 basic. This tableau T_3 can be obtained by pivoting on T_2 with the last element in u_1^2 column as the pivot. The x variables corresponding to this step get changed as follows:

$$x = x + u_q^1 \, E \, d \, / \, (e_q^T D \, d) \tag{5.13}$$

where d is the first column of N_2 and e_q is the q-th column vector of identity matrix of size q. This in our notation is

$$x = x + t_1 \, H \, n^+ \tag{5.14}$$

where $t_1 = u_q^+(x)/r_q$. u_1^2 becomes $u_q^1/e_q^T Dd$, which corresponds in our method to the updating of the LM of the constraint to be added into the basis. The updating of the other u variables is the same as updating LM in our case. Thus, the step direction and length are identical and updating of the tableau is equivalent to dropping the q-th constraint from the basis. Defining $\bar{u}^1$ as u^1 with u_q^1 replaced by u_1^2 and $\bar{u}^2$ as u_2 with u_1^2 replaced by u_q^1, we find that

variable	x	s^1	s^2	$\bar{u}^1$	$\bar{u}^2$
x	I	$-\bar{D}^T$	0	0	$-\bar{E}\bar{N}_2$
s^2	0	$-\bar{N}_2\bar{D}^T$	I	0	$-\bar{N}_2^T\bar{E}\bar{N}_2$
$\bar{u}^1$	0	$-(\bar{N}_1^T G^{-1}\bar{N}_1)^{-1}$	0	I	$-\bar{D}\bar{N}_2$

Tableau T_3.

where $\bar{N}_1$ and $\bar{N}_2$ are different from N_1 and N_2 respectively in the same fashion as $\bar{u}^1$ and $\bar{u}^2$ differ from u^1 and u^2 and $\bar{D}$ and $\bar{E}$ are defined as D and E respectively with the appropriate change in N_1 (that is, N_1 in the definition is replaced by $\bar{N}_1$.). (s_1^2, u_1^2) form the basic pair and

(s_q^1, u_q^1) form the non-basic pair in this non-standard tableau. The choice of entering variable is the primal member of the nonbasic pair, namely, s_1^2 is in $\hat{J}$. This is same as calculating the step length as shown before. Therefore, the two methods will take the same path in this case. Since there can be only standard and non-standard tableau, our demonstration is complete.

Although the two algorithms follow identical paths when started from the same dual feasible point, they differ considerably in terms of the amount of work per iteration. The dual method in tableau form costs (m+k+1) (k+1)+q+1 multiplications and/or divisions per iteration. If one were to take advantage of the symmetry, the above number is reduced by k(k-1)/2. At the same time, if one were to implement the dual projection method using the projection operators, H and N^*, assuming symmetry, the number of operations are given by the following formulae.

a) iteration where a constraint is dropped

$$\frac{3}{2} m\ (m+1) + 2mq + 4m + q + 2$$

b) iteration where a constraint is added

$$\frac{3}{2} m\ (m+1) + 2mq + q + 2 + (k-q)m$$

Under the most unfavourable conditions for the projection method, (i.e.), q=m when dropping a constraint and q=0 when adding a constraint, the projection method takes fewer operations, for k slightly greater than m while dropping and for k greater than 1.73m when adding a constraint. It is interesting to note that when dropping a constraint, the number of operations do not involve k. It is because of the fact that the slack of the constraint to be added is the only one required to be known at this stage. Thus we have shown that even under the most unfavourable conditions for the dual projection method, it costs less per iteration than the dual tableau method of VW for k slightly greater than m and 1.73 m for an iteration with a drop and an iteration with an add respectively.

6. QUADRATIC PROGRAMS IN NON-LINEAR PROGRAMS (NLP)

Powell (31) presents a fast algorithm for solving general NLP based upon a variable metric approach due to Han (23,24) and Biggs (3). This NLP method solves a series of positive definite QPPs. The solution of the QPP is used to determine the step direction for the original NLP and the LMs at the solution of the QPP are used to update the positive definite quadratic form for the next invocation of the QPP. It has been shown by Powell (30,31) that the method achieves superlinear convergence. Schittkowsky (35,36) in a recent report has compared 13 different NLP codes with randomly generated test problems and comments favourably on Powell's algorithm which uses Fletcher's QP algorithm to solve the QPPs. He also notes on Powell's code: "the calculation time seems to be correlated to the number of variables and might not be most effective for solving large problems". Since our dual method is efficient for large problems (see section 8.2), we expect Powell's algorithm with our dual method for QPPs to be more effective.

Description of Powell's NLP algorithm

Consider the following general NLP.

$$\textit{minimize} \quad F(x) \tag{6.1}$$

$$\textit{subject to} \quad c_i(x) = 0, \ i=1,2,...,k' \tag{6.2a}$$

$$c_i(x) \geq 0, \ i=k'+1,...,k. \tag{6.2b}$$

where x is a vector of m real variables and F(x) is a real function. The objective and constraint functions are differentiable and the first derivatives can be calculated. Let g = g(x) be the gradient vector

$$g = \nabla_x F(x). \tag{6.3}$$

At each iteration of the NLP, a starting point x and g are computed. A positive definite matrix, G say, which is set to the unit matrix initially, defines the current metric. The vector d that minimizes

$$Q(d) = d^T g + 1/2\, d^T G\, d, \tag{6.4}$$

the quadratic function (6.4) subject to constraints obtained by linearizing the constraints in (6.2) about x, is computed. It is used as a search direction in the x variable space, x being replaced by the vector x',

$$x' = x + y\, d \tag{6.5}$$

where y is a positive multiplier whose value depends on the form of the function of one variable,

$$\psi(y) = F(x + y\, d) \tag{6.6}$$

The matrix G is revised, using gradients g and $g' = \nabla F(x')$ and another iteration is begun.

The constraints for the QPP at each iteration can be formulated as follows:

$$\nabla c_i^T d + c_i = 0,\ i = 1,2,\ldots,k'$$

$$\nabla c_i^T d + c_i \geq 0,\ i = k+1,\ldots,k \tag{6.7}$$

where $c_i = c_i(x)$. It may happen that the above linear constraints (6.7) on d are inconsistent. This does not necessarily imply that the constraints in (6.2) are inconsistent. Therefore, an extra variable v, is introduced into the QP calculations and replace the constraints (6.7) by the conditions

$$\nabla c_i^T d + c_i v = 0,\ i = 1,2,\ldots,k'$$

$$\nabla c_i^T d + c_i v_i \geq 0,\ i = k'+1,\ldots,k \tag{6.8}$$

where v_i has the value

$$v_i = 1,\ c_i > 0$$

$$v_i = v,\ c_i \leq 0 \tag{6.9}$$

and v is made as large as possible subject to the condition

$$0 \leq v \leq 1. \tag{6.10}$$

Any positive value of v allows a helpful correction to x of the form (6.5). For more detailed description of the method see Powell (29,30,31).

This new variable v appears only in the linear part of the objective function of the QP in Powell's original code which uses Fletcher's QP algorithm. This linear term is given a very high penalty so that the v variable will take a value as close to 1 as possible. Since the dual method requires a positive definite quadratic form, the objective function is modified in both the quadratic and linear terms in the form of $1/2\, p\, v^2 - 2\, p\, v$, rather than - p v, as in Powell's original code, where p is a very large positive number, say 10^6. This modification does not affect any iterations of the NLP algorithm or Fletcher's QP.

Numerical results

Before we present various results, it is important to note some major differences in the implementation of Fletcher's QP algorithm and our dual projection algorithm used in Powell's algorithm described earlier. The NLP iterations however, were essentially the same for both.

Differences in implementation

The interface between the NLP code and the QP code is different in the sense that the original code using Fletcher's QP requires upper and lower bounds on all the variables of the QPP, whereas, for the dual QP, no such bounds are necessary except for the new variable v whose bounds are treated as constraints.

The LM is readily available at the solution of the dual method and is calculated in Powell's original code after the problem is solved.

Fletcher's QP method was implemented using the projection operators H and N^*. The dual method used the numerically stable implementation discussed in section 4. Number of operations (multiplications and/or divisions) required to update the operators H and N^* when there is a change in the basis equals (assuming symmetry)

$$(3/2)\ m\ (m+1) + m + 2\ m\ q,$$

where q is the number of active constraints in the current basis. The corresponding formulae for updating the factors in the dual method are

$$4\,m^2 - 3\,m\,q + m - 4\,q - 4,$$

when adding a constraint and

$$(q - p)\ (3\ m + 3\ (q\text{-}p\text{-}1)/2 + 4)$$

when dropping the p-th constraint. The formulae for the factors assume that $Q^T L^{-1}$ is stored and updated rather than Q.

Fletcher's QP was used with mode = 1, which means that a general QP is being solved and the inverse of $\begin{pmatrix} G & N \\ N^T & 0 \end{pmatrix}$ is computed to check the feasibility and optimality conditions at the solution, where N is the matrix of active normals. Dual method does not compute such inverse at the solution.

An LP routine is used in Fletcher's QP algorithm to find an initial feasible vertex. At a vertex, the projection operators are just H = 0 and $N^* = N^{-1}$. Moreover, the old basis is included in the infeasible basis at the start of the feasible point routine. The dual method was implemented in two different ways. In the first implementation preference was given to the active constraints of the previous iteration when adding constraints to the basis, while the second implementation ignored any such information from the previous iteration. The first type was found to be approximately 20% less expensive than the second and resulted in fewer drops. The results presented are for the dual method of the first type. It was observed that the intersection of the active constraints at the successive iterations of the QPP was very high and should help Fletcher's QP algorithm substantially. In fact, it was 100% for four of the six problems and over 90% for the other two problems.

Problems tested

We solved six NLP problems with the necessary modifications to Powell's code which uses

Fletcher's QP algorithm and our dual projection method. We report here on our experience with a problem suggested by Powell (1969) (31), Rosenbrock's post office parcel problem (POP), Powell's triangle problem (29) and three of Colville's test problems (see Dembo(8) and Himmelblau(25)). Their characteristics are as shown in table 6.1 in appendix D.

Discussion of results

The last column in table 6.1 shows the average cardinality of the active basis at the solution of the QPPs and the column of the variables under the QP characteristics indicates the maximum possible active constraints at the QP solution (in other words, for a vertex to be the solution). The manifold on which the QP solutions occur is always a vertex in two of the six problems and no more than 2 dimensional manifolds in the remaining four problems.

The dual method outperformed the primal method in all cases in terms of the total number of operations. Fletcher's (10,11) QP and LP codes do not require any square root evaluations. In most cases, the dual method required only a little more effort to solve the problems than the feasible point routine (FP) in the primal method. The ratio of the number of operations for FP to the dual (including sqpare roots at 10 operations) ranges from 0.38 to 1.19 with an average of 0.77. This ratio is the lowest for Powell's problem which has only equality constraints. The range of the same ratio for FQ (Fletcher's QP routine) to dual is 1.44 to 2.57 with an average of 1.91. This range is 1.85 to 3.33 for (FP+FQ)/D with an average of 2.7. It is the lowest for Powell's problem (1.85) and is the highest for Colville 2 (3.33). For triangle problem it is 3.21. It is interesting to note that this ratio is high for large variable problems indicating that the dual is more efficient for large problems.

The number of steps in the dual is more than the primal method, mainly because the series of QPPs generated are such that they favour Fletcher's QP method and the feasible point routine. Even under such circumstances, the dual is more efficient. The primal hardly ever adds while the dual hardly ever drops.

7. STEEPEST EDGE OR FACE CONSIDERATIONS

In primal algorithms one must, at certain points, decide which constraint(s) in the active set to drop. The usual practice is that the constraint corresponding to the most negative LM is dropped. This choice is also the simplest computationally. One could drop a constraint among the eligible constraints in such a way that the rate of decrease of the objective function in an appropriate metric is the greatest. If that metric is G, it has been shown in Goldfarb (19) that the constraint to be dropped can be found for primal algorithms as the one that maximizes the following function.

$$\underset{u_j > 0}{max.} \left\{ u_j^2 / e_j^T (N^T G^{-1} N)^{-1} e_j \right\} \tag{7.1}$$

Since $(N^T G^{-1} N)^{-1}$ is not stored or updated, it can be calculated from N^* as shown in (7.2).

$$(N^T G^{-1} N)^{-1} = N^* G\ N^{*T} \tag{7.2}$$

Similar method for LP is described in Goldfarb (18). Even though we only require the diagonal elements of (7.2), the above process is quite expensive. It is therefore convenient to update only the required elements of (7.2) whenever a basis change takes place. This can be done at the cost of $O(n^2)$ operations as shown in Goldfarb (20).

Adding a constraint to the basis

Let h_i be the i-th diagonal element of (7.2). The recurrences for updating h when a constraint whose normal is n^+ is added to the basis is given by

$$\begin{aligned} \bar{h}^+ &= 1/\ n^{+T} H\ n^+ \\ \bar{h}_i &= h_i + r_i^2\, \bar{h}^+ ,\ i = 1,2,...,q \end{aligned} \tag{7.3}$$

where r_i is the i-th infeasibility multiplier w.r.t. n^+ and $\bar{h}$ is the updated h. This updating can be accomplished at the cost of $m^2 + mq + m + 2q + 1$ multiplications.

Dropping a constraint from the basis

Let us define a q vector v as follows:

$$v = N^* G \; n^{-*T} \tag{7.4}$$

Then

$$\bar{h}_i = h_i - v_i^2 / h^- \tag{7.5}$$

where h^- is the value of h corresponding to the constraint to be dropped. This updating costs $m^2 + mq + 2q - 2$ operations.

The above updating formulae use the projection operators H and N^*. However, if one were to use the orthogonal factors discussed in section 4, the corresponding updating can be done as shown below.

Adding a constraint with factors.

Compute $w = Q^T L^{-1} n^+$. Let us denote the first q elements of w as w_1 and let w_2 be the length of the vector comprising of the last m-q elements of w. Then the vector h can be updated as

$$\bar{h}^+ = (1/w_2)^2 \tag{7.6}$$

$$\bar{h}_i = h_i + e_i^T R^{-1} w_1 w_1^T R^{-T} e_i \; \bar{h}^+, \; i = 1,2,...,q$$

This costs q(q+1)/2 +2q+2 multiplications assuming that w_1 and w_2 are readily available.

Dropping a constraint with factors

If p-th constraint is being dropped, where p $\leq$ q, the vector

$$v = R^{-1} R^{-T} e_p \tag{7.7}$$

can be computed and formula (7.5) is now applicable. This costs q(q+1)/2 + 2q - 2 + (q-p+1) (q-p+2)/2 operations.

Greatest increase choice in the dual method

There is no choice to be made of the constraint to be dropped from the basis in the dual method. However, it is possible to choose the constraint to be added. One possibility is to choose the one that will produce the greatest increase in the objective function if a full step is taken. This choice of adding a constraint is made after a full step is taken or at the beginning of the algorithm. As in the primal methods, this does not necessarily, theoretically decrease the number of steps or basis changes needed to obtain the solution.

If a full step is taken, the change in the objective function is given by

$$(s(\{p\},x))^2/\ n^{+T} H\ n^{+} \tag{7.9}$$

when $|Hn^{+}| \neq 0$. Using the projection operators, this will cost $m^2 + m + 2$ operations, assuming the s vector is available for every violated constraint. This adds considerably to the cost in the dual method. Therefore, one could apply this method of picking the constraint after the basis is almost full.

When $|Hn^{+}| = 0$, the corresponding forula to (7.9) is given by

$$1/2\ (s(\{p\},x)^2\ /\ n^{+T} H^{-} n^{+}) - s(\{p\},x)\ u^{-}/r^{-}. \tag{7.10}$$

where u^{-} and r^{-} are the LM and IM corresponding to the constraint to be dropped as shown in theorem 3.

8. PERFORMANCE COMPARISON WITH OTHER METHODS

In this section, six primal methods and the dual method are compared with respect to the number of steps, basis changes and operations. (multiplications and divisions and square root calculations, with 1 square root = 10 multiplications) All of the primal methods were started from the same feasible point for each given problem. The feasible point routine started with a random starting point, and had no control of the first feasible basis. One could also control the basis for the first feasible point by applying Lemma 3, but, this was not done. The average intersection of initial and final bases was found to be 0.5 times the number of constraints active at the solution; (i.e.) on an average, half of the final basis appeared in the initial basis. The feasible point routine used the algorithm suggested in Goldfarb (16), which is described in detail in Idnani (26). It is a generalization of a feasible point routine given by Rosen (33) applied to QPP. The dual method always used the unconstrained minimum as the initial point. The computer program was capable of solving the QPP with six primal-dual methods as well, one for each of the primal method. Thus, the program could run 14 different algorithms. The program listing is in appendix E.

The six primal methods are variations of the two primal methods of Goldfarb (16), one of which is identical to the one by Fletcher (12). (See Appendix A) We will denote these two primal methods as P.1 and P.2. (P.1 is identical to Fletcher's QP.) The difference between P.1 and P.2 is that, P.2 allows more than one constraint to be dropped at a point if certain additional conditions on the LM holds. The sufficient condition on the additional requirements is that the LM of a constraint to be dropped should be monotonically non-increasing as constraints are dropped at the same point. Using the steepest face considerations for primal methods, one could drop the constraint among eligible set as the one suggested in section 7.(see 7.1) This produces two more primal methods and we call them P.1.1 and P.2.1. Methods of type P.2 take steps on higher dimensional manifolds than methods of type P.1. The remaining two primal methods are variations of the primal method of type P.2 and are based on the necessary and sufficient conditions for dropping more than one constraint at a point. This

ensures that by dropping one of the currently active constraints, the step direction is such that any previously dropped constraint is not violated. These and the negativity of the LM are the necessary conditions for dropping more than one constraint. Since the step direction in the primal methods is the projected gradient, this can be calculated by updating recurrences as shown in Goldfarb (20). This uses similar information as the steepest face algorithms, namely $R^{-1}R^{-T}$. So, one could derive a two phased primal algorithm for the dropping part of methods of type P.2 as follows.

Drop phase 1: Use the sufficient conditions for dropping more than one constraint in addition to LM being negative. When none can be dropped further, apply drop phase 2.

Drop phase 2: Among eligible constraints (all negative LM constraints are eligible), drop the one that would not result in a step direction which would violate a previously dropped constraint in both drop phases so far. If none can be dropped, take step, otherwise repeat drop phase 2.

Since this two phase approach can be applied to P.2 and P.2.1 only, we will denote these as P.2' and P.2'.1 respectively. Table 8.1 summarizes all the six primal methods.

The letter D will stand for the dual method and the prefix D. followed by any one of the primals, for the primal-dual method corresponding to the primal used. The letter F will denote the feasible point routine. The primal-dual methods were only verified for convergence. Rigorous testing was done for all the primal and dual methods.

Problem generation

For performance comparison, 24 different problem types with known solutions were constructed. These problems were randomly generated as described in Idnani (26) and in appendix B. The method used was the one by Rosen and Suzuki (34) modified for QPP. The problems generated had 9, 27 or 81 variables. The positive definite matrix G was generated in two different ways, such that G was either well conditioned or ill-conditioned. In the well

conditioned case, the eigenvalues were close and in the ill-conditioned case, they were spread. In either case, the off diagonal elements are generated first using random numbers between -1 and +1. Then the i-th diagonal element is calculated as follows:

Well conditioned G: $G_{ii} = RS(i) + R(0,1) + 1$

Ill conditioned G: $G_{ii} = RS(i) + R(0,1)$ for $i = 1$.

$$G_{ii} = RS(i) + R(0,1) + G_{i-1,i-1} + RS(i-1),\ i \neq 1$$

where RS(i) is sum of absolute values of the off diagonal elements of the i-th row of G, and R(a,b) is a random number between a and b.

	Author	symbol	remarks
1.	Fletcher & Goldfarb	P.1	at most one drop before taking step. Drop most negative LM constraint.
2.	Goldfarb	P.1.1	same as P.1, except, use steepest face considerations for drop.
3.	Goldfarb	P.2	Apply sufficient conditions for multidropping, drop most negative LM constraint.
4.	Goldfarb	P.2.1	same as P.2, except use steepest face considerations also.
5.	Goldfarb	P.2'	same as P.2. Once sufficient conditions exhausted for drop, apply necessary conditions.
6.	Goldfarb	P.2'.1	same as remarks for P.2', replace P.2 by P.2.1.

Table 8.1 Various Primal Methods.

The largest possible diagonal element in the well conditioned case is m+2. In the ill conditioned case, it is 13202 for the 81 variable problem. But, the expected value of these numbers is only half of the maximum. The maximum value of i-th diagonal element in the ill

conditioned case is given by (2i-1) (m+1).

The number of constraints for an m variable problem were tested at two levels, k = m and k = 3m. Thus the 81 variable problems had a set of problems with 81 constraints and another with 243 constraints. The number of active constraints at the solution was also kept at two levels, k/9 and k/3. The problems at k/9 level had close to an interior point as the solution, whereas, the problems with k/3 level could have a vertex as the solution. Thus, in all, 24 problem types as shown in table 8.2 were generated. (See appendix D.)

The coefficients in the constraint normals were generated between -1 and +1. The components of the solution vector x were generated between -5 and +5. The constraint normals were normalized. The slacks of inactive constraints lie between 0 and 1. The LMs of active constraints at the solution were set randomly between 0 and 30 for run 1, for types of problems 1 to 16. For run 2, for the same set of problems, the LMs for active constraints were randomly generated between 0 and 30k, where k is the number of constraints for the problem. The difference between runs 1 and 2 is that the run 2 has more linear objective function than run 1. Run 3 comprises of 81 variable problems with LMs between 0 and 81k. The K-T conditions are used to calculate the vector a. Runs 1 and 2 were replicated five times, whereas, run 3 was replicated only once. (i.e.) 5 different seeds were used to generate a problem type which was then solved. Totally, 168 problems were generated. All the primal methods described earlier, the dual method and a primal-dual method using primal of type P.1 were tested in all the runs except in run 3, which did not test the primal-dual method.

Observation

The dual method required fewer operations in most problems than the feasible point routine used for the primal methods (see appendix D). In addition, the primal methods take at least as many operations as the feasible point routine. The only cases where the feasible point routine took fewer operations than the dual were for problems of type 7 and 8 for runs 1 and 2, and type 4, 15 for run 2 and 23 for run 3. The problems of type 7, 8, 15 and 23 have a vertex as

the optimum solution. Problem types 7, 15 and 23 also have well conditioned G, whereas, 4 and 8 have ill conditioned G. The condition of G does not have any effect on the feasible point routine. Surprisingly, the dual method is sensitive to the condition of G in runs 2 and 3 only. In these runs, where the objective function is more linear, the problems with ill conditioned G take fewer operations than those with well conditioned G. In the problems with a vertex as the solution, the difference is quite significant. This strange behaviour of the dual method can be explained as follows.

Problems of runs 2 and 3 have a heavily weighted linear function. The gradient of the function is written as Gx + a, can be thought of as the sum of two vectors. One of them is the constant vector a and the other is the variable vector Gx. For well conditioned G, the constant vector dominates over the variable vectors in runs 2 and 3. For the same runs when G is ill conditioned, the contribution by the variable part Gx could be significant, because G has positive diagonals with much higher value than the off diagonals. In our dual method, all along the step direction, $H^{+}g = 0$ which implies that the gradient can be written as a linear combination of the LMs, of the normals. When the change in g is significant, the LMs vary significantly. The rate of change of LM with IM is what determines t_1, the partial step length. In other words, when there is little change in g, the LMs remain high permitting full steps, whereas, when the gradient changes rapidly, the LMs change rapidly and this results in partial steps. That is, the contribution of the curvature information in the well conditioned case is not as high as in the ill-conditioned case. Another explanation due to Goldfarb can be described as follows: Let x^* and x^0 be the constrained and unconstrained optimums for (2.1), (2.2). Let $d = x^* - x^0$ and $y = \|d\|_2$. Then, $d = G^{-1}\bar{N}\bar{u}$, where $\bar{N}$ is the matrix of active normals at x^* and $\bar{u}$, the associated LMs. y is largest when G is well conditioned, $\bar{N}$ is full and $\bar{u}$ is large and these precisely are the problems which were most difficult to solve for the dual method. This is because the dual method is more likely to take full steps when y is large. Run 2 has a higher $\bar{u}$ for the same problem type than run 1, and therefore, y is greater.

An interesting property of the dual method is that it remains on as high dimensional manifolds as possible. Of the 168 problems tested, it picked the correct basis without dropping in 110 problems (see tables 8.6 and 8.7). This was also confirmed in the NLP runs (see table 6.2). These tables appear in appendix D.

Table 8.8 shows the number of active constraints at the first feasible point for the primal methods and its intersection with the final basis. On an average half of the final basis appeared in the initial one.

Table 8.9 shows the average of the total number of basis changes per step for all the runs. To some extent, this ratio indicates the dimension of the manifold on which steps are taken. When this ratio is 1, for the dual, primal-dual and feasible point methods, it indicates that steps are taken on high dimensional manifolds. For primal methods, the higher the ratio, the more interior or the higher dimensional manifold, on which steps are taken. Methods of type P.2 have higher ratio than methods of type P.1. the highest ratio is for P.2'.1, but, there is an additional computational requirement to achieve this increase over P.2 as shown in tables 8.3 - 8.5. As expected, we see that this ratio for the dual method is less than that for the feasible point routine and is slightly greater than 1.

Table 8.10 shows the effect of the objective function's weighting on the linear term. When the function is made more linear, as in runs 2 and 3, the dual and primal-dual methods experience a much greater increase in the number of steps and basis changes than the primal methods. A substantial part of the increase comes from the problems where the solution is a vertex with a well conditioned G. Among the primal methods, P.1.1 decreased in number of steps, while the number of basis changes increased. Generally speaking, the linearity of the function had some effect on all the primal methods, but not to the same degree as in the dual methods.

Steepest face considerations

As shown in section 7, the steepest face considerations cost a little more at the time of updating the basis. Since only the diagonal elements of $(R^TR)^{-1}$, corresponding to the negative LMs or some other criterion which is even more stringent, are required, the recurrence relations as described in section 7 were not used in our computer program. Instead, only the necessary diagonal element was computed by first finding the appropriate row of R^{-1} and taking the inner product of it with itself. This can be accomplished for the p-th diagonal at the cost of $y(y+3)/2$ operations, where $y = q-p+1$. It is not clear which method is better. If one has to calculate even a few diagonal elements, it is possible that it can become expensive. The cost mainly depends upon q-p. If one were to recur, either, one has to recur during the feasible point routine as well or can calculate it after the feasible point is found.

An alternative to recurring the diagonal elements for methods of type P.2, is that, after the first drop takes place, one could recur only the diagonals of the eligible set of constraints with the idea that the eligibility set is very small. The steepest face directions do result in fewer steps and basis changes for all primals. This is not so significant in the primals of type P.2 and P.2' as in P.1. Method P.2 produced on the average the best results overall among all the six different primal methods tested.

Primal-dual methods

The primal-dual methods as implemented in our program must be modified a little in order to repeat the multiplier computations in the primal or dual part. These methods always start from the unconstrained minimum, although in general, they do not have to. The computer program must be modified accordingly for general use. Since the various primal-dual methods behaved very similarly when started from the unconstrained minimum, only D.P.1 was run. A peculiar phenomenon of dropping and adding occured in the primal part of the D.P.1 method. It can be explained as follows:

Assume that x solves a subproblem of q active constraints, say, 1,2,...,q. Let q+1 be a linearly independent violated constraint. The D.P methods will take a full step and add the q+1-st constraint to the basis in the dual part of the method. Now let the LMs of constraint 1 and 2 become negative. Therefore, the primal part is invoked to solve this problem of q+1 constraints. Assuming LM of constraint 2 to be the most negative, the primal method would drop the constraint 2 and take a full step to the minimum on the new manifold. Now, if LM of constraint 1 is negative, it would drop 1 and take a step. However, if constraint 2 will be violated in the new step direction, only a partial step can be taken and constraint 2 added back into the basis. The dual method unlike the primal-dual method, under the same circumstances, would have taken a partial step and dropped constraint 1 and then taken a full step to add constraint q+1. The difference between the dual and the primal-dual can be explained by the fact that the dual does not drop constraints according to the most negative LM, but, by the order in which they become negative in the step direction.

Summary

As far as the number of operations are concerned, the dual method required fewer operations overall for all problems. It is not surprising that the ratio of the number of operations for the primal methods to the dual method is quite high even under the most unfavourable conditions for the dual method. If one were to add the cost of finding a feasible point, the dual is 1.69 to 8.55 times faster than P.1 and 1.49 to 4.98 times faster than P.2. On the average, these numbers are of the order 4.8 and 3.2 respectively. There are other methods of finding a feasible point, such as the one used in Fletcher's QP method. This method favours problems with low dimensional manifold solutions and may be more efficient than the feasible point routine used in our program. Moreover, methods of type P.2 which drop more than one constraint at a point and move across the interior, are expected to benefit when the solution occurs on a high dimensional manifold. The dual method performs quite well for large and small problems, and, it is the starting point, in our belief, that reduces the number of iterations and results in fewer drops.

9. CONCLUSION

For positive definite QP, it is clear that the dual method is quite efficient compared with primal methods even under the most unfavorable conditions for the dual algorithm. More importantly, if the solution occurs on some high dimensional manifold, it is able to pick the correct basis without dropping constraints in most cases. If the operators, H, N^* or the factors are readily available, then, in our opinion, a primal-dual method is strongly recommended. Even though any one of the primal methods is suitable, P.2 is recommended for large problems.

The dual method is more efficient in terms of the number of operations than primal methods even when it takes more steps because:

a) the gradient is never calculated if started from the unconstrained minimum.

b) the gradient is never projected in the dual, whereas it is in the primal. Projection of a constraint normal takes place in both primal and dual methods in order to update the operators or factors. This projected normal is also used as the step direction in the dual method.

c) starting at the unconstrained minimum, it remains on the infeasible side of the solution, i.e. it approaches the solution from the infeasible side of the manifold of the active constraints at the solution (whose LM are of the correct sign).

d) starting at the unconstrained minimum does not require much more additional effort.

e) step length calculations are not as expensive as in the primal methods where one has to calculate one or two inner products for the constraints not in the basis.

Another advantage of the dual method is that it has a choice of which constraint to add to the basis unlike the primal methods. Thus, it is possible to maintain a linearly independent basis to a great extent.

For our implementation, we store the matrices $Q^T L^{-1}$ and R, the vectors LM, IM and z in addition to the matrix of constraint normals, the matrix G and the vectors x, a and b. In all, it requires 3/2 m(m+1)+2m locations in addition to the storage of the problem.

For applications in Powell's NLP algorithm, it is not clear if a primal-dual method is more suitable than the dual method. Since the solution basis of the previous iteration is very much identical to the solution basis of the current iteration, a primal-dual method may be useful. But, the quadratic form is different and the constraint normals may be different at each iteration, thus the projection operators may not readily be available and hence they have to be calculated. In such a case, a matrix inversion has to take place and application of the dual as described in section 6 seems appropriate.

Choice of constraint to add

The dual method as implemented in our programs always picked the most violated constraint to add to the basis among all the violated constraints. An alternative rule was presented in section 7 as the greatest increase choice. This involves much more computational effort and there is no guarantee that it will produce a significant improvement. We will present two more alternatives in this regard for future research and experimentation. Both methods choose the constraint to be added to the basis among the violated constraints as the one whose normal makes the least angle with another "base" vector. One choice for the base vector is the sum of all the violated constraint normals. In this, the infeasibility is used as penalty term for the choice of constraint to enter the basis. Another alternate strategy which can be used, except, at the unconstrained minimum, is to choose the gradient vector as the base vector. Hybrid strategies could also be used.

Linear Programming (LP)

It is the starting point, in our belief, that resulted in an efficient dual projection method for the QPPs. A similar method for initiating the LP may be efficient. Consider the LP:

$$\textit{minimize} \quad a^T x \tag{9.1}$$

$$\textit{subject to} \quad C^T x - b = s \geq 0 \tag{9.2}$$

$$l_i \leq x \leq h_i \tag{9.3}$$

where a, x, l, h are m vectors,

C is the mxk matrix of k constraint normals,

b is a k vector and

s is the k vector of nonnegative slacks.

The constraints in (9.3) are the simple bounds on the m variables.

The above LP is just a special case of the QP (2.1), (2.2). In fact, the dual projection algorithm developed in section 3 can be applied to this LP. It is well known that the solution of an LP has to occur at a vertex. We know that at a vertex H = 0 and $N^* = N^{-1}$, where N is the matrix of m linearly independent constraint normals active at the vertex. Notice that the operators are independent of the matrix G. If one starts at a dual feasible vertex and picks a violated constraint to add to the basis (say normal n^+), one can then apply theorem 3 and obtain the constraint to drop according to the least nonnegative ratio of LM/IM, where $LM = N^{-1}a$ and $IM = N^{-1}n^+$. Since the gradient is constant, a full dual step can be taken arriving at another vertex, that is, an exchange of the constraint to drop and add is sufficient and the new x can be computed. This is the dual simplex method for LP.

One can start at a vertex which can be thought of as comparable to the unconstrained optimum in the positive definite QPP. One such vertex can be obtained by solving (9.1) w.r.t. (9.3), in other words, by picking either the lower or the upper bound of x_i depending upon the sign of a_i. In the case when a_i is zero, one of the bounds is chosen. It can be verified that the LMs at such a starting point are of the correct sign and hence, the x so chosen is a dual feasible solution. The inverse of this basis matrix is available trivially.

It is easy to see that the increase in the objective function at every step is given by $-s(\{p\},x)\ u^-/r^-$, which is nonnegative, and zero when u^- is zero, where s({p},x) is the

violation of the constraint to be added and u^-, r^- are the LM and IM of the constraint to be dropped. The greatest increase of the objective function can be obtained by maximizing the above function with respect to all the constraints that are violated.

The same technique for the starting vertex can be used for phase 1 of the primal simplex method for LP as well and is expected to reduce computational effort during phase 2. Notice that this technique requires no additional computation and only further experimentation can show the gain by such an approach. Furthermore, the recommendations made earlier for the choice of the constraint to add for the QPP are applicable for LP. One can also use the same numerically stable computer implementation used for the positive definite QPP with a minor change, that is, by setting G = L = I initially.

Dual Projection Method for Convex NLP

In the QPP, the matrix G was constant. However, for convex NLP, the corresponding matrix is the Hessian which is updated at every iteration, just like in the case of Powell's algorithm for NLP described in section 6. Moreover, the updated Hessian is always positive definite. Using ideas similar to the dual projection method for QPP, it is possible that one could develop a dual projection algorithm for solving convex NLP with linear constraints using quasi-Newton, modified Newton or Newton like methods. This is an area for future research.

General QPP

It is not clear at this time of writing how this method can be applied to QPP with positive semi-definite G or an indefinite G. One of the major problems is the starting point in these cases. This is also an area of future study.

Summary

An efficient dual projection algorithm and its implementation in a numerically stable fashion is presented for solving positive definite QPP applicable in solving general NLP problems. A primal-dual projection method is also presented. These methods derived from the primal problem has the advantage of geometric interpretation.

Appendix A

Primal projection methods.

Method P.1

Step 1) Compute some x in the feasible region, $g = Gx + a$. Compute H and N^* defined by the linearly independent defining hyperplanes on which x lies. Let A be the set of these constraints.

Step 2) Compute $z = Hg$; if $|z| = 0$ go to step 4.

Step 3) Set $x = x - tz$, compute $g = Gx + a$ and s where $t = min.\ (1,y)$

$$\text{and} \qquad \begin{matrix} y = min. \\ n_j^T z > 0 \\ j \notin A \end{matrix} \quad \left(s_j / (n_j^T z) \right) \qquad \text{(A.1)}$$

If $t < 1$, add constraint to the basis corresponding to the minimum value of y. Update A, H, N^*. Go to step 2.

If $t = 1$, go to step 4.

Step 4) Compute LM, $u = N^* g$. Determine min. $u_j = u_p$. If $u_p \geq 0$, terminate; x solves the problem. Otherwise, $u_p < 0$, drop constraint p from the basis, update A, H, N^* and go to step 2●

Method P.2

Step 1) Same as in P.1.

Step 2) Compute LM, $u = N^* g$. Set $i = 0, J_1 = A$. Determine $\min_{j \in J_1} u_j = u_p$. If $u_p \geq 0$, go to step 4. Otherwise, $u_p < 0$, go to step 3.

Step 3) Drop $p-th$ constraint from the basis, update A, H, N^* and J_1. Set $i = i + 1$. Compute LM, $u' = N^* g$. Determine $J_2 = \{j \mid u_j' \leq u_j\}$, set $J_1 = J_1 \cap J_2$ and $u \leftarrow u'$. Determine $\min_{j \in J_1} u_j = u_p$. If $u_p \geq 0$, go to step 4. Otherwise, repeat this step.

Step 4) Compute $z = Hg$. If $|z| = 0$ terminate; x solves the problem.

If $|z| \neq 0$, set $x = x - tz$, compute $g = Gx + a$ and s where $t = min.\ (1,y)$ and y is given in (A.1).

If $t = 1$ and $i = 0$ terminate; x solves the problem.

If $t = 1$ and $i \neq 0$, go to step 2.

If $t < 1$, add constraint corresponding to the minimum value of y to the basis, update A, H, N^* and go to step 2●

Appendix B

Generation of Random positive definite QPPs with known solution.

For a given number of variables m, number of constraints k, number of active constraints at the solution q^*, the positive definite QPP can be generated using the Rosen-Suzuki(34) approach as follows. (We use the expression '$\epsilon\ R(a,b)$' to mean element of a set of pseudo random numbers uniformly distributed between a and b.)

Step 1) *Solution*

a) Generate the components of an arbitrary optimum point $x^* \epsilon\ R(-5,5)$.

b) Specify the index set A of q^* active constraints at the solution.

Step 2) *Positive definite matrix G*

a) Generate the off-diagonal elements $\epsilon\ R(-1,1)$.

b) Generate the diagonal elements of G as described in section 8 for well and ill conditioned cases of G.

Step 3) *Constraint matrix C*

a) Generate the elements of C $\epsilon\ R(-1,1)$.

b) Normalize the columns of C to unit length.

Step 4) *Complementary Slackness*

a) For $j \epsilon A$, set $s_j = 0$ and generate LM, $u_j \epsilon\ R(0,y)$ where y = (30, 30k, 81k) for runs 1, 2, 3 respectively.

b) For $j \notin A$, set LM, $u_j = 0$ and generate $s_j \epsilon\ R(0,1)$.

Step 5) Feasibility: Compute $b = s - c^T x^*$

Step 6) Optimality: Compute $a = C\ u - G\ x^*$●

Appendix C

Van de Panne - Whinston's Dual Tableau Method

Definitions

Primal member, s_i:	slack or primal variable corresponding to constraint i.
Dual member, u_i:	LM or dual variable corresponding to constraint i.
Basic pair:	(s_i, u_i) are both basic.
Non-basic pair:	(s_i, u_i) are both non-basic.
Standard Tableau:	A tableau with no basic pair.
Non-Standard tableau:	A tableau that is not standard.

Dual Tableau Method

Repeat rules 1 and 2 to select the pivot and

a) terminate with optimal solution if rule 1 cannot be applied.

b) terminate with no feasible solution if rule 2 cannot be applied.

Rule 1 (Pick an entering variable)

a) *Standard Tableau*

Dual variable corresponding to the most negative primal variable.

b) *Non-Standard Tableau*

Primal member of the non-basic pair.

Rule 2 (choice of leaving variable)

Choose the basic variable to leave the basis according to the least nonnegative value of the ratio.

$$\underset{j \in \hat{J}}{min.} \left[d_j / w_j \right]$$

where d_j is the value of the basic variable,

w_j is the value corresponding to the entering variable column for that variable and $\hat{J}$ is the set as described below.

a) *Standard Tableau*

$\hat{J}$ is the set of basic dual variables and the basic primal variable corresponding to the entering dual variable.

b) *Non-Standard Tableau*

$\hat{J}$ is the set of basic dual variables excluding the dual variable of the basic pair, and the primal member of the basic pair●

Note: Start with a standard tableau which is dual feasible (i.e. the dual variables are non-negative).

Appendix D

Tables

Problem	NLP characteristics			QP characteristics					average cardinality of basis at QP Soln.
	var.	cons.	eqns.	var.	Fletcher QP			dual QP	
					cons.	low bd	hi bd	cons.	
Powell	5	3	3	6	3	6	6	5	4.0
POP	3	7	0	4	7	4	4	9	2.2
Triangle	7	9	0	8	9	8	8	11	6.1
Colville 1	5	15	0	6	15	6	6	17	6.0
Colville 2	15	20	0	16	20	16	16	22	14.0
Colville 3	5	16	0	6	16	6	6	18	6.0

Table 6.1. Problem characteristics.

Problem	Total of all iterations.													Ratio of Operations** to Dual		
	operations			sqrts.	steps			adds			drops					
	FP	FQ*	D	D	FP	FQ	D	FP	FQ	D	FP	FQ	D	FP	FQ	FP+FQ
Powell	2353	9237	4830	144	2	12	24	2	0	24	2	12	0	.38	1.47	1.85
POP	2228	3509	1384	49	15	11	13	15	0	13	15	11	0	1.19	1.87	3.06
Triangle	21394	70474	24236	440	33	40	83	33	6	81	33	31	2	.75	2.46	3.21
Colville 1	5896	12077	6060	119	14	7	32	14	0	30	14	4	4	.81	1.66	2.47
Colville 2	196142	665394	237983	2056	59	46	239	59	4	224	59	36	15	.76	2.57	3.33
Colville 3	4147	8012	4711	85	8	0	27	8	0	26	8	0	2	.75	1.44	2.19
Average														.77	1.91	2.69

Table 6.2. Numerical Results.

FP - feasible point routine for Fletcher's QP (phase 1 only)

FQ - Fletcher's QP algorithm (phase 2 only)

D - Dual projection algorithm

Note: See differences in implementation on page 43
Operations do not include square root calculations.

FP, FQ do not require square root calculations.

* includes effort for $\begin{pmatrix} G & N \\ N^T & 0 \end{pmatrix}^{-1}$ at the solution of each QPP, i.e. final reinversion

** includes square root calculation for the dual (10 operations/sqrt.)

Problem type	variables	constraints	constraints in final basis	conditon of G
1	9	9	1	well
2	9	9	1	ill
3	9	9	3	well
4	9	9	3	ill
5	9	27	3	well
6	9	27	3	ill
7	9	27	9	well
8	9	27	9	ill
9	27	27	3	well
10	27	27	3	ill
11	27	27	9	well
12	27	27	9	ill
13	27	81	9	well
14	27	81	9	ill
15	27	81	27	well
16	27	81	27	ill
17	81	81	9	well
18	81	81	9	ill
19	81	81	27	well
20	81	81	27	ill
21	81	243	27	well
22	81	243	27	ill
23	81	243	81	well
24	81	243	81	ill

Table 8.2 Problem Types

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1	D.P.1
1	19	5	37	35	12	11	10	10	5
2	21	5	41	41	15	16	12	12	5
3	24	15	22	22	11	11	9	8	15
4	24	17	33	33	17	17	16	17	17
5	47	15	50	45	22	23	13	13	15
6	44	17	74	79	21	20	18	18	17
7	55	60	44	33	39	37	38	37	64
8	50	54	45	35	46	42	47	41	54
9	73	15	103	101	30	31	16	16	15
10	67	15	95	97	31	34	20	21	15
11	67	45	75	75	30	30	28	28	45
12	74	45	114	106	44	37	40	36	45
13	147	47	186	165	43	43	27	28	47
14	191	47	197	211	50	50	46	48	47
15	172	141	155	108	107	121	115	117	141
16	211	144	140	115	128	109	127	122	144
Totals	1286	687	1411	1301	646	632	582	572	691

Table 8.3(a)* Number of steps run 1.

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1	D.P.1
1	19	5	32	30	24	24	24	24	5
2	21	5	36	36	30	30	28	28	5
3	24	15	17	17	15	15	15	15	15
4	24	17	29	29	29	31	29	31	17
5	52	15	61	53	45	43	43	41	15
6	49	17	72	78	56	54	50	50	17
7	65	61	86	64	78	74	76	74	65
8	56	55	83	65	89	81	91	81	55
9	73	15	98	96	86	82	76	76	15
10	67	15	90	92	94	94	78	78	15
11	67	45	70	70	64	64	68	68	45
12	74	45	109	101	103	91	99	91	45
13	164	47	215	193	145	139	127	129	47
14	247	47	224	246	178	174	166	172	47
15	209	141	292	212	214	242	230	234	141
16	287	144	202	168	256	218	254	244	144
Totals	1498	689	1716	1550	1506	1466	1454	1436	693

Table 8.3(b)* Number of basis changes, run 1.

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1	D.P.1
1	10702	5975	24768	24444	12369	11979	11990	11951	5490
2	11510	5975	27229	28239	14589	15549	13194	13257	5490
3	12365	11369	15276	15814	9835	10263	9338	8999	13075
4	12155	12173	23081	23805	16163	17028	16241	17275	15173
5	28414	14615	56518	54170	31176	33816	26529	26808	16315
6	27487	15595	74385	83368	31403	31567	29878	30260	19231
7	32700	36414	63647	52333	49423	49190	50816	50269	62935
8	29550	34363	63500	55132	57757	55861	62564	57132	51832
9	282262	114658	508384	529042	236616	243620	221812	224109	105235
10	266428	114665	482830	513580	251333	273137	227738	235871	105235
11	267196	237742	386032	400781	228353	234753	244528	247748	280460
12	283173	237749	576078	580236	344399	310554	395827	365435	280460
13	698267	317597	1537625	1558327	547619	609889	719114	738303	376857
14	932444	318186	1640739	2021097	659355	723717	1019411	1079672	377432
15	828776	714197	1684396	1420029	1086798	1277296	1359581	1379179	992003
16	1035779	722888	1291100	1252035	1290434	1176664	1574555	1586534	1031804

Table 8.3(c)* Number of operations, run 1

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1	D.P.1
1	1.79	1.00	4.15	4.09	2.07	2.00	2.01	2.00	0.92
2	1.93	1.00	4.56	4.73	2.44	2.60	2.20	2.22	0.92
3	1.09	1.00	1.34	1.39	0.87	0.90	0.82	0.79	1.15
4	1.00	1.00	1.90	1.96	1.33	1.40	1.33	1.42	1.25
5	1.94	1.00	3.87	3.71	2.13	2.31	1.82	1.83	1.12
6	1.76	1.00	4.77	5.35	2.01	2.02	1.92	1.94	1.23
7	0.90	1.00	1.75	1.44	1.36	1.35	1.40	1.38	1.73
8	0.86	1.00	1.85	1.60	1.68	1.63	1.82	1.66	1.51
9	2.46	1.00	4.43	4.61	2.06	2.12	1.93	1.95	0.92
10	2.32	1.00	4.21	4.48	2.19	2.38	1.99	2.06	0.92
11	1.12	1.00	1.62	1.69	0.96	0.99	1.03	1.04	1.18
12	1.19	1.00	2.42	2.44	1.45	1.31	1.66	1.54	1.18
13	2.20	1.00	4.84	4.91	1.72	1.92	2.26	2.32	1.19
14	2.93	1.00	5.16	6.35	2.07	2.27	3.20	3.39	1.19
15	1.16	1.00	2.36	1.99	1.52	1.79	1.90	1.93	1.39
16	1.43	1.00	1.79	1.73	1.79	1.63	2.18	2.19	1.43
Aver.	1.63	1.00	3.19	3.28	1.73	1.79	1.84	1.86	1.20

Table 8.3(d)* Ratio of number of operations to dual, run 1

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1	D.P.1
1	19	5	37	37	13	13	11	11	5
2	21	5	47	45	19	19	13	13	5
3	24	15	22	22	11	11	9	8	15
4	24	19	37	37	21	21	19	20	19
5	47	17	46	45	22	23	13	13	17
6	44	19	70	64	24	25	23	23	19
7	55	84	44	32	39	38	36	36	88
8	50	69	44	38	43	42	52	44	71
9	73	15	111	111	32	33	20	20	15
10	67	17	103	99	28	26	17	18	17
11	67	45	85	87	41	41	34	34	45
12	74	47	112	110	37	37	36	36	47
13	147	73	205	170	63	55	46	39	73
14	191	73	204	179	50	53	44	46	73
15	172	350	144	118	133	127	142	146	380
16	211	191	134	92	102	104	117	122	195
Totals	1286	1094	1445	1286	678	668	634	631	1084

Table 8.4(a)* Number of steps, run 2

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1	D.P.1
1	19	5	32	32	28	28	26	26	5
2	21	5	42	40	38	36	32	32	5
3	24	15	17	17	15	15	15	15	15
4	24	19	33	33	35	37	35	37	19
5	52	17	57	53	45	43	43	41	17
6	49	19	84	78	60	64	60	60	19
7	65	113	88	64	78	76	72	72	117
8	56	77	87	75	85	83	103	87	79
9	73	15	106	106	88	86	82	82	15
10	67	17	98	94	78	78	74	74	17
11	67	45	80	82	82	82	80	80	45
12	74	47	107	105	83	83	91	91	47
13	164	73	251	211	189	167	165	151	73
14	247	73	256	222	164	160	164	168	73
15	209	397	288	236	266	254	284	292	427
16	287	191	256	180	204	208	234	244	195
Totals	1498	1178	1882	1628	1538	1500	1560	1552	1218

Table 8.4(b)* Number of basis changes, run 2.

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1	D.P.1
1	10702	5975	24510	25526	13594	14047	12646	12964	5490
2	11510	5975	30483	30402	18270	18341	14573	14805	5490
3	12365	11369	15274	15742	10079	10420	9587	9156	13075
4	12155	12923	25215	26055	19431	20735	19231	20540	16478
5	28414	15227	52793	53805	31435	34051	26453	27032	19174
6	27487	16483	77016	74989	35385	38551	37124	37769	22080
7	32700	54823	64544	52524	50062	50452	48658	49611	84104
8	29550	42137	64600	60359	54194	56896	69842	60894	69728
9	282262	114658	544452	579394	251260	261776	245874	249555	105235
10	266428	120922	516523	519380	218950	219228	206170	215599	122147
11	267196	237742	429467	456927	301953	308302	282255	284612	280460
12	283173	243963	555094	585850	295400	305927	331246	337180	297334
13	698267	403156	1747620	1633473	763232	715243	784594	722874	559393
14	932444	413383	1761174	1754503	647635	730399	894491	942016	644029
15	828776	1442354	1610623	1513305	1323407	1332172	1572056	1599277	3147916
16	1035779	874863	1455188	1195073	1028918	1114449	1365317	1449225	1563452

Table 8.4(c)* Number of operations, run 2.

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1	D.P.1
1	1.79	1.00	4.10	4.27	2.28	2.35	2.12	2.17	0.92
2	1.93	1.00	5.10	5.09	3.06	3.07	2.44	2.48	0.92
3	1.09	1.00	1.34	1.38	0.89	0.92	0.84	0.81	1.15
4	0.94	1.00	1.95	2.02	1.50	1.60	1.49	1.59	1.28
5	1.83	1.00	3.40	3.47	2.02	2.19	1.70	1.74	1.23
6	1.67	1.00	4.67	4.55	2.15	2.34	2.25	2.29	1.34
7	0.60	1.00	1.18	0.96	0.91	0.92	0.89	0.90	1.53
8	0.70	1.00	1.53	1.43	1.29	1.35	1.66	1.45	1.65
9	2.46	1.00	4.75	5.05	2.19	2.28	2.14	2.18	0.92
10	2.20	1.00	4.27	4.30	1.81	1.81	1.71	1.78	1.01
11	1.12	1.00	1.81	1.92	1.27	1.30	1.19	1.20	1.18
12	1.16	1.00	2.28	2.40	1.21	1.25	1.36	1.38	1.22
13	1.73	1.00	4.33	4.05	1.89	1.77	1.95	1.79	1.39
14	2.26	1.00	4.26	4.24	1.57	1.77	2.16	2.28	1.56
15	0.57	1.00	1.12	1.05	0.92	0.92	1.09	1.11	2.18
16	1.18	1.00	1.66	1.37	1.18	1.27	1.56	1.66	1.79
Aver.	1.45	1.00	2.98	2.97	1.63	1.70	1.66	1.68	1.33

Table 8.4(d)* Ratio of number of operations to dual, run 2.

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1
17	44	9	72	70	13	13	8	7
18	38	9	62	60	13	12	10	10
19	32	29	68	64	40	42	42	42
20	38	29	72	72	26	33	32	31
21	105	103	189	161	60	58	75	68
22	93	35	155	128	31	36	48	35
23	107	252	135	91	129	119	128	131
24	118	113	133	90	105	111	120	111
Totals	575	579	886	736	417	424	463	435

Table 8.5(a)* Number of steps, run 3.

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1
17	44	9	71	69	53	49	49	47
18	38	9	61	59	49	49	47	47
19	32	29	67	63	83	85	87	87
20	38	29	71	71	61	75	73	71
21	129	103	278	222	168	166	202	188
22	105	35	202	164	114	124	148	122
23	133	267	270	182	258	238	256	262
24	155	113	264	180	210	222	240	222
Totals	674	594	1284	1010	996	1008	1102	1046

Table 8.5(b)* Number of basis changes, run 3

* See notes on page 79

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1
17	1436003	550164	2991116	3230184	944420	959405	1204284	1065050
18	1295627	550163	2645193	2753002	922019	947848	1323632	1350157
19	1145963	1225579	2900305	2980464	2500936	2618295	2795222	2827916
20	1295627	1227923	3048899	3466566	1717564	2188260	2379792	2347479
21	4289510	3638952	14982501	16152724	5984002	6367082	10624157	10538954
22	3801085	1799584	11593327	12852311	3545462	4558379	8783870	8276938
23	4447039	8016384	12897024	12534633	11008810	10722045	14526959	14366066
24	4912031	4421230	12800635	12638596	9015265	9937241	13307823	12832151

Table 8.5(c)* Number of operations, run 3.

Problem type	F	D	P.1	P.1.1	P.2	P.2.1	P.2'	P.2'.1
17	2.61	1.00	5.44	5.87	1.72	1.74	2.19	1.94
18	2.36	1.00	4.81	5.00	1.68	1.72	2.41	2.45
19	0.94	1.00	2.37	2.43	2.04	2.14	2.28	2.31
20	1.06	1.00	2.48	2.82	1.40	1.78	1.94	1.91
21	1.18	1.00	4.12	4.44	1.64	1.75	2.92	2.90
22	2.11	1.00	6.44	7.14	1.97	2.53	4.88	4.60
23	0.55	1.00	1.61	1.56	1.37	1.34	1.81	1.79
24	1.11	1.00	2.90	2.86	2.04	2.25	3.01	2.90
Ave.	1.48	1.00	3.76	4.01	1.73	1.90	2.67	2.59

Table 8.5(d)* Ratio of Number of operations to dual, run 3.

* See notes on page 79

Notes on Tables 8.3 through 8.5

The following remarks apply to the tables 8.3, 8.4 and 8.5.

The numbers under six primal method columns assume that the feasible point is readily available. Thus for comparison purposes with column D, one should add the appropriate columns labelled by F.

The number of operations include square roots (10 operations).

In tables 8.3(a,b,c), 8.4(a,b,c) the entries for given algorithm and given problem type are summed over the five replicates for that run. In tables 8.3(d), 8.4(d) the averages are column averages.

N_d= number of drops	number of problems with N_d drops		
	run 1	run 2	run 3
0	62	46	2
1	12	8	2
2	5	3	0
3 and up	1	23	4
total	80	80	8
Total drops	25	244	153

Table 8.6 Frequency of drops for the dual.

Problem Type	run 1.LM = (0,30) repetition					run 2. LM = (0,30k) repetition					run 3. LM = (0,81k)
	1	2	3	4	5	1	2	3	4	5	
1	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	1	0	0	0	0	2	
5	0	0	0	0	0	0	1	0	0	0	
6	0	0	0	0	1	0	0	1	0	1	
7	3	2	1	1	1	12	7	5	6	4	
8	1	2	1	0	1	4	3	3	2	4	
9	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	1	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	1	0	0	0	
13	0	0	0	1	0	0	0	7	6	1	
14	0	0	1	0	0	1	5	5	2	1	
15	0	1	0	2	0	24	27	20	35	25	
16	1	0	2	2	0	4	7	5	3	9	
17											0
18											0
19											1
20											1
21											38
22											4
23											93
24											16

Table 8.7. Number of drops in dual

Problem Type	initial basis. repetition					final basis.	initial ∩ final basis repetition				
	1	2	3	4	5		1	2	3	4	5
1	5	2	3	3	6	1	0	1	0	0	0
2	3	6	3	4	5	1	0	1	0	0	0
3	6	7	4	5	2	3	3	3	3	1	2
4	4	9	3	3	5	3	1	3	0	1	2
5	9	7	8	9	9	3	2	1	2	1	2
6	8	8	6	7	9	3	2	2	0	1	1
7	9	9	9	9	9	9	4	6	2	6	2
8	9	9	8	9	9	9	2	3	4	6	2
9	16	17	16	9	15	3	1	1	3	1	3
10	13	9	13	17	15	3	1	1	2	3	1
11	14	14	16	10	13	9	7	5	6	5	4
12	11	11	13	20	19	9	3	6	3	6	6
13	24	27	27	27	25	9	5	6	7	4	8
14	27	27	27	27	27	9	6	1	5	9	7
15	27	27	27	27	27	27	15	18	14	13	12
16	27	27	27	27	27	27	17	11	13	19	15
17	44					9	6				
18	38					9	4				
19	32					27	10				
20	38					27	11				
21	81					27	14				
22	81					27	13				
23	81					81	40				
24	81					81	38				

Table 8.8 Intersection of initial and final bases for primal methods.

Method	Run 1	Run 2	Run 3
F	1.16	1.16	1.17
D	1.00	1.08	1.03
D.P.1	1.00	1.12	not run
P.1	1.21	1.30	1.45
P.1.1	1.19	1.27	1.37
P.2	2.33	2.27	2.39
P.2.1	2.32	2.25	2.38
P.2'	2.50	2.46	2.38
P.2'.1	2.51	2.46	2.40

Table 8.9. Ratio of basis changes to step

Method	Run 2 - Run 1 steps	Run 2 - Run 1 basis changes
D	407	489
D.P.1	393	525
P.1	34	166
P.1.1	-15	78
P.2	32	32
P.2.1	36	34
P.2'	52	104
P.2'.1	59	116

Table 8.10. Effect of linearity of function on algorithms.

Appendix E

Program Listing

Following are the names and functions of various routines used in the experimental run.

NAME		*FUNCTION*
MAIN	-	It is the main program, sets up I/O and summary.
GENER	-	Generate a random problem with known solution.
DECOMP	-	Decompose G into LL^T
INVL	-	Invert L
SOLVL	-	Solve linear system $Ly = w$
SOLVLT	-	Solve linear system $L^T y = w$
SOLVR	-	Solve linear system $Ry = w$
SOLVRT	-	Solve linear system $R^T y = w$
UNCONM	-	Find unconstrained minimum
RAND	-	Generate random number
GRA	-	Calculate gradient
PRTMAT	-	Print matrix
PROJ	-	Project vector onto the current manifold
RBAR	-	Calculate $Q^T L^{-1} y$
MXMIN	-	Find max. or min. of a vector
ADD	-	Add a constraint to the basis
GIVENM	-	Calculate coefficients of Givens Matrix
DROP	-	Remove a constraint from basis
INTLQL	-	Set $Q^T L^{-1} = L^{-1}$
GVNUPV	-	Multiply a vector of two elements with Givens Matrix
LAMDA	-	Calculate Slack vector

XCOR	-	Perform X-correction
REIN	-	Perform reinversion
FEAS	-	Feasible point routine
STPLEN	-	Calculate step length for primal methods
FLET	-	Primal algorithm P.1, P.1.1
NSGNST	-	Calculate a diagonal elemet of $R^{-1}R^{-T}$
STPEDG	-	Calculate steepest edge
DUAL	-	Dual Projection method D
DSTPLN	-	Dual step length
UPRIN	-	Update logging print array
PRIMAL	-	Primal routine initializer
ACTCHK	-	Check active constraints on manifold
RENCHK	-	Check if reinversion necessary
PRDUAL	-	Primal-dual initializer
GOLD	-	Primal algorithm P.2, P.2.1, P.2', P.2'.1
CHECK	-	Checks with known solution
SINNER	-	Calculate inner product of two vectors (Assembly routine)

```
FORTRAN 1V G LEVEL  21                  MAIN                    DATE = 80105

            C     THIS IS THE MAIN PROGRAM
0001                  IMPLICIT REAL*8(A-H,O-Z)
            COMMON C1
0002               COMMON /C01/PROJV,LENY,QTLZ
0003               REAL * 8 PROJV(30),LENY,QTLZ(30)
            COMMON C1 END
            COMMON 02
0004               COMMON /C02/LG
0005               REAL*8 LG(30,31),L(30,30),G(30,30)
0006               EQUIVALENCE(LG(1,1),L(1,1))
0007               EQUIVALENCE(LG(1,2),G(1,1))
            COMMON 02 END
            COMMON 03
0008               COMMON /C03/QTLINV
0009               REAL*8  QTLINV(30,30)
            COMMON 03 END
            COMMON 04
0010               COMMON /C04/Q1R
0011               REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0012               EQUIVALENCE(Q1R(1,1),Q1(1,1))
0013               EQUIVALENCE(Q1R(1,3),R(1,1))
            COMMON 04 END
            COMMON 05
0014               COMMON /C05/C,B,XLAMDA,ICONOK
0015               REAL*8 C(30,90),B(90),XLAMDA(90)
0016               INTEGER*2 ICONOK(90)
            COMMON 05 END
            COMMON 06
0017               COMMON /C06/FF,FNQ,FQ
0018               INTEGER FNQ(30),FQ
0019               REAL*8 FF
            COMMON C6 END
            COMMON 07
0020                COMMON /C07/   FQTLIN
0021                REAL*8 FQTLIN(30,30)
            COMMON C7 END
            COMMON 08
0022                COMMON /C08/   FXLINR
0023                REAL*8 FXLINR(30,32),FX(30),LINV(30,30),FR(30,30)
0024                EQUIVALENCE(FXLINR(1,1),FX(1))
0025                EQUIVALENCE(FXLINR(1,2),LINV(1,1))
0026                EQUIVALENCE(FXLINR(1,3),FR(1,1))
            COMMON C8 END
            COMMON C9
0027               COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  1,    ISEED,ISOPT,IAOPT,IDOPT
0028                INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
            COMMON C9 END
            COMMON 10
0029                 COMMON /C10/UNCONX,UNCONF
0030                 REAL*8 UNCONX(30),UNCONF
            COMMON 10 END
            COMMON 11
0031                 COMMON /C11/X
0032                 REAL*8 X(30)
            COMMON 11 END
```

```
FORTRAN IV G LEVEL  21                  MAIN                    DATE = 80105

            COMMON 12
0033                 COMMON /C12/NQ
0034                 INTEGER NQ(30)
            COMMON 12 END
            COMMON 13
0035                 COMMON /C13/LM,IM
0036                 REAL*8 LM(30),IM(30)
            COMMON 13 END
            COMMON 14
0037                   COMMON /C14/F,GRAD
0038                   REAL*8 F,GRAD(30)
            COMMON 14 END
            COMMON 15
0039                   COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
            COMMON 15 END
            COMMON 16
0040              COMMON/C16/HIX,LOX,HIXLOX,HIG,LOG,HIGLOG,HILM,HIC,LOC,HICLOC,HI8
0041              REAL*8 HIX,LOX,HIXLOX,HIG,LOG,HIGLOG,HILM,HIC,LOC,HICLOC,HI8
            COMMON 16 END
            COMMON 17
0042              COMMON/C17/IUNIT5,IUNIT6,IUNIT8
            COMMON 17 END
            COMMON 18
0043                 COMMON /C18/SOLNX,SOLNF,SOLNQ
0044                 REAL*8 SOLNX(30),SOLNF
0045                 INTEGER SOLNQ
            COMMON 18 END
            COMMON 19
0046                 COMMON /C19/IPRIN,IPTR,IPMAX,NBRK
0047                 INTEGER*2 IPRIN(5000),IPTR,IPMAX,NBRK(15)
            COMMON 19 END
            COMMON 20
0048                  COMMON /C20/A
0049                  REAL*8 A(30)
            COMMON 20 END
            COMMON 21
0050              COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NRE IN,NXCOR
                 1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
            COMMON 21 END
            COMMON 22
0051                 COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0052                 REAL*8 TITLE(15)
0053                 INTEGER PRTOPT
            COMMON 22 END
            COMMON 23
0054                 COMMON/C23/GC,GH,GP,GB
0055                 REAL * 8 GC(30),GH(30),GP(30),GB(30)
            COMMON 23 END
0056                 INTEGER IF11(12)
0057                 EQUIVALENCE(IF11( 1),IAG    )
0058                 EQUIVALENCE(IF11( 2),NSTEPS)
0059                 EQUIVALENCE(IF11( 3),NADDS )
0060                 EQUIVALENCE(IF11( 4),NDROPS)
0061                 EQUIVALENCE(IF11( 5),NOPS   )
0062                 EQUIVALENCE(IF11( 6),NSQRT )
0063                 EQUIVALENCE(IF11( 7),NREIN )
```

```
FORTRAN IV G LEVEL  21                  MAIN                DATE = 80105

0064                EQUIVALENCE(IF11( 8),NXCOR )
0065                EQUIVALENCE(IF11( 9),NFSTEP)
0066                EQUIVALENCE(IF11(10),NPSTEP)
0067                EQUIVALENCE(IF11(11),ICC   )
0068                    EQUIVALENCE(IF11(12),NTIME)
0069                INTEGER * 2 IJPRIN(14)
0070                REAL * 8 BLANK,TYPRIN(12),APRIN,TYPALG(14),APROB
0071                  DATA APROB/'PROB DEF'/
0072                    REAL * 8 DAT
0073                DATA BLANK,TYPRIN/'        ','ALG. #  ','# STEPS ',
                   1'# ADDS  ','# DROPS ','#MLT/100','# SQRT  ','# REINV ',
                   1'# XCOR  ','# F STPS','# P STPS','COMPCODE','TIM*10|3'/
0074                DATA TYPALG/'FEAS PT ','DUAL ','FLET- 1 ','FLET-STP','GOLD- 1 ',
                   1'GOLD-STP','G-1+VIOL','GSTP+VIO','PD-F-1  ','PD-F-STP',
                   1'PD-G-1  ','PD-G-STP','PDG1VIOL','PDGSTPVI'/
0075                    INTEGER ITOTS(16,14,5)
0076                    DATA ITOTS / 1120 * 0/
0077                   REAL*8 TYPTOT(6)
0078                   REAL*8 FTOTS(17,14)
0079                   EQUIVALENCE(FTOTS,IPRIN)
0080                DATA TYPTOT/'#STEPS ','#BAS.CHG','#OPS.','TIME   ','#PROBS  ',
                   1'RATIO D '/
                C       THIS IS THE END OF DECLARATIONS
                C
                C
                C
                CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
                C
                C       DECLARATIONS FOR PROBLEM RUN ETC.
0081                    INTEGER * 2 IPHRUN(80),NPROB,IAGRUN(14),PRDEF(16,5),ICPT(14,3)
0082                    INTEGER * 4 PSEED(80)
                C     NPROB IS NUMBER OF PROBLEMS
                C     IPHPUN IS PROBLEM NUMBERS TO BE RUN (NPROB IS ITS INDEX)
                C     IAGRUN IS ALGORITHMS TO BE RUN
                C       -1  - FEASIBLE PT. ROUTINE
                C       -2  - DUAL ROUTINE
                C       -3  - PRIMAL FLETCHER
                C       -4  - PRIMAL FLETCHER STEEPEST EDGE
                C       -5  - PRIMAL GOLDFARB
                C       -6  - PRIMAL GOLDFARB STEEPEST EDGE
                C       -7  - PRIMAL GOLDFARB WITHOUT VIOLATING ALREADY DROPPED B4 STEP
                C       -8  - PRIMAL GOLDFARB (COMBO OF 6 & 7)
                C       -9  - DUAL PRIMAL(283)
                C       -10 -  "    "    (284)
                C       -11 -  "    "    (285)
                C       -12 -  "    "    (286)
                C       -13    "    "    (287)
                C       -14    "    "    (288)
                C
                C      PRDEF IS PROBLEM DEFINITION OF ALL 16 PROBLEMS (ROWS)
                C          COL. 1 IS  M - NUMBER OF VARIABLES
                C               2 IS  K - NUMBER OF CONSTRAINTS
                C               3 IS  TYPG - TYPE OF G (IN OBJ. FN.  1 - WELL, 2 - ILL)
                C             4 IS  SGLNQ - NUMBER OF CONSTRAINTS IN FINAL BASIS(FIRST)
                C               5 IS IMAGE (IN BINARY REP)
                C
```

```
FORTRAN IV G LEVEL  21                  MAIN                DATE = 80105

                C
                C     NOW THE MAIN PROGRAM STARTS
                C       READ THE TITLE OF RUN
0083                     ISIZ = 30
0084                     IUNIT5 = 5
0085                     IUNIT6 = 6
0086                     IUNIT8 = 8
0087                     IPMAX  = 5000
0088                     ON = 1
0089                     OFF = 0
0090                        CALL DATE(DAT)
0091                       WRITE(IUNIT6,4011)DAT
0092           4011        FORMAT('1',//,5X,'RUN DATE = ',A8)
0093                       WRITE(IUNIT8,4012)DAT
0094            4012       FORMAT(5X,'RUN DATE = ',A8)
                C
0095                     WRITE(IUNIT6,18)
0096            18       FORMAT(/////////)
0097            17       READ(IUNIT5,19)IOMIT,(TITLE(J),J=1,20)
0098            19       FORMAT(I1,19A4,A3)
0099                     WRITE(IUNIT6,16)IOMIT,(TITLE(J),J=1,20)
0100                       WRITE(IUNIT8,19)IOMIT,(TITLE(J),J=1,20)
0101            16       FORMAT(' ',5X,I1,19A4,A3)
0102                     IF(IOMIT.EQ.0)GOTO 17
0103                     NLINE = 0
0104                     NPAGE = 1
0105                     READ(IUNIT5,50)TITLE
0106           50        FORMAT(10A8/5A8)
                C
                C    READ THE NUMBERS OF THE PROBLEMS TO BE RUN
                C            NRPT IS THE NO. OF REPETITION OF A PROBLEM (5)
                C     NOALG IS NUMBER OF DIFFERENT ALGS = 14
                C     NOPROB IS NUMBER OF DIFFERENT TYPES OF PROBLEMS (16)
                C              NTPROB IS TOTAL NO OF PROBLEMS
0107                      NOALG = 14
0108                      NRPT = 5
0109                     WRITE(IUNIT6,31) TITLE,NPAGE
0110                       WRITE(IUNIT8,51)TITLE
0111            51         FORMAT(15A8)
0112            31       FORMAT('1',1X,15A8,  'PAGE=',I4,//,5X,'   PROBLEMS TO BE RUN
0113                      NOPROB = 16
0114                     NPROB = 0
0115                     DO 60 I=1,1000
0116                       READ(IUNIT5,65)IOMIT,(IPRIN(J),J=1,15)
0117           65         FORMAT(16(I2,1X))
0118                      IF(IOMIT.LT.0) GOTO 60
0119                     WRITE(IUNIT6,62)(IPRIN(J),J=1,15)
0120            62       FORMAT(' ',15I4)
0121                      DO 70 J=1,15
0122                        IF(IPRIN(J))90,70,68
0123           68             NPROB = NPROB + 1
0124                          IPHRUN(NPROB) = IPRIN(J)
0125           70         CONTINUE
0126           60        CONTINUE
                C
                C      READ THE ALGORITHMS TO BE RUN
```

```
FORTRAN IV G LEVEL  21                MAIN                DATE = 80105

C
0127            WRITE(IUNIT6,81)
0128       81   FORMAT(' ' ,5X,'ALGS TO BE RUNC START, ADD,DROP OPTIONS ARE ')
0129       90   DO 80 I = 1,NOALG
0130       75   FORMAT(4(I2,1X))
0131            READ(IUNIT5,75)IAGRUN(I),(IOPT(I,J),J=1,3)
0132       80     WRITE(IUNIT6,82)I,TYPALG(I),IAGRUN(I),(IOPT(I,J),J=1,3)
0133       82     FORMAT(' ',5X,I5,3X,A8,4I5)
C    IOPT IS THE VARIOUS OPTIONS FOR THE ALGORITHMS
C       COL. 1 IS FOR STARTING PT FOR FR EAS PT.
C                1 - UNCONSTRAINED MIN.
C                2 - RANDOM STARTING PT.
C       COL. 2 IS FOR ADDING CONSTRAINT(IF APPLICABLE)
C                1 - ADD MOST VIOLATED CONS.
C                2 - ADD LEAST VIOLATED CONS.
C                3 - ADD RANDOM VIOLATED CONS.
C       COL. 3 IS FOR DROPPING CONSTRAINT(IF APPLIC.)
C                1 - DROP MOST -VE LM OR IM.
C                 2 - DROP LEAST -VE LM OR IM.
C                 3 - DROP RANDOM -VE LM OR IM(INFEASIBILITY MULT.)
C
C
C     READ THE TOLERANCES
0134             READ(IUNIT5,85)TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
0135             WRITE(IUNIT6,83)TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
0136       83    FORMAT(' ',5X,'TOLERANCES: DCM,LDEP,INV,PROJ,LAMDA,MULT',/
                1,5X,6F10.7)
0137       85    FORMAT(6F10.7)
C
C    TOLDEP - LINEAR DEPENDENCE TOLERANCE
C    TOLPR  - PROJECTION TOLERANCE
C    TOLAM  - LAMDA(VIOLATION) TOLERANCE
C    TOLMUL - MULTIPLIER( LM & IM) TOLERANCE
C
0138               READ(IUNIT5,95)MXSTPS,MXREIN
0139              WRITE(IUNIT6,91)MXSTPS,MXREIN
0140       91     FORMAT(' ',5X,'MAX STEP,REIN ',2I5)
0141       95     FORMAT(2(I4,1X))
C      MXSTPS IS MAX NUMBER OF STEPS IN AN ALGORITHM
C      MXREIN IS MAX NUMBER OF REINVERSIONS IN AN ALGORITHM
C
0142               READ(IUNIT5,75)PRTOPT
0143              WRITE(IUNIT6,92)PRTOPT
0144       92     FORMAT(' ',5X,'PRINT OPTION :0 - LESS,1 - MORE ',I5)
C
C        PRTOPT IS PRINT OPTION
C     0 - LEAST PRINTING
C     1 - MORE
C
C     NOW READ THE VARIOUS PROBLEM DEFINTIONS
0145            WRITE(IUNIT6,103)
0146       103  FORMAT(' ',5X,'PROB. DEF.: M,K,G(1-WELL,2-ILL),FB,BINARY')
0147            DO 110 I= 1,NOPROB
0148              READ(IUNIT5,105)(PRDEF(I,J),J=1,5)
0149             WRITE(IUNIT6,106)I,(PRDEF(I,J),J=1,5)
0150       106   FORMAT(' ',5X,616)
```

```
FORTRAN IV G LEVEL  21                  MAIN                DATE = 80105

0151         105       FORMAT(5I5)
0152         110      CONTINUE
0153                   NTPROB = NDPROB * NRPT
0154                   READ(IUNIT5,111)HIC,LOC,HIG,LOG,HIX,LOX,HIB,HILM
0155          111      FORMAT(4F10.5/4F10.5)
0156                   HICLOC = HIC - LOC
0157                   HIGLOG = HIG - LOG
0158                   HIXLOX = HIX - LOX
0159                   WRITE(IUNIT6,112)HIC,LOC,HIG,LOG,HIX,LOX,HIB,HILM
0160          112      FORMAT(' ',2X,'HIC,LOC',2F10.5,5X,'HIG,LOG',2F10.5,
                     1  5X,'HIX,LOX',2F10.5,5X,'HIB,HILM',2F10.5)
0161                   DO 120 I=1,NTPROB
0162         115       FORMAT(I10)
0163                   READ(IUNIT5,115)PSEED(I)
0164         120       CONTINUE
                     C
                     C
                     C       RUN ALL THE PROBLEMS
                     C
                     C
0165                   WRITE(IUNIT8,1017)TYPRIN
0166          1017     FORMAT(/'PROB #',5X,12(A8,1X))
0167                      DO 1000 I = 1,NPROB
0168                      IPROB = IPMRUN(I)
0169                      IPRTYP = IPROB- IPROB    /NDPROB *NDPROB
0170                      IF(IPRTYP.EQ.0)IPRTYP = NDPROB
0171                      M = PRDEF(IPRTYP,1)
0172                      K = PRDEF(IPRTYP,2)
0173                      TYPG = PRDEF(IPRTYP,3)
0174                    SOLNQ = PRDEF(IPRTYP,4)
0175                      KINDPR = PRDEF(IPRTYP,5)
0176                      ISEED = PSEED(IPROB)
0177                    WRITE(IUNIT6,125)TITLE
0178          125       FORMAT('1',/,2X,15A8)
0179                    WRITE(IUNIT6,127)IPROB,IPRTYP,M,K,TYPG,SOLNQ,KINDPR
0180          127       FORMAT(' ',/5X,' PROB. NO.',I5,' PROB. TYPE',I5,
                     1' M= ',I5,' K= ',I5,'  G(1 -WELL,2-ILL)',I5,
                     2'  FINAL BASIS(FIRST) ',I5,' CONS. ',' KIND OF PROB',I5)
                     C
                     C      NOW CALL GENER TO GENERATE THE PROBLEM
                     C
                     C
0181                      NOPS = 0
0182                      NSQRT = 0
0183                      CALL GENER (IRC,III,ITIMDI)
0184                      NGOPS = NOPS
0185                      NGSQRT = NSQRT
0186                      IF(IRC.EQ.0) GOTO 140
0187                    WRITE(IUNIT6,129)IRC
0188         129        FORMAT(' ','***** GENERATION OF PROBLEM FAILED - RC =',I5)
0189                    GOTO 1000
                     C
                     C
                     C      NOW FIND THE UNCONSTRAINED MINIMUM FOR THE PROBLEM
                     C
                     C
```

```
                C
0190             140      NOPS = 0
0191                      NSQRT = 0
0192                        CALL CLOCK(0)
0193                      CALL UNCONM
0194                        CALL CLOCK(1,TIME)
0195                        ITIMU = TIME * 100000
0196                      NUOPS = NOPS
0197                      NUSQRT = NSQRT
                C
                C
                C
                C     SET FEASOKFLAG TO BE OFF
0198                      FSOKFG = OFF
                C
0199                          IPTR = 0
0200                        DO 900 IJK = 1,NDALG
0201                       NBRK(IJK) = IPTR + 1
0202                       DO 250 IJKL=1,12
0203                       IPTR = IPTR + 1
0204                      IF11(IJKL) = 0
0205             250       IPRIN(IPTR) = 0
0206                     IAG = IJK
0207                        IF(IAGRUN(IAG).LE.0)GOTO 900
                C
                C       SET THE COUNTERS TO ZERO
                C       COPY THE OPTIONS FOR ALG. TO BE RUN
0208                        ISOPT = IOPT(IAG,1)
0209                      IAOPT = IOPT(IAG,2)
0210                      IDOPT = IOPT(IAG,3)
0211                     GOTO(141,142,143),IAOPT
0212             141     IAOPT = 1
0213                     GOTO 144
0214             142     IAOPT = -1
0215                     GOTO 144
0216             143     IAOPT = 0
0217             144     GOTO(145,146,147),IDOPT
0218             145     IDOPT = 1
0219                     GOTO 148
0220             146     IDOPT = -1
0221                     GOTO 148
0222             147     IDOPT = 0
0223             148     CONTINUE
0224                      WRITE(IUNIT6,153)TYPALG(IAG)
0225             153      FORMAT(' ',3X,A8)
                C
                C     NOW WE HAVE COPIED THE OPTIONS IN EFFECT
                C
0226                       CALL CLOCK(0)
0227                       GOTO(1,2,3,3,3,3,3,3,9,9,9,9,9,9),IAG
0228             1         CALL FEAS(IRC,III)
0229                       CALL CLOCK(1,TIME)
0230                       NTIME = TIME * 100000
0231                      NFOPS  = NOPS
0232                      NFSQRT = NSQRT
0233                       NOPS=NOPS+NGOPS
```

```
FORTRAN IV G LEVEL  21                MAIN                DATE = 80105

0234                    NSQRT=NSQRT+NGSQRT
0235                   INT = 0
0236                   IF(Q.EQ.0.OR.SOLNQ.EQ.0)GOTO 230
0237                   DO 210 II = 1,Q
0238                   IF(NQ(II).LE.SOLNQ)INT=INT+1
0239            210    CONTINUE
0240            230    WRITE(IUNIT8,220)IPROB,IPRTYP,KINDPR,M,K,SOLNQ,TYPQ,INT
                   1,ITIMDI,NGOPS,NGSQRT,ITIMU,NUOPS,NUSQRT
                   1,APROB
0241            220    FORMAT(I2,3X,I2,3X,I4,'  M=',I3,'  K=',I3,' F8=',
                   1I3,'(FIRST)',' Q-',I3,' INT. ',I3,6I7,A8)
0242                   ICC = INT
0243                    IF (IRC.EQ.0)FSOKFG = ON
0244                    GOTO 850
0245            2       CALL DUAL(IRC)
0246                     CALL CLOCK(1,TIME)
0247                   NOPS=NOPS+NGOPS+NUOPS
0248                   NSQRT=NSQRT+NGSQRT+NUSQRT
0249                    NTIME = TIME * 100000
0250                   GOTO 800
0251            3       IPAG = IAG - 2
0252                   IF(FSOKFG.EQ.ON)GOTO 7
0253                   WRITE(IUNIT6,6)IAG
0254            6     FORMAT(' ',2X,'SINCE FEAS PT IS NOT FOUND - ALG. NOT RUN',I5)
0255                  GOTO 900
0256            7      CONTINUE
0257                    CALL PRIMAL(IPAG,IRC,0)
0258                     CALL CLOCK(1,TIME)
0259                     NTIME = TIME * 100000
0260                   GOTO 800
0261            9       IPDAG = IAG - 8
0262                   CALL PRDUAL(IPDAG,IRC,1)
0263                      CALL CLOCK(1,TIME)
0264                     NTIME = TIME * 100000
0265                   GOTO 800
0266            800    CONTINUE
0267                   CALL CHECK
0268            850     CONTINUE
0269                   WRITE(IUNIT8,852)IPROB,IF11,TYPALG(IAG)
0270            852    FORMAT(I2,1X,I2,318,I12,719,8X,A8)
0271                     ITOTS(IPRTYP,IAG,1) = ITOTS(IPRTYP,IAG,1) + NSTEPS
0272                     ITOTS(IPRTYP,IAG,2) = ITOTS(IPRTYP,IAG,2) + NADDS + NDROPS
0273                     ITOTS(IPRTYP,IAG,3) = ITOTS(IPRTYP,IAG,3) + NOPS + 10*NSQRT
0274                     ITOTS(IPRTYP,IAG,4) = ITOTS(IPRTYP,IAG,4) + NTIME
0275                     ITOTS(IPRTYP,IAG,5) = ITOTS(IPRTYP,IAG,5) + 1
0276                   WRITE(IUNIT6,5371)TYPRIN
0277            5371   FORMAT(' ',12(A8,1X))
0278                     WRITE(IUNIT6,853)IF11
0279            853      FORMAT(' ',5X,I2,318,I12,719)
0280                   IF11(5) = (IF11(5)+50)/100
0281                    NTIME = (NTIME + 50)/100
0282                   DO 860 II = 1,12
0283            860    IPRIN(NBRK(IAG)+II-1) = IF11(II)
0284           900      CONTINUE
0285                   NMAX = 0
0286                   WRITE(IUNIT6,903)TYPALG
```

```
FORTRAN IV G LEVEL  21                  MAIN                  DATE = 80105

0287          903      FORMAT(' ',6X,14(A8,1X)/)
0288                   NBRK(NDALG+1)=IPTR+1
0289                   DO 910 II = 1,NDALG
0290                   IF(NBRK(II+1)-NBRK(II).LE.NMAX)GOTO910
0291                   NMAX = NBRK(II+1) - NBRK(II)
0292          910      CONTINUE
0293                   DO 940 II = 1,NMAX
0294                   IM1 = II - 1
0295                   DO 930 JJ = 1,NDALG
0296                   NBB = NBRK(JJ) + IM1
0297                   IJPRIN(JJ) = 0
0298                   IF(NBB.LT.NBRK(JJ+1))IJPRIN(JJ) = IPRIN(NBB)
0299          930      CONTINUE
0300                   APRIN = BLANK
0301                   IF(II.LE.12)APRIN = TYPRIN(II)
0302                   WRITE(IUNIT6,931)APRIN,IJPRIN
0303          931      FORMAT(' ',A5,1X,14I9)
0304          940      CONTINUE
0305         1000       CONTINUE
0306                   NDPR1=NDPROB+1
0307                   DO 605 I=1,17
0308                   DO 605 J=1,14
0309          605      FTOTS(I,J)=0.DO
0310                  DO 610 I=1,NDALG
0311                  SUM=0.DO
0312                  DO 607 J=1,NDPROB
0313                  FTOTS(J,I)=ITOTS(J,I,3)/(0.DO+ITOTS(J,2,3))
0314                   SUM=SUM+FTOTS(J,I)
0315          607     CONTINUE
0316                  FTOTS(NDPR1,I)=SUM/NDPROB
0317          610     CONTINUE
0318                     DO 540 KK = 1,6
0319                     WRITE(IUNIT6,573)TYPTOT(KK),TYPALG
0320                     WRITE(IUNIT8,574)TYPTOT(KK),TYPALG
0321                     IF(KK.NE.6)GOTO 535
0322                  WRITE(IUNIT6,575)(II,(FTOTS(II,JJ),JJ=1,NDALG),II=1,NDPR1)
0323                  WRITE(IUNIT8,576)(II,(FTOTS(II,JJ),JJ=1,NDALG),II=1,NDPR1)
0324          575     FORMAT(17(1X,I2,14F9.4/))
0325          576     FORMAT(17(1X,I2,5X,14F9.4/))
0326                  GOTO 540
0327          535     CONTINUE
0328          573        FORMAT(' ',5X,A8/1X,'PRTP',14(1X,A8))
0329          574        FORMAT(A8/14(1X,A8))
0330                  WRITE(IUNIT6,571)(II,(ITOTS(II,JJ,KK),JJ=1,NDALG),II=1,NDPROB)
0331          571        FORMAT(16(1X,I2,14I9/))
0332                  WRITE(IUNIT8,572)(II,(ITOTS(II,JJ,KK),JJ=1,NDALG),II=1,NDPROB)
0333          572        FORMAT(16(1X,I2,3X,14I9/))
0334          540        CONTINUE
0335                     STOP
0336                     END
```

```
FORTRAN IV G LEVEL  21                  GENER                DATE = 80105

0001                 SUBROUTINE GENER (IRC,III,ITIMDI)
0002                   IMPLICIT REAL*8(A-H,O-Z)
            COMMON C2
0003                   COMMON /C02/LG
0004                   REAL*8 LG(30,31),L(30,30),G(30,30)
0005                   EQUIVALENCE(LG(1,1),L(1,1))
0006                   EQUIVALENCE(LG(1,2),G(1,1))
            COMMON 02 END
            COMMON C4
0007                   COMMON /C04/Q1R
0008                   REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0009                   EQUIVALENCE(Q1R(1,1),Q1(1,1))
0010                   EQUIVALENCE(Q1R(1,3),R(1,1))
            COMMON C4 END
            COMMON 05
0011                   COMMON /C05/C,B,XLAMDA,ICONOK
0012                   REAL*8 C(30,90),B(90),XLAMDA(90)
0013                   INTEGER*2 ICONOK(90)
            COMMON C5 END
            COMMON C8
0014                    COMMON /C08/   FXLINR
0015                    REAL*8 FXLINR(30,32),FX(30),LINV(30,30),FR(30,30)
0016                    EQUIVALENCE(FXLINR(1,1),FX(1))
0017                    EQUIVALENCE(FXLINR(1,2),LINV(1,1))
0018                    EQUIVALENCE(FXLINR(1,3),FR(1,1))
            COMMON C8 END
            COMMON 09
0019                 COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    1,    ISEED,ISOPT,IAOPT,IDOPT
0020                  INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
            COMMON C9 END
            COMMON 10
0021                   COMMON /C10/UNCONX,UNCONF
0022                   REAL*8 UNCONX(30),UNCONF
            COMMON 10 END
            COMMON 11
0023                   COMMON /C11/X
0024                   REAL*8 X(30)
            COMMON 11 END
            COMMON 13
0025                    COMMON /C13/LM,IM
0026                    REAL*8 LM(30),IM(30)
            COMMON 13 END
            COMMON 14
0027                      COMMON /C14/F,GRAD
0028                      REAL*8 F,GRAD(30)
            COMMON 14 END
            COMMON 16
0029                 COMMON/C16/HIX,LOX,HIXLOX,HIG,LOG,HIGLOG,HILM,HIC,LOC,HICLOC,HIB
0030                 REAL*8 HIX,LOX,HIXLOX,HIG,LOG,HIGLOG,HILM,HIC,LOC,HICLOC,HIB
            COMMON 16 END
            COMMON 17
0031                 COMMON/C17/IUNIT5,IUNIT6,IUNITd
            COMMON 17 END
            COMMON 18
0032                 COMMON /C18/SOLNX,SOLNF,SOLNQ
```

```
0033                  REAL*8 SOLNX(30),SOLNF
0034                  INTEGER SOLNQ
                COMMON 18 END
                COMMON 19
0035                  COMMON /C19/IPRIN,IPTR,IPMAX,NBRK
0036                  INTEGER*2 IPRIN(5000),IPTR,IPMAX,NBRK(15)
                COMMON 19 END
                COMMON 20
0037                   COMMON /C20/A
0038                   REAL*8 A(30)
                COMMON 20 END
                COMMON 22
0039                  COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0040                  REAL*8 TITLE(15)
0041                  INTEGER PRTOPT
                COMMON 22 END
0042                  REAL*8 TEMP(30),TEMP1(30)
                C
                C
                C       THIS SR WILL GENERATE A PROBLEM
                C
                C
                C     GENERATE G (UPPERTRIANGULAR)
                C
                C     CALCULATE RANGE OF NUMBERS FOR G
                C
0043                      IRC = 0
0044                        III = ISEED
                C
0045                       DO 10 I = 2,M
0046                         I1 =I-1
0047                       DO 10 J=1,I1
0048            10         G(J,I) = RAND(III)*HIGLOG+LOG
                C
                C    NOW FILL UP THE DIAGONALS OF G
                C
                C  DEPENDING UPON TYPE OF G (WELL OR ILL)
                C
                C
0049                      DO 70 I=1,M
0050                      SUM = 0.D0
0051                        DO 40 J= 1,M
0052                        IF(I-J)30,40,20
0053            20          SUM=SUM+DABS(G(J,I))
0054                        GOTO 40
0055            30          SUM=SUM+DABS(G(I,J))
0056            40          CONTINUE
0057                      G(I,I) = SUM + RAND(III)
0058                      IF(TYPE.EQ.1)GOTO 60
                C       IT IS ILL CONDITIONED G
                C
0059                      IF(I.EQ.1) GOTO 50
0060                      G(I,I)=G(I-1,I-1)+G(I,I)+SUM1
0061            50        SUM1 = SUM
0062                      GOTO 70
0063            60        G(I,I) = G(I,I)+1.D0
```

```
0064          70      CONTINUE
              C
              C
              C
              C
              C     NOW DECOMPOSE G
              C
              C
0065                  IF(PRTOPT.GT.2)
             1CALL PRTMAT(G,1,M,M,-1,M,ISIZ,ISIZ+1,1,1)
0066                  CALL CLOCK (0)
0067                    CALL DECOMP (IRC)
0068                  IF(PRTOPT.GT.2)
             1CALL PRTMAT(L,1,M,1, 1,M,ISIZ,1,0,2)
              C
              C
              C           IF(IRC.NE.0) GOTO 1000
              C
              C     NOW CALL INVL TO INVERT L
              C
              C
0069                    CALL INVL
0070                  CALL CLOCK(1,TIME)
0071                  ITIMDI = TIME * 100000
0072                 IF(PRTOPT .GT.2)
             1CALL PRTMAT(LINV,1,M,1, 1,M,ISIZ,1,0,2)
              C
              C
0073                    IF (IRC.NE.0)GOTO 1000
              C
              C
              C     GENERATE OPTIMUM PT.
              C
              C
0074                  DO 80 I=1,M
0075          80      SOLNX(I)=RAND(III)*HIXLOX + LOX
0076                 IF(PRTOPT.GT.1)WRITE(6,159)(SOLNX(I),I=1,M)
0077           159   FORMAT(' ',5X,'SOLNX =',(/9D14.7))
              C
              C
              C     GENERATE +VE LM FOR CONS. IN FINAL BASIS
              C
              C
0078                    DO 90  I=1,K
0079                    LM(I) = 0.D0
0080                  IF(I.LE.SOLNQ)LM(I) = RAND(III)*HILM
0081                        DO 100 J= 1,M
              C
              C
              C     GENERATE THE CONSTRAINT I
              C
              C
0082                  C(J,I) = RAND(III)*HICLOC +LOC
0083          100     CONTINUE
0084                   S =DSQRT(SINNER(C(1,I),C(1,I),M))
0085                   DO 110 J=1,M
```

```
0086          110         C(J,I) = C(J,I)/S
0087                      B(I) = SINNER(C(1,I),SOLNX,M)
0088                      XLAMDA(I) = 0.D0
0089                    IF(I.GT.SOLNQ)XLAMDA(I) = RAND(III) * HIB
0090                     B(I) = B(I) - XLAMDA(I)
0091                     IF(PRTOPT.GT.2)WRITE(6,158)J,(C(J,I),J=1,M)
0092           158     FORMAT(' ',5X,'CONSTRAINT ',I5,/(9D14.7))
0093          90         CONTINUE
0094                     IF(PRTOPT.GT.1)WRITE(6,164)(LM(I),I=1,SOLNQ)
0095           164     FORMAT(' ',5X,'L.M. OF ACTIVE CONS.',(/9D14.7))
0096                     IF(PRTOPT.GT.1)WRITE(6,165)(B(I),I=1,K)
0097           165     FORMAT(' ',5X,'RHS OF CONSTRAINTS',(/,9D14.7))
0098                     IF(PRTOPT.GT.1)WRITE(6,166)(XLAMDA(I),I=1,K)
0099           166     FORMAT(' ',5X,'LAMDAS OF CONSTRAINTS',(/9D14.7))
               C
               C
               C
               C     CALCULATE A = NQ *LM
               C
               C
0100                        DO 130 I=1,M
               C
               C     CALC GX
               C
0101          130          TEMP(I) = SINNER(L(1,I),SOLNX(I),M-I+1)
0102                       DO 140 I=1,M
0103          140          TEMP1(I) = SINNER(L(I,1),TEMP(1),I,ISIZ)
0104                     IF(SOLNQ.EQ.0)GOTO200
0105                       DO 150 I=1,M
0106          150        A(I) = SINNER(C(I,1),LM(1),SOLNQ,ISIZ)-TEMP1(I)
0107          200          DO 210 I=1,M
0108          210          X(I)=SOLNX(I)
0109                     IF(PRTOPT.GT.1)WRITE(6,167) (A(J),J=1,M)
0110           167     FORMAT(' ',5X,'A VECTOR IN F ',(/9D14.7))
               C
               C
               C      CALC GRADIENT AND FIND THE OBJ. FN
               C
               C
0111                     SOLNF = 0.5 * SINNER(SOLNX,TEMP1,M) + SINNER(SOLNX,A,M)
0112           1000       WRITE(IUNIT6,1005)IRC,SOLNF
0113           1005    FORMAT(' ',4X,'  END GENER, IRC = ',I5,'  SOLN F =',F16.7)
0114                     RETURN
0115                     END
```

```
FORTRAN IV G LEVEL  21                  DECOMP              DATE = 80105

0001                  SUBROUTINE DECOMP (IRC)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                  COMMON C2
0003                  COMMON /C02/LG
0004                  REAL*8 LG(30,31),L(30,30),G(30,30)
0005                  EQUIVALENCE(LG(1,1),L(1,1))
0006                  EQUIVALENCE(LG(1,2),G(1,1))
                  COMMON C2 END
                  COMMON 09
0007                 COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    1,    ISEED,ISOPT,IAOPT,IDOPT
0008                  INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON C9 END
                  COMMON 15
0009                     COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
                  COMMON 15 END
                  COMMON 17
0010                  COMMON/C17/IUNIT5,IUNIT6,IUNIT8
                  COMMON 17 END
                  COMMON 21
0011                  COMMON/C21/IAD,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                    1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  C     THIS SR WILL DECOMPOSE G, A +VE DEF. SYMM. MATRIX INTO ITS
                  C      CHOLESKY FACTORS, LL(T).
                  C      G IS ASSUMED TO BE STORED AS UPPER TRIANGULAR
                  C
                  C      TOLDCM IS TOLERANCE REQUIREMENT OF DECOMP.
                  C       (I.E.) G - LL(T) > TOLDCM (ABS. ON BOTH SIDES)
                  C
                  C    FOR TESTING GENERATE G
0012                     IRC = 0
0013                     NSQRT = NSQRT + M
0014                     L(1,1) = DSQRT(G(1,1))
0015                     IF(M.LE.1) GOTO 100
0016                      DO 20 I = 2,M
0017                         DO 20 J=1,I
0018                          SUM = 0.D0
0019                          IF(J.GT.1)SUM = SINNER(L(I,1),L(J,1),J-1,ISIZ,ISIZ)
0020                     NOPS = NOPS + J - 1
0021                          L(I,J) = G(J,I) - SUM
0022                          IF(I.EQ.J) GOTO 10
0023                          L(I,J) = L(I,J)/L(J,J)
0024                     NOPS = NOPS + 1
0025                          GOTO 20
0026              10          L(I,J) = DSQRT(L(I,J))
0027              20       CONTINUE
                  C     NOW VERIFY BY FINDING LL(T)
                  C
                  C         DO 30 I=1,M
                  C           DO 30 J= 1,I
                  C           DIFF = G(J,I) - SINNER(L(I,1),L(J,1),J,ISIZ,ISIZ)
                  C           IF(DABS(DIFF).LE.TOLDCM)GOTO 30
                  C           IRC = IRC + 1
                  C           WRITE(6,25)I,J,G(J,I),DIFF
                  C25         FORMAT(' ',5X,'DECOMP',5X,'I,J,G(J,I),G-LL(T)',2I5,2F20.10)
                  C30         CONTINUE
0028              100         RETURN
0029                          END
```

```
FORTRAN IV G LEVEL  21                    INVL                    DATE = 80105

0001                   SUBROUTINE INVL
0002                     IMPLICIT REAL*8(A-H,O-Z)
                  COMMON 02
0003                     COMMON /C02/LG
0004                     REAL*8 LG(30,31),L(30,30),G(30,30)
0005                     EQUIVALENCE(LG(1,1),L(1,1))
0006                     EQUIVALENCE(LG(1,2),G(1,1))
                  COMMON 02 END
                  COMMON 08
0007                     COMMON /C08/   FXLINR
0008                     REAL*8 FXLINR(30,32),FX(30),LINV(30,30),FR(30,30)
0009                     EQUIVALENCE(FXLINR(1,1),FX(1))
0010                     EQUIVALENCE(FXLINR(1,2),LINV(1,1))
0011                     EQUIVALENCE(FXLINR(1,3),FR(1,1))
                  COMMON 08 END
0012                   COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                      1,    ISEED,ISOPT,IAOPT,IDOPT
0013                   INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON C9 END
                  COMMON 15
0014                     COMMON /C15/TOLDCH,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
                  COMMON 15 END
                  COMMON 17
0015                   COMMON/C17/IUNIT5,IUNIT6,IUNIT8
                  COMMON 17 END
                  COMMON 21
0016                   COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                      1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  C      THIS SR INVERTS L AND INVERSE IS PUT IN LINV
                  C      LINV IS LOWER TRIANGULAR, SO IS L
                  C
0017                       IRC = 0
0018                       LINV(1,1) = 1.DO/L(1,1)
0019                       NOPS = NOPS + 1
0020                       IF(M.LE.1) GOTO 100
0021                     DO 10 I=2,M
0022                       LINV(I,I) = 1.DO/L(I,I)
0023                       IM1 = I-1
0024                       DO 20 J1=1,IM1
0025                     J = I-J1
0026            20       LINV(I,J) = -SINNER(LINV(I,J+1),L(J+1,J),J1,ISIZ)/L(J,J)
0027                     NOPS = NOPS + I * (I+1)/2 - 1
0028            10       CONTINUE
                C        DO 30 I = 1,M
                C           DO 30 J=1,I
                C             DIFF = 0.DO
                C             IF(I.EQ.J) DIFF=1.DO
                C             DIFF = DIFF - SINNER(L(I,J),LINV(J,J),I-J+1,ISIZ)
                C             IF(ABS(DIFF).LE.TOLINV)GOTO 30
                C           WRITE(6,25)I,J,DIFF
                C25          FORMAT(' ',5X,'INVL  ',5X,'I,J,I - LL(-1)',2I5,F15.7)
                C           IRC = IRC + 1
                C30        CONTINUE
0029            100        RETURN
0030                       END
```

```
FORTRAN IV G LEVEL  21                  SOLVL                   DATE = 80105

0001                  SUBROUTINE SOLVL(Y,S)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                    COMMON C2
0003                     COMMON /C02/LG
0004                     REAL*8 LG(30,31),L(30,30),G(30,30)
0005                     EQUIVALENCE(LG(1,1),L(1,1))
0006                     EQUIVALENCE(LG(1,2),G(1,1))
                    COMMON 02 END
                    COMMON 09
0007                     COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                   1,    ISEED,ISOPT,IAOPT,IDOPT
0008                     INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    COMMON 09 END
                    COMMON 21
0009                     COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NRE IN,NXCOR
                   1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                    COMMON 21 END
                    C        THIS SR WILL SOLVE FOR Y IN L Y = S
                    C
                    C     L IS LOWER TRIANGULAR MATRIX
0010                      REAL*8  Y( 1),S( 1)
                    C
                    C
                    C        Y IS OUTPUT FROM THIS ROUTINE
                    C        P IS VECTOR INPUT
                    C        M IS NO. OF VARIABLES (= M)
                    C        ISIZ IS OFFSET FOR L (30)
                    C
                    C
                    C
0011                        Y(1) = S(1)/L(1,1)
0012                        IF(M.LE.1)GOTO 100
0013                        DO 10 I=2,M
0014                        Y(I) = (S(1) - SINNER(L(I,1),Y(1),I-1,ISIZ))/L(I,I)
                    C       DIFF = S(I)-SINNER(L(I,1),Y(1),I,ISIZ)
0015                10       CONTINUE
0016                 100    NOPS = NOPS + M * (M+1)/2
0017                        RETURN
0018                        END
```

```
FORTRAN IV G LEVEL  21                  SOLVLT                  DATE = 80105

0001                      SUBROUTINE SOLVLT(Y,S)
0002                      IMPLICIT REAL*8(A-H,O-Z)
                    COMMON C2
                      COMMON /C02/LG
0003                  REAL*8 LG(30,31),L(30,30),G(30,30)
0004                  EQUIVALENCE(LG(1,1),L(1,1))
0005                  EQUIVALENCE(LG(1,2),G(1,1))
0006                COMMON C2 END
                    COMMON C9
                   COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
0007              1,    ISEED,ISOPT,IAOPT,IDOPT
0008                INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    COMMON C9 END
                    COMMON 21
                   COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NRE IN,NXCOR
0009              1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                    COMMON 21 END
                    C
                    C      THIS SR SOLVES FOR Y IN L(T)Y = S
                    C         L IS STORED LOWER TRIANGULAR
                    C         M IS NUMBER OF VARIABLES
                    C
                    C
0010                      REAL*8 Y( 1),S( 1)
0011                      IRC = 0
0012                      Y(M) = S(M)/L(M,M)
0013                      IF(M.LE.1)GOTO  100
0014                      I = M
0015                      MM1 = I - 1
0016                      DO  10 J= 1,MM1
0017                      I=I-1
0018                      Y(I) = (S(I)- SINNER(L(I+1,I),Y(I+1),M-I))/L(I,I)
0019               10     CONTINUE
0020                100   NOPS = NOPS + M * (M+1)/2
0021                      RETURN
0022                      END
```

```
FORTRAN IV G LEVEL  21                  SOLVR              DATE = 80105

0001                  SUBROUTINE SOLVR(Y,S,ILIM,IST)
0002                    IMPLICIT REAL*8(A-H,O-Z)
              COMMON 04
0003                  COMMON /C04/Q1R
0004                  REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0005                  EQUIVALENCE(Q1R(1,1),Q1(1,1))
0006                  EQUIVALENCE(Q1R(1,3),R(1,1))
              COMMON 04 END
              COMMON 09
0007                COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                   1,    ISEED,ISOPT,IAOPT,IDOPT
0008                INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
              COMMON 09 END
              COMMON 21
0009                COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                   1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
              COMMON 21 END
              COMMON 22
0010                  COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0011                  REAL*8 TITLE(15)
0012                  INTEGER PRTOPT
              COMMON 22 END
              C       THIS SR WILL SOLVE FOR Y IN R Y = P
              C
              C          R IS UPPER TRIANGULAR AND STORED SUCH
              C          Y IS OUTPUT AND IS A VECTOR
              C          S IS INPUT AND RHS SIDE VECTOR
              C       ILIM IS NUMBER OF VARIABLES
              C          ISIZ IS OFFSET FOR R
              C
              C
0013                     REAL*8  Y( 1),S(1)
              C
0014                     IF(IST.LT.ILIM)GOTO 110
0015                     Y(IST) = S(IST)/R(IST,IST)
0016                     IF(IST.LE.ILIM)GOTO 100
0017                     I=IST
0018                     I1 = IST-ILIM
0019                     DO 10 J= 1,I1
0020                     I=I-1
0021                     Y(I)=(S(I)-SINNER(R(I,I+1),Y(I+1),IST-I,ISIZ))/R(I,I)
              C              DIFF = S(I) - SINNER(R(I,I),Y(I),IST-I+1,ISIZ)
              C              IF(ABS(DIFF).LE.0.00005D0)GOTO 10
              C              WRITE(6,64)I,DIFF,S(I),(R(I,JK),JK=I,IST)
              C64    FORMAT(' ',5X,'SOLVR- I, DIFF,S,RI,JK,JK=1,IST)',/I5,(9D12.6))
0022          10       CONTINUE
              C
              C       VERIFY IF R Y = S?
              C
0023           100      NOPS = NOPS + (IST-ILIM+1)*(IST-ILIM+2)/2
0024                   IF(PRTOPT.GT.1)WRITE(6,105)ILIM,IST,(Y(J),J=ILIM,IST)
0025           105     FORMAT(' ','SOLVR- IFRO - ITO',2I5,/(9D14.7))
0026           110     RETURN
0027                     END
```

```
FORTRAN IV G LEVEL  21                  SOLVRT              DATE = 80105

0001                  SUBROUTINE SOLVRT(Y,S)
0002                    IMPLICIT REAL*8(A-H,O-Z)
               COMMON C4
0003                  COMMON /C04/Q1R
0004                  REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0005                  EQUIVALENCE(Q1R(1,1),Q1(1,1))
0006                  EQUIVALENCE(Q1R(1,3),R(1,1))
               COMMON C4 END
               COMMON 09
0007                 COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    1,    ISEED,ISOPT,IAOPT,IDOPT
0008                  INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
               COMMON 09 END
               COMMON 21
0009                 COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NRE IN,NXCOR
                    1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
               COMMON 21 END
               C
               C          R IS UPPER TRIANGULAR AND STORED SUCH
               C          Y IS OUTPUT AND IS A VECTOR
               C          S IS INPUT AND RHS SIDE VECTOR
               C          M IS NUMBER OF VARIABLES
               C
               C
0010                    REAL*8  Y( 1),S(1)
               C
0011                  Y(1) = S(1)/R(1,1)
0012                  IF(Q.LE.1)GOTO 100
0013                  DO 10 I=2,Q
0014                  Y(I) = (S(I)-SINNER(R(1,I),Y(1),I-1))/R(I,I)
0015           10     CONTINUE
0016            100    NOPS = NOPS + Q * (Q+1)/2
0017                  RETURN
0018                  END
```

```
FORTRAN IV G LEVEL  21                    UNCONM                    DATE = 80105

0001                  SUBROUTINE UNCONM
0002                     IMPLICIT REAL*8(A-H,O-Z)
                  COMMON 09
0003                 COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    1,    ISEED,ISOPT,IAOPT,IDOPT
0004                  INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON 09 END
                  COMMON 10
0005                    COMMON /C10/UNCONX,UNCONF
0006                    REAL*8 UNCONX(30),UNCONF
                  COMMON 10 END
                  COMMON 17
0007                 COMMON/C17/IUNIT5,IUNIT6,IUNIT8
                  COMMON 17 END
                  COMMON 20
0008                  COMMON /C20/A
0009                  REAL*8 A(30)
                  COMMON 20 END
                  COMMON 21
0010                 COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                    1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  COMMON 22
0011                  COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0012                  REAL*8 TITLE(15)
0013                  INTEGER PRTOPT
                  COMMON 22 END
                  C
                  C        THIS SR SOLVES FOR THE UNCONSTRAINED MIN OF THE PROBLEM
                  C
0014                  CALL SOLVL(UNCONX,A)
0015                  CALL SOLVLT(UNCONX,UNCONX)
0016                  DO 10 I=1,M
0017             10   UNCONX(I) = -UNCONX(I)
0018                  UNCONF = 0.5 * SINNER(A,UNCONX,M)
0019                 WRITE(IUNIT6,11)UNCONF
0020              11 FORMAT(' ',6X,'UNCONSRAINED MINIMIUM = ', F16.7)
0021                  IF(PRTOPT.GT.1)WRITE(IUNIT6,12)(UNCONX(I),I=1,M)
0022              12 FORMAT(' ',5X,'UNCONX'/(9D14.6/))
0023                  NOPS = NOPS + M
0024                  RETURN
0025                  END
```

```
FORTRAN IV G LEVEL  21                    RAND                      DATE = 80105

0001                        FUNCTION RAND(I)
0002                  REAL*8 RAND
0003                  I=I*59049
0004                  IF (I.LT.0) I=I+2147483647+1
0005                   RAND=I*0.4656613D-9
0006                  RETURN
0007                  END
```

```
FORTRAN IV G LEVEL  21                    GRA                    DATE = 80105

0001                      SUBROUTINE GRA
0002                    IMPLICIT REAL*8(A-H,O-Z)
                  COMMON 02
0003                    COMMON /C02/LG
0004                    REAL*8 LG(30,31),L(30,30),G(30,30)
0005                    EQUIVALENCE(LG(1,1),L(1,1))
0006                    EQUIVALENCE(LG(1,2),G(1,1))
                  COMMON 02 END
                  COMMON 09
0007                  COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                     1,    ISEED,ISOPT,IAOPT,IDOPT
0008                  INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON C9 END
                  COMMON 11
0009                    COMMON /C11/X
0010                    REAL*8 X(30)
                  COMMON 11 END
                  COMMON 14
0011                    COMMON /C14/F,GRAD
0012                    REAL*8 F,GRAD(30)
                  COMMON 14 END
                  COMMON 20
0013                    COMMON /C20/A
0014                    REAL*8 A(30)
                  COMMON 20 END
                  COMMON 21
0015                  COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                     1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  COMMON 22
0016                    COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0017                    REAL*8 TITLE(15)
0018                    INTEGER PRTOPT
                  COMMON 22 END
                  C
                  C
                  C THIS SR CALCULATES THE GRADIENT OF THE OBJ.FN
                  C   AND EVALUATES THE VALUE OF THE FN.
                  C
                  C     INPUT     - L (LOWER TRIANGULAR MATRIX)
                  C                 X (PT. X WHERE GRADIENT REQD)
                  C                 A LINEAR PART OF THE OBJ. FN.
                  C                 M NO. OF VARIABLES
                  C                 ISIZ  SIZE OF THE MATRIX L (30)
                  C
                  C     OUTPUT    - GRAD  GRADIENT VECTOR
                  C                 F     OBJ. FN.
                  C                 IRC   RETURN CODE (0)
                  C
                  C     TEMP. VAR   TEMP  VECTOR OF LENGTH ISIZ
                  C
                  C
                  C     METHODOLOGY:
                  C                  TEMP = L(T) * X
                  C                  GRAD = L * TEMP
                  C                  F    = 0.5*( A(T) * X + X(T) * GRAD)
```

```
C
C     DATE     JULY 25 1979
C
      REAL*8 TEMP(30)
        IRC = 0
        DO 10 I = 1 , M
        TEMP(I) = SINNER(L(I,1),X(I),M-I+1)
10      GRAD(I) = A(I) + SINNER(L(I,1),TEMP(1),I,ISIZ)
        F = 0.5 * (SINNER(X,GRAD,M) + SINNER(A,X,M))
      NOPS = NOPS + 2*M*(M+2) + 1
20      FORMAT (' ',5X,'F = ',F16.7)
        IF(PRTOPT.GT.0)WRITE(6,20) F
       IF(PRTOPT.GT.1)WRITE(6,21)(GRAD(J),J=1,M)
21     FORMAT(' ',5X,'GRADIENT',(/9D14.7))
        RETURN
        END
```

```
FORTRAN IV G LEVEL  21                    PRTMAT              DATE = 80105

0001                  SUBROUTINE PRTMAT(ARAY,NSTART,NROWS,NCOL1R,NRTCHG,MXCOL,
                     1                  ROFSET,RPOFST,POFSET,IWHAT)
0002                  IMPLICIT REAL*8(A-H,O-Z)
                C
                C
                C     THIS SR WILL PRINT A MATRIX, UPPER, LOWER, UPPER HESSENBERG
                C           SQUARE
                C     ARAY IS THE ARRAY THAT GETS REFERNCED
                C     NSTART IS THE FIRST ELEMENT IN (HE MATRIX
                C     NROWS TH NO. OF ROWS TO BE PRINTED
                C     NCOL1R IS THE NO. OF COLUMNS IN 1ST ROW TO BE PRINTED
                C     NRTCHG IS THE RATE OF CHANGE IN NO. OF COLUMNS/ROWCHANGE
                C           1 FOR LOWER TRIANGULAR, UPPER HESSENBERG
                C           0 FOR SQUARE
                C          -1 FOR UPPER TRIANGULAR
                C     MXCOL  IS MAX. NO OF COLUMNS TO BE PRINTED IN ANY ROW
                C     ROFSET IS ADJACENT ELEMENTS IN SAME ROW OFFSET(USUALLY ISIZ)
                C     RPOFST IS ADJACENT ROW TO BE PRINTED ELEMENT OFFSET(ISIZ+1 FOR
                C              UPPER TRIANGULAR
                C     POFSET IS FOR PRINTING OFFSET TO SHOW (EG - UPPER TRIANGULAR
                C              WILL LOOK UPPER TRIANGULAR IF IT IS 1
                C     IWHAT IS 1 FOR MATRIX G
                C              2 L
                C              3 LINV
                C              4 QTLINV
                C              5 Q1
                C              6 R
                C
                C
0003                    REAL*8 ARAY(1)
0004                   REAL*8 IDATA(6)/'   G    ','   L    ',
                     1'  LINV  ',' QTLINV ','   Q1   ','    R   '/
0005                    INTEGER * 4 ROFSET,RPOFST,POFSET,CUREL,PREL
0006                    REAL*8 PARAY(30)
                C
                C
0007                  WRITE(6,5)IDATA(IWHAT)
0008             5    FORMAT(' ',5X,'MATRIX',4X,A8)
0009                  DO 40 I= 1,NROWS
0010                  PREL = NSTART + (I-1) * RPOFST
0011                  CUREL = (I-1)*POFSET
0012                  IF(CUREL.LE.0) GOTO 20
0013                  DO 10 J=1,CUREL
0014            10    PARAY(J)=0.D0
0015            20    NCOL = NCOL1R+NRTCHG*(I-1)
0016                  IF(NCOL.GT.MXCOL)NCOL=MXCOL
0017                  DO 30 J=1,NCOL
0018                  CUREL=CUREL+1
0019                  PARAY(CUREL)=ARAY(PREL)
0020                  PREL=PREL+ROFSET
0021            30    CONTINUE
0022                  WRITE(6,35)(PARAY(J),J=1,CUREL)
0023            35    FORMAT(' ',9D14.6)
0024            40    CONTINUE
0025                  RETURN
0026                  END
```

```
FORTRAN IV G LEVEL  21                PROJ              DATE = 80105

0001                  SUBROUTINE PROJ
0002                     IMPLICIT REAL*8(A-H,O-Z)
                  COMMON 01
0003                  COMMON /C01/PROJV,LENY,QTLZ
0004                  REAL * 8 PROJV(30),LENY,QTLZ(30)
                  COMMON 01 END
                  COMMON 03
0005                  COMMON /C03/QTLINV
0006                  REAL*8  QTLINV(30,30)
                  COMMON 03 END
                  COMMON 09
0007                  COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                     1,    ISEED,ISOPT,IAOPT,IDOPT
0008                  INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON 09 END
                  COMMON 21
0009                  COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NRE IN,NXCOR
                     1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  COMMON 22
0010                  COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0011                  REAL*8 TITLE(15)
0012                  INTEGER PRTOPT
                  COMMON 22 END
                  C
                  C      THIS SR WILL FIND PROJECTION VECTOR PROJV FOR A GIVEN QTLZ
                  C
                  C      INPUT
                  C        QTLINV IS THE FACTOR OF B (Q(T)L(-1))
                  C        M - NO. OF VARIABLES
                  C        Q - NO. ACTIVE CONSTRAINTS
                  C        ISIZ - USED TO FIND THE OFFSET (30)
                  C
                  C      OUTPUT
                  C        PROJV - PROJECTED VECTOR
                  C        LENY - LENGTH OF THE PROJECTED VECTOR
                  C
                  C      METHODOLOGY:
                  C        IF Q = M THEN Y=LENY=0.0
                  C           ELSE
                  C        QTLZ=(W1 W2) WHERE W2 IS OF SIZE M-Q
                  C       CALCULATE W2
                  C       LET W= (0 W2)
                  C       PROJV = QTLINV(T) * W
                  C       CALCULATE LENGTH OF PROJV
                  C
                  C       DATE : JULY 28 1979
                  C
                  C
0013                    INTEGER QP1,MMQ
                  C
0014                    QP1=Q+1
0015                    MMQ=M-Q
0016                    LENY=0.D0
0017                    IF(Q.LT.M)GOTO 5
0018                     DO 3 I=1,M
```

```
0019          3         PROJV(I)=0.D0
0020                    GOTO 30
              C
              C          CALCULATE QTLINV(T) (0 W2)
              C
0021          5        DO 20 I=1,M
0022          20       PROJV(I)=SINNER(QTLINV(QP1,I),QTLZ(QP1),MMQ)
0023                   LENY=DSQRT(SINNER(PROJV,PROJV,M))
0024                   NSQRT = NSQRT + 1
0025           30      IF(PRTOPT.GT.1)WRITE(6,11)LENY
0026           11     FORMAT(' ',5X,'LENGTH OF PROJV  = ', F16.7)
0027                  IF(PRTOPT.GT.1)WRITE(6,12)(PROJV(I),I=1,M)
0028           12     FORMAT(' ',5X,'PROJV '/(9D14.6/))
0029                   NOPS = NOPS + (MMQ+1)*M
0030                   RETURN
0031                   END
```

```
FORTRAN IV G LEVEL  21                    RBAR                    DATE = 80105

0001                  SUBROUTINE RBAR(Z,RZ,IFRO,ITO)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                    COMMON 03
0003                     COMMON /C03/QTLINV
0004                     REAL*8  QTLINV(30,30)
                    COMMON 03 END
                    COMMON 09
0005                    COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                       1,    ISEED,ISOPT,IAOPT,IDOPT
0006                     INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    COMMON 09 END
                    COMMON 21
0007                    COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                       1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                    COMMON 21 END
                    COMMON 22
0008                     COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0009                     REAL*8 TITLE(15)
0010                     INTEGER PRTOPT
                    COMMON 22 END
                    C
                    C     THIS SR WILL CALCULATE QTLINV * Z = RZ FOR GIVEN Z
                    C
                    C     INPUT
                    C      Z - VECTOR FOR WHICH RZ IS       REQD. (GRAD OR N(Q+1))
                    C      QTLINV - Q(T)L(-1) MATRIX
                    C      Q - SIZE OF R
                    C      M - SIZE OF QTLINV
                    C      IFRO,ITO- THE LOWER AND UPPER BOUNDS FOR WHICH RZ IS DESIR.
                    C
                    C     OUTPUT
0011                     REAL*8 Z( 1),RZ( 1)
                    C
                    C
0012                        IF(ITO.LT.IFRO)RETURN
0013                        DO 10 I=IFRO,ITO
0014                 10     RZ(I)=SINNER(QTLINV(I,1),Z(1),M,ISIZ)
0015                        NOPS=NOPS+(ITO-IFRO+1)*M
0016                     IF(PRTOPT.GT.1)WRITE(6,12)IFRO,ITO,(RZ(I),I=IFRO,ITO)
0017                 12  FORMAT(' ',5X,'BAR,IFRO-ITO',2I5,/(9D14.6))
0018                     RETURN
0019                     END
```

```
0001                  SUBROUTINE MXMIN(Z,MANY,MAXMIN,POSNEG,TOL,WHICH,VALUE)
0002                    IMPLICIT REAL*8(A-H,O-Z)
                C
                C      THIS SR WILL CALCULATE MAX OR MIN OF A VECTOR
                C       FURTHER, IT CAN SELECT AMONG THOSE THAT ARE +VE OR NEG.
                C
                C      INPUT
                C       Z VECTOR FOR WHICH MAX OR MIN HAS TO BE FOUND
                C       MANY - NO. OF NUMBERS IN Z
                C       MAXMIN - 1 - MOST, 0 - RANDOM, -1 - FOR LEAST
                C       POSNEG - 1 - +VE, 0 - ALL OK, -1 - NEG. (CHOOSE ONLY THES
                C       TOL - TOLERANCE TO BE IGNORED - USED ONLY WHEN POSNEG ~=0
                C
                C     OUTPUT
                C       WHICH - 0 IF NONE SATISFY REQUIREMENT, ELSE SET TO THE
                C               INDEX IN Z CORRESPONDING TO THE OPTIMUM
                C       VALUE - VALUE OF THE OPTIMUM
                C
                C     METHODOLOGY:
                C       FIND THE INTERVAL TO BE ACCEPTED FOR CONSIDERATION
                C        (HIVAL & LOVAL)
                C       FOR ALL THE NUMBERS THAT BELONG IN THE INTERVAL FIND
                C       THE BEST ONE THAT MEETS THE REQUIREMENT(MAX OR MIN)
                C
                C     DATE : JULY 28, 1979
                C
0003                    REAL*8 Z(1),TOL,VALUE,HIVAL,LOVAL
0004                    INTEGER POSNEG,WHICH,MAXMIN,MANY
0005                     HIVAL  = 1.E70
0006                     LOVAL  =-1.E70
0007                     IF(POSNEG)10,30,20
0008              10     HIVAL  = -TOL
0009                     GOTO 30
0010              20     LOVAL  = TOL
0011              30     WHICH  = 0
0012                     VALUE  = 1.E70
0013                     IF(MAXMIN.GT.0)VALUE = 0.D0
0014                     ICNT = 0
0015                     DO 90 I=1,MANY
0016                     IF(Z(I).GE.LOVAL.AND.Z(I).LE.HIVAL)GOTO 50
0017                     GOTO 90
0018              50     ICNT = ICNT + 1
0019                        ZI = DABS(Z(I))
0020                     IF(MAXMIN)60,80,70
0021              60     IF(ZI   .GT.VALUE)GOTO 90
0022                     GOTO 80
0023              70     IF(ZI   .LT.VALUE)GOTO 90
0024              80     VALUE=ZI
0025                     WHICH=I
0026              90     CONTINUE
                C     RANDOM OPTION IS NOT IN EFFECT
0027                      IF(WHICH.NE.0)VALUE = Z(WHICH)
0028                     IF(WHICH.EQ.0)VALUE = 0.D0
0029                     RETURN
0030                     END
```

```
FORTRAN IV G LEVEL  21                    ADD                    DATE = 80105

0001                   SUBROUTINE ADD(ICONAD)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                 COMMON 01
0003                   COMMON /C01/PROJV,LENY,QTLZ
0004                   REAL * 8 PROJV(30),LENY,QTLZ(30)
                 COMMON 01 END
                 COMMON 03
0005                   COMMON /C03/QTLINV
0006                   REAL*8  QTLINV(30,30)
                 COMMON 03 END
                 COMMON 04
0007                   COMMON /C04/Q1R
0008                   REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0009                   EQUIVALENCE(Q1R(1,1),Q1(1,1))
0010                   EQUIVALENCE(Q1R(1,3),R(1,1))
                 COMMON 04 END
                 COMMON 05
0011                   COMMON /C05/C,B,XLAMDA,ICONOK
0012                   REAL*8 C(30,90),B(90),XLAMDA(90)
0013                   INTEGER*2 ICONOK(90)
                 COMMON 05 END
                 COMMON 09
0014                 COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    1,    ISEED,ISOPT,IAUPT,IDOPT
0015                 INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                 COMMON 09 END
                 COMMON 12
0016                   COMMON /C12/NQ
0017                   INTEGER NQ(30)
                 COMMON 12 END
                 COMMON 17
0018                 COMMON/C17/IUNIT5,IUNIT6,IUNIT8
                 COMMON 17 END
                 COMMON 21
0019                 COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                    1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                 COMMON 21 END
                 COMMON 22
0020                   COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0021                   REAL*8 TITLE(15)
0022                   INTEGER PRTOPT
                 COMMON 22 END
                 COMMON 23
0023                   COMMON/C23/GC,GH,GP,GB
0024                   REAL * 8 GC(30),GH(30),GP(30),GB(30)
                 COMMON 23 END
0025                    INTEGER QP1
                 C
                 C     THIS SR. WILL ADD A CONSTRAINT TO THE BASIS AND UPDATE
                 C     THE OPERATORS, QTLINV AND R.
                 C
                 C      INPUT
                 C        Q # OF ACTIVE CONSTRAINTS
                 C        M # OF VARIABLES
                 C        ISIZ OFFSET FOR ROW STORAGE IN AN ARRAY
                 C        QTLZ = QTLINV * N(Q+1)
```

```
FORTRAN IV G LEVEL  21                ADD                DATE = 80105

C       R UPPER TRIANGULAR FACTOR OF B (L(-1)N(Q))
C       QTLINV SQUARE MATRIX (Q(T) L(-1))
C
C    N(Q+1) IS ASSUMED TO BE LIN. INDEPENDENT.
C
C     METHODOLOGY:
C        Q+1ST COL. OF R IS SET TO QTLZ
C        CALCULATE COEFFICIENTS IN THE GIVENS MATRIX REQD. TO
C        REDUCE THE Q+1ST COL. OF R TO 0.0 EVERYWHERE BELOW
C        ELEMENT Q+1.
C
C          THIS IS DONE BY WORKING FROM BOTTOM UP AND FINDING
C        GC(J),GH(J), J = M-1,Q+1,-1
C          THIS CALCULATION IS DONE IN A WAY DESCRIBED IN DANIEL,
C        GRAGG,KAUFMAN ET AL. PAPER SO AS TO AVOID OVERFLOW AND
C        UNDERFLOW.
C
0026              QP1 = Q+1
0027              MM1 = M-1
0028             MMQ = M - Q
C        NOW CALCULATE THE PRODUCT OF THE GIVENS MATRICES.
C        SINCE THE PRODUCT HAS A SPECIAL FORM AS DESCRIBED IN
C        GILL GOLUB SAUNDERS ET AL. PAPER.   CALCULATE
C        THE REQD. VECTORS TO PRODUCE THE PRODUCT.
C        PRODUCT MATRIX IS UPPER TRIANGULAR AND A SUBDIAGONAL.
C
C        QTLINV IS NOW UPDATED
C
C        NEW QTLINV = ( I 0  )   OLD QTLINV
C                     ( 0 Q1 )
C
C            WHERE Q1 IS THE PRODUCT OF GIVENS MATRICES AS ABOVE.
C
C        DATE JULY 29, 1979
C
C      QTLZ IS THE VECTOR Q(T)L(-1) * (N+).
C    IT MUST HAVE BEEN CALCULATED PRIOR TO CALL OF THE ROUTINE
C
C    CALCULATE GIVENS MATRIX COEFFICIENTS FROM M-1 THRU Q+1
C
C
0029        30       ICURR = M
0030                   IF(QP1.EQ.M)GOTO 70
0031                 DO 60 I = QP1,MM1
0032                 ICURR1 = ICURR - 1
0033                 CALL GIVENM(QTLZ(ICURR1),QTLZ(ICURR),GC(ICURR1),GH(ICURR1),
            1GP(ICURR1))
0034                 ICURR = ICURR1
0035          60     CONTINUE
C
C    NOW FIND THE PRODUCT WITH   GIVENS MATRICES
0036                 DO 64 I=1,M
0037                 IC=M
0038                 DO 63 J=QP1,MM1
0039                 IC1=IC-1
0040                 CALL GVNUPV(GC(IC1),GH(IC1),GP(IC1),QTLINV(IC1,I),
```

```
     1QTLINV(IC,I),NOPS)
0041          IC=IC-1
0042    63    CONTINUE
0043    64    CONTINUE
      C
0044    70    DO 65 I = 1,QP1
0045    65    R(I,QP1)=QTLZ(I)
      C
      C     UPDATE Q
      C
0046          Q = QP1
0047           CALL UPRIN(ICONAD)
0048           NADDS = NADDS + 1
0049          NQ(Q) = ICONAD
0050          IF(PRTOPT.GT.0)WRITE(IUNIT6,71)Q,(NQ(I),I=1,Q)
0051    71    FORMAT(' ',I7,' - IN BASIS ',3013)
0052          RETURN
0053          END
```

```
FORTRAN IV G LEVEL  21                  GIVENM            DATE = 80105

0001                    SUBROUTINE GIVENM(R1,R2,GC1,GH1,GMU)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                  COMMON 21
0003                    COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                       1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  C THIS SR WILL FIND THE COEFFICIENTS IN GIVENS MATRIX
                  C   FOR A PAIR OFNUMBERS OF THE RHS
                  C
                  C       ( GC1   GH1 )   R1     =     NEW R1
                  C       ( GH1  -GC1 )   R2                0.0 = NEW R2
                  C
                  C      ON RETURN FROM THE ROUTINE R2 WILL BE SET TO 0.0 & R1
                  C      WILL BE SET TO SIGN(MAX(ABS(R1),ABS(R2)) * SQRT(R1**2+R2**2))
                  C      GC & GH WILL BE CALCULATED AS EXPLAINED IN DANIEL,GRAGG,KAUFM
                  COMMON 22
0004                     COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0005                     REAL*8 TITLE(15)
0006                     INTEGER PRTOPT
                  COMMON 22 END
0007                      IF(R2         .NE.0.DO)GOTO 40
0008                      GC1 = 1.DO
0009                      GH1 = 0.DO
0010                      GMU=0.DO
0011                      GOTO 60
0012              40       IF(R1.NE.0.DO)GOTO 45
0013                       GC1=0.DO
0014                       GH1=1.DO
0015                       R1=R2
0016                       R2=0.DO
0017                       GMU=1.DO
0018                       GOTO 60
0019              45       RMU1=DABS(R1)
0020                       RMU2=DABS(R2)
0021                       IF(RMU1.LT.RMU2)GOTO 47
0022                       TAU=R1*DSQRT(1.DO+(RMU2/RMU1)**2)
0023                       GOTO 50
0024              47       TAU=R2*DSQRT(1.DO+(RMU1/RMU2)**2)
0025              50       GC1=R1/TAU
0026                       GH1=R2/TAU
0027                       GMU=R2/(R1+TAU)
0028                        R1=TAU
0029                        R2=0.DO
0030                       NSQRT=NSQRT+1
0031                       NOPS=NOPS+4
0032                     IF(GC1.LT.1.DO)GOTO 55
0033                     GC1 = 1.DO
0034                     GH1=0.DO
0035                     GOTO 60
0036              55      IF(GH1.LT.1.DO)GOTO 60
0037                     GC1=0.DO
0038                     GH1=1.DO
0039              60          IF(PRTOPT.GT.3)WRITE(6,61)R1,R2,GC1,GH1,GMU
0040              61         FORMAT(' ','R1,R2,GC1,GH1,GMU',5D16.8)
0041                         RETURN
0042                          END
```

```
FORTRAN IV G LEVEL  21                    DROP                    DATE = 80105

0001                  SUBROUTINE DROP(ICONDR)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                    COMMON C3
                          COMMON /C03/QTLINV
0003                      REAL*8  QTLINV(30,30)
0004                COMMON 03 END
                    COMMON 04
                          COMMON /C04/Q1R
0005                      REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0006                      EQUIVALENCE(Q1R(1,1),Q1(1,1))
0007                      EQUIVALENCE(Q1R(1,3),R(1,1))
0008                COMMON 04 END
                    COMMON 05
                          COMMON /C05/C,B,XLAMDA,ICONOK
0009                      REAL*8 C(30,90),B(90),XLAMDA(90)
0010                      INTEGER*2 ICONOK(90)
0011                COMMON 05 END
                    COMMON 09
0012                     COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                        1,    ISEED,ISOPT,IAOPT,IDOPT
0013                      INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    COMMON C9 END
                    COMMON 12
0014                        COMMON /C12/NQ
0015                        INTEGER NQ(30)
                    COMMON 12 END
                    COMMON 13
0016                       COMMON /C13/LM,IM
0017                       REAL*8 LM(30),IM(30)
                    COMMON 13 END
                    COMMON 17
0018                     COMMON/C17/IUNIT5,IUNIT6,IUNIT8
                    COMMON 17 END
                    COMMON 21
0019                     COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                        1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                    COMMON 21 END
                    COMMON 22
0020                      COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0021                      REAL*8 TITLE(15)
0022                      INTEGER PRTOPT
                    COMMON 22 END
                    COMMON 23
0023                      COMMON/C23/GC,GH,GP,GB
0024                      REAL * 8 GC(30),GH(30),GP(30),GB(30)
                    COMMON 23 END
                    C   THIS SR WILL UPDATE FACTORS WHILE DROPPING A CONS. FROM BASI
                    C
                    C    INPUT
                    C       P - THE INDEX OF THE CONSTRAINT IN THE BASIS TO BE DROPP
                    C       Q NO. OF CURRENTLY ACTIVE CONSTRAINTS.
                    C       M NO. OF VARIABLES
                    C       ISIZ OFFSET FOR MATRICES TO FIND ADJACENT ELEMENTS OF A
                    C       R   UPPER TRIANGULAR FACTOR
                    C       QTLINV SQUARE MATRIX OF SIZE M (FACTOR * L(-1))
                    C
```

```
C     METHODOLOGY:
C        IF (Q = 1 ) THEN QTLINV = LINV & RETURN
C        FIND GIVENS MATRIX COEFF. REQD TO REDUCE DIAGONAL
C        ELEMENTS OF R TO 0.0 FROM P+1 TO Q.
C        MOVE R TO LEFT BY 1 COL.SO THAT COLS. P+1 TO Q
C        OCCUPY COL. P TO Q-1
C
C        NOW CALCULATE THE PRODUCT OF THE GIVENS MATRICES.
C        THIS IS LOWER TRIANGULAR AND A SUPERDIAGONAL FORM
C        INSTEAD OF STORING THE MATRIX STORE VECTORS
C        GP,GB AND GH SO THAT ANY ELEMENT OF THE PRODUCT CAN BE
C        CALCULATED EASILY FROM THE ABOVE VECTORS.
C
C        NOW UPDATE QTLINV AS FOLLOWS.
C
0025               INTEGER PP1,QM1
0026                 REAL*8 MU,MU1,TAU
C                  NEW QTLINV = (I 0  0 )    OLD QTLINV
C                               (0 Q1 0 )
C                               (0 0  I )
C      WHERE Q1 IS PRODUCT OF GIVENS MATRICES OF THE FORM
C       LOWER TRIANGULAR WITH SUPER DIAGONAL
C
C
C      DATE JULY 29, 1979
C
0027                  P = ICONDR
0028                  JCONDR = NJ(ICONDR)
0029               PP1 = P + 1
0030               QM1 = Q - 1
0031               IF(Q.NE.1) GOTO 40
0032                CALL INTEQL
0033               GOTO 100
0034          40   IF(P.EQ.Q)GOTO 100
0035               IZ=0
C
C     CALCULATE GIVENS MATRIX COEFFICIENTS, P THRU QM1
C
0036               DO 80 I=P,QM1
0037               IP1 = I+1
0038               IP2 = I+2
0039               CALL GIVENM(R(I,IP1),R(IP1,IP1),GC(I),GH(I),GP(I))
0040               IF(PRTOPT.GT.4)WRITE(6,17)I,GC(I),GH(I)
0041           17  FORMAT(' ',5X,'ICURR1,GC,GH    ',15,2F14.7)
C
C     NOW UPDATE I,IP1 ROWS OF R TO THE RIGHT OF COL IP1
C
0042           65  IF(I.EQ.QM1)GOTO 75
0043               DO 70 J=IP2,Q
0044                CALL GVNUPV(GC(I),GH(I),GP(I),R(I,J),R(IP1,J),NOPS)
0045           70  CONTINUE
0046           75     CONTINUE
0047           80  CONTINUE
C
C     FIND THE PRODUCT OF GIVENS MATRICES
C
```

```
FORTRAN IV G LEVEL  21                  DROP                DATE = 80105

              C
              C     UPDATE QTLINV
              C
              C
0048                    DO 85 I=1,M
0049                    DO 83 J=P,QM1
0050                    JP1=J+1
0051                    CALL GVNUPV(GC(J),GH(J),GP(J),QTLINV(J,I),QTLINV(JP1,I),
             1NCPS)
0052           83       CONTINUE
0053           85       CONTINUE
              C    NOW MOVE P+1ST THRU Q TH COLS. OF R TO P THRU Q-1
              C    MOVE R TO LEFT
              C
0054                    IF(P.EQ.Q)GOTO 100
0055                    DO 90 I = P,QM1
0056                    IP1 = I+1
0057                    NQ(I) = NQ(IP1)
0058                    LM(I)=LM(IP1)
0059                    IM(I)=IM(IP1)
0060                    DO 90 J = 1,I
0061           90       R(J,I) = R(J,IP1)
0062          100       Q = Q-1
0063                    CALL UPRIN(-JCONDR)
0064                    NDROPS = NDROPS +1
0065                    IF(Q.EQ.0)NQ(1)=0
0066                    IF(PRTOPT.GT.0)WRITE(IUNIT6,71)Q,(NQ(I),I=1,Q)
0067           71       FORMAT(' ',I7,' - IN BASIS ',30I3)
0068                    RETURN
0069                    END
```

```
FORTRAN IV G LEVEL  21                    INTLQL              DATE = 80105

0001                    SUBROUTINE INTLQL
0002                     IMPLICIT REAL*8(A-H,O-Z)
               COMMON C3
0003                    COMMON /C03/QTLINV
0004                    REAL*8  QTLINV(30,30)
               COMMON C3 END
               COMMON C8
0005                     COMMON /C08/   FXLINR
0006                     REAL*8 FXLINR(30,32),FX(30),LINV(30,30),FR(30,30)
0007                     EQUIVALENCE(FXLINR(1,1),FX(1))
0008                     EQUIVALENCE(FXLINR(1,2),LINV(1,1))
0009                     EQUIVALENCE(FXLINR(1,3),FR(1,1))
               COMMON C8 END
               COMMON C9
0010                  COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                     1,    ISEED,ISOPT,IAOPT,IDOPT
0011                  INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
               COMMON C9 END
               C     SET QTLIV = LINV
               C
0012                     DO 30 I = 1,M
0013                     DO 30 J = 1,M
0014                     IF(J-I)10,20,20
0015           10        QTLINV(J,I) = 0.D0
0016                     GOTO 30
0017           20        QTLINV(J,I) = LINV(J,I)
0018           30        CONTINUE
0019                      RETURN
0020                      END

FORTRAN IV G LEVEL  21                    GVNUPV              DATE = 80105

0001                     SUBROUTINE GVNUPV(GC1,GH1,GMU,QT1,QT2,NOPS)
               C
               C  THIS SUBROUTINE MULTIPLIES A VECTOR OF 2 ELEMENTS WITH GIVENS
               C   MATRIX IN AN EFFICIENT MANNER
               C
               C
               C      ( GC1   GH1) ( QT1 ) = ( QT1 )
               C      ( GH1  -GC1) ( QT2 ) = ( QT2 )
               C
               C      OLD QT1,QT2 ARE REPLACED BY THE NEW VALUES ON EXIT
               C      GMU IS  R2/(R1+TAU) AS GIVEN IN BARTELS, GILL, ETAL
               C
0002                     IMPLICIT REAL*8(A-H,O-Z)
0003                     IF(GC1.NE.1.D0)GOTO 10
0004                     QT2=-QT2
0005                     GOTO 30
0006            10       IF(GH1.NE.1.D0)GOTO 20
0007                     QTEMP=QT1
0008                     QT1=QT2
0009                     QT2=QTEMP
0010                     GOTO 30
0011            20       QTEMP=QT1*GC1+QT2*GH1
0012                     QT2=GMU*(QT1+QTEMP)-QT2
0013                     QT1=QTEMP
0014                     NOPS=NOPS+3
0015            30       RETURN
0016                     END
```

```
FORTRAN IV G LEVEL  21                  LAMDA                   DATE = 80105

0001                  SUBROUTINE LAMDA
0002                     IMPLICIT REAL*8(A-H,O-Z)
                  COMMON 11
0003                    COMMON /C11/X
0004                    REAL*8 X(30)
                  COMMON 11 END
                  COMMON C5
0005                    COMMON /C05/C,B,XLAMDA,ICONOK
0006                    REAL*8 C(30,90),B(90),XLAMDA(90)
0007                    INTEGER*2 ICONOK(90)
                  COMMON 05 END
                  COMMON 09
0008                   COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                      1,    ISEED,ISOPT,IAOPT,IDOPT
0009                    INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON C9 END
                  COMMON 21
0010                   COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                      1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  COMMON 22
0011                    COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0012                    REAL*8 TITLE(15)
0013                    INTEGER PRTOPT
                  COMMON 22 END
                  C      THIS SR CALCULATES LAMDAS (I.E.) VIOLATIONS
                  C       INPUT
                  C        M     NO. OF VARIABLES
                  C        K     NO. OF CONSTRAINTS
                  C        X     POINT X
                  C        C(*,*) CONSTRAINT NORMALS 1/COLUMN OF C
                  C        B     VECTOR OF LIMITING VALUES(RHS OF C)
                  C        XLAMDA VECTOR C(T) * X - B
                  C
                  C        METHODOLOGY:
                  C         CALC. C(T)X - B
                  C         DATE   JULY 31 1979
0014                         DO 10 I=1,K
0015                10       XLAMDA(I) = SINNER(C(1,I),X(1),M) - B(I)
0016                        IF(PRTOPT.GT.1)WRITE(6,11)(XLAMDA(I),I=1,K)
0017                11      FORMAT(' ',5X,'LAMDAS',/(9D14.7))
0018                       NOPS = NOPS + M * K
0019                           RETURN
0020                           END
```

```
FORTRAN IV G LEVEL  21                    XCOR                    DATE = 80105

0001                      SUBROUTINE XCOR
0002                     IMPLICIT REAL*8(A-H,O-Z)
0003                      REAL*8 TEMP(30),TEMP1(30)
                C
                C      THIS SR WILL DO A X-CORRECTION SO THAT X MAY LIE ON THE
                C       DESIRED MANIFOLD
                C      ASSUMES NQ(*) IS REASONABLY ACCURATE.
                C      SEE ROSENS PAPER
                C
                C      METHODOLOGY:
                C              X = X - NQ(*)(T) * LAMDA
                C
                COMMON C3
0004                    COMMON /C03/QTLINV
0005                    REAL*8  QTLINV(30,30)
                COMMON C3 END
                COMMON 05
0006                    COMMON /C05/C,B,XLAMDA,ICONOK
0007                    REAL*8 C(30,90),B(90),XLAMDA(90)
0008                    INTEGER*2 ICONOK(90)
                COMMON 05 END
                COMMON 09
0009                  COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                     1,    ISEED,ISOPT,IAOPT,IDOPT
0010                   INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                COMMON C9 END
                COMMON 11
0011                    COMMON /C11/X
0012                    REAL*8 X(30)
                COMMON 11 END
                COMMON 12
0013                    COMMON /C12/NQ
0014                    INTEGER NQ(30)
                COMMON 12 END
                COMMON 21
0015                  COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                     1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                COMMON 21 END
                COMMON 22
0016                    COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0017                    REAL*8 TITLE(15)
0018                    INTEGER PRTOPT
                C
                COMMON 22 END
0019                      IF (Q.EQ.0) RETURN
0020                      DO 10 I=1,Q
0021             10       TEMP(I) = XLAMDA(NQ(I))
                C
                C    SOLVE FOR Y IN R(T) * Y = LAMDA
                C
0022                      CALL SOLVRT(TEMP1,TEMP)
                C
                C     UPDATE X
                C
0023                      DO 20 I=1,M
0024             20       X(I) = X(I) - SINNER(QTLINV(1,I),TEMP1(1),Q)
0025                      IF(PRTOPT.GT.2)WRITE(6,21)(X(J),J=1,M)
0026             21       FORMAT(' ',5X,'X AFTER X-CORRECTION',(/,9D14.7))
0027                     NOPS = NOPS + M * Q
0028                       NXCOR = NXCOR + 1
0029                       CALL UPRIN(-500-NXCOR)
                C      INDICATE THAT AN XORRECTION WAS JUST MADE
                C
0030                      RETURN
0031                      END
```

```
FORTRAN IV G LEVEL  21                  REIN                DATE = 80105

0001                  SUBROUTINE REIN (IRC)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                   COMMON 01
0003                     COMMON /C01/PROJV,LENY,QTLZ
0004                     REAL * 8 PROJV(30),LENY,QTLZ(30)
                   COMMON 01 END
                   COMMON 05
0005                     COMMON /C05/C,B,XLAMDA,ICONOK
0006                     REAL*8 C(30,90),B(90),XLAMDA(90)
0007                     INTEGER*2 ICONOK(90)
                   COMMON 05 END
                   COMMON 09
0008                     COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSQKFG,TYPG
                        1,    ISEED,ISOPT,IAOPT,IDOPT
0009                     INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSQKFG,TYPG
                   COMMON 09 END
                   COMMON 12
0010                       COMMON /C12/NQ
0011                       INTEGER NQ(30)
                   COMMON 12 END
                   COMMON 15
0012                       COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
                   COMMON 15 END
                   COMMON 17
                   C
0013                     COMMON/C17/IUNIT5,IUNIT6,IUNIT8
                   COMMON 17 END
                   COMMON 21
0014                     COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                        1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                   COMMON 21 END
                   C
                   C     THIS SR WILL REINVERT, (I.E.) WILL RECOMPUTE THE
                   C      OPERATORS , QTLINV AND R FROM SCRATCH
                   C
                   C       METHODOLOGY:
                   C         QTLINV = LINV
                   C         IF Q = 0 RETURN
                   C         SAVE Q AND NQ INTO COPYQ AND COPYNQ
                   C         ADD THE FIRST CONS TO THE BASIS, UPDATE OPS.
                   C         DO OVER THE REST OF THE CONSTRAINTS
                   C          PROJECT NORMAL
                   C          IF L. INDEP THEN ADD TO THE BASIS & UPDATE
                   C         END DO
                   C         CALCULATE LAMDA
                   C         DO AN XCORRECTION
                   C         CHECK IF ACTIVE CONSTRAINTS ARE SATISFIED
                   C         IF YES, RETURN ELSE SET IRC = CONS. # THAT IS VIOL.
                   C
                   C
                   C        DATE JULY 30 1979
                   C
                   C
0015                      INTEGER COPYNQ(30),COPYQ
0016                        IRC = 0
0017                         IF(NREIN.LT.MXREIN)GOTO 5
```

```
FORTRAN IV G LEVEL  21                  REIN                DATE = 80105

0018                    WRITE(6,6)MXREIN
0019                   IRC = 1
0020          6        FORMAT(' ',5X,'NO. OF REINVERSIONS EXCEED MAX OF ',I5)
0021                    GOTO 110
0022          5         CONTINUE
0023                    NREIN = NREIN + 1
0024                    CALL UPRIN(-600-NREIN)
0025                  CALL INTLQL
              C
0026                    IF(Q.EQ.0) GOTO 100
              C         COPYQ = Q
0027                    COPYQ = Q
0028                    DO 40 I=1,Q
0029          40        COPYNQ(I) = NQ(I)
0030                    Q=0
0031                  CALL RBAR(C(1,COPYNQ(1)),QTLZ,1,M)
0032                   CALL ADD(COPYNQ(1))
0033                   IF(COPYQ.LE.Q) GOTO 100
0034                   DO 50 I=2,COPYQ
0035                   CALL RBAR(C(1,COPYNQ(I)),QTLZ,Q+1,M)
0036                   LENY = SINNER(QTLZ(Q+1),QTLZ(Q+1),M-Q)
0037                   NOPS=NOPS+M-Q
0038                   IF(LENY.LE.TOLDEP)GOTO 50
0039                   CALL RBAR(C(1,COPYNQ(I)),QTLZ,1,Q)
0040                   CALL ADD(COPYNQ(I))
0041          50       CONTINUE
0042          100      CALL UPRIN(-600-NREIN)
0043          110       RETURN
0044                   END
```

```
FORTRAN IV G LEVEL  21                    FEAS                    DATE = 80105

0001                  SUBROUTINE FEAS(IRC,III)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                    COMMON C1
0003                      COMMON /C01/PROJV,LENY,QTLZ
0004                      REAL * 8 PROJV(30),LENY,QTLZ(30)
                    COMMON 01 END
                    COMMON 02
0005                      COMMON /C02/LG
0006                      REAL*8 LG(30,31),L(30,30),G(30,30)
0007                      EQUIVALENCE(LG(1,1),L(1,1))
0008                      EQUIVALENCE(LG(1,2),G(1,1))
                    COMMON 02 END
                    COMMON 03
0009                      COMMON /C03/QTLINV
0010                      REAL*8  QTLINV(30,30)
                    COMMON 03 END
                    COMMON 04
0011                      COMMON /C04/Q1R
0012                      REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0013                      EQUIVALENCE(Q1R(1,1),Q1(1,1))
0014                      EQUIVALENCE(Q1R(1,3),R(1,1))
                    COMMON 04 END
                    COMMON 05
0015                      COMMON /C05/C,B,XLAMDA,ICONOK
0016                      REAL*8 C(30,90),B(90),XLAMDA(90)
0017                      INTEGER*2 ICONOK(90)
                    COMMON 05 END
                    COMMON 06
0018                      COMMON /C06/FF,FNQ,FQ
0019                      INTEGER FNQ(30),FQ
0020                      REAL*8 FF
                    COMMON 06 END
                    COMMON 07
0021                       COMMON /C07/   FQTLIN
0022                       REAL*8 FQTLIN(30,30)
                    COMMON 07 END
                    COMMON 08
0023                       COMMON /C08/   FXLINR
0024                       REAL*8 FXLINR(30,32),FX(30),LINV(30,30),FR(30,30)
0025                       EQUIVALENCE(FXLINR(1,1),FX(1))
0026                       EQUIVALENCE(FXLINR(1,2),LINV(1,1))
0027                       EQUIVALENCE(FXLINR(1,3),FR(1,1))
                    COMMON 08 END
                    COMMON 09
0028                      COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                         1,    ISEED,ISOPT,IAOPT,IDOPT
0029                       INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    COMMON 09 END
                    COMMON 10
0030                        COMMON /C10/UNCONX,UNCONF
0031                        REAL*8 UNCONX(30),UNCONF
                    COMMON 10 END
                    COMMON 11
0032                        COMMON /C11/X
0033                        REAL*8 X(30)
                    COMMON 11 END
```

```
FORTRAN IV G LEVEL  21                FEAS                DATE = 80105

            COMMON 12
0034                  COMMON /C12/NQ
0035                  INTEGER NQ(30)
            COMMON 12 END
            COMMON 13
0036                  COMMON /C13/LM,IM
0037                  REAL*8 LM(30),IM(30)
            COMMON 13 END
            COMMON 14
0038                  COMMON /C14/F,GRAD
0039                  REAL*8 F,GRAD(30)
            COMMON 14 END
            COMMON 15
0040                  COMMON /C15/TOLDCN,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
            COMMON 15 END
            COMMON 16
0041                COMMON/C16/HIX,LOX,HIXLOX,HIG,LOG,HIGLOG,HILM,HIC,LOC,HICLOC,HIB
0042                REAL*8 HIX,LOX,HIXLOX,HIG,LOG,HIGLOG,HILM,HIC,LOC,HICLOC,HIB
            COMMON 16 END
            COMMON 17
0043                COMMON/C17/IUNIT5,IUNIT6,IUNIT8
            COMMON 17 END
            COMMON 19
0044                COMMON /C19/IPRIN,IPTR,IPMAX,NBRK
0045                INTEGER*2 IPRIN(5000),IPTR,IPMAX,NBRK(15)
            COMMON 19 END
            COMMON 21
0046                COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                  1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
            COMMON 21 END
            COMMON 22
0047                COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0048                REAL*8 TITLE(15)
0049                INTEGER PRTOPT
            COMMON 22 END
            C
            C     THIS SR WILL FIND A FEASIBLE POINT, IF ANY.
            C       ELSE, IT WILL PRINT OUT AND SET IRC = 1
            C
            C      METHODOLOGY:
            C      DEPENDING UPON THE START OPTION, SET STARTING PT. TO
            C       BE UNCONS. MIN OR RANDOM PT.
            C
            C       SET QTLINV = LINV
            C
            C       CALL LAMDA
            C
            C,      ADD ALL THE CONS. X IS CURRENTLY ON
            C
            C       FOLLOW ROSEN START METHOD, I.E
            C       A) ADD A VIOLATED CONS. IF LIN. INDEP.
            C       B) IF L. DEP. THEN FIND A CONS TO DROP BY CALCULATING
            C          INFEASIBILITY MULTIPLIERS.
            C          IF ALL IM -VE THEN NO FEAS PT. , ELSE FIND A =VE LM
            C          CONSTRAINT AND DROP IT AND CONTINUE (A)
            C       C) UPDATE X = X - NG(*)(T) * LAMDA
```

```
C          GOTO A
C
C       DATE JULY 30 1979
C
C
0050                GOTO(5,6),ISOPT
C     HERE IF WE HAVE UNCONS. MIN. TO START
C
C
0051   5          DO 10 I=1,M
0052   10         X(I)=UNCONX(I)
0053               GOTO 30
C
C     RANDOM START
C
0054   6          DO 20 I=1,M
0055   20         X(I) = RAND(III)*H1XLOX + LOX
C
C       COPY LINV TO QTLINV
C
0056   30         CALL INTLQL
0057              Q=0
0058              CALL LAMDA
0059              DO 70 I=1,K
0060              IF(DABS(XLAMDA(I)).GT.TOLAM)GOTO 70
C
C    WE HAVE A CONSTRAINT X IS ON
C
0061             CALL RBAR(C(1,I),QTLZ,1,M)
0062             CALL PROJ
0063              IF(LENY.LE.TOLDEP)GOTO 70
0064              CALL ADD(I)
0065   70         CONTINUE
C
C
0066                GOTO 90
C
C
0067   80         CALL RENCHK(IRC)
0068                IF(IRC.EQ.0) GOTO 90
0069                IRC=1
C       NO FEAS PT. FOUND BECAUSE OF REINVERSION WONT HELP
C
0070                GOTO 1000
C
C
C       EVERYTHIN OK
C
C
0071   90          CONTINUE
0072   4              CALL MXMIN(XLAMDA,K,IAOPT ,-1,TOLAM,ICONAD,VALUE)
0073                  IF(ICONAD.EQ.0)GOTO 130
0074   95             CALL RBAR(C(1,ICONAD),QTLZ,Q+1,M)
0075                 IF(Q.EQ.M)GOTO 110
0076                 IF(Q.EQ.0)GOTO 98
0077                 LENY = SINNER(QTLZ(Q+1),QTLZ(Q+1),M-Q)
```

```
0078                    NOPS=NOPS+M-Q
0079                    IF(LENY.LE.TOLDEP)GOTO 110
0080          97        CALL RBAR(C(1,ICONAD),QTLZ,1,Q)
0081          98        NSTEPS = NSTEPS + 1
0082                    CALL UPRIN(NSTEPS+1000)
              C       CALL RBAR(C(1,ICONAD),QTLZ,1,Q)
0083                    CALL ADD(ICONAD)
0084                  VAL = XLAMDA(ICONAD)/R(Q,Q)
0085                    DO 100 I=1,M
0086          100       X(I) = X(I) - QTLINV(Q,I) * VAL
0087                  NOPS = NOPS + M + 1
0088                  IF(PRTOPT.GT.1)WRITE(6,1202)(X(I),I=1,M)
0089          1202    FORMAT(' ',' VECTOR X AFTER STEP IS',(/9D14.7))
0090                    GOTO 80
              C
              C       NOW WE MUST DROP A CONSTRAINT
              C
0091          110       CALL RBAR(C(1,ICONAD),QTLZ,1,Q)
0092                    CALL SOLVR(IM,QTLZ,1,Q)
0093          14        CALL MXMIN(IM,Q,IDOPT ,+1,TOLMUL,ICONDR,VALUE)
0094                    IF(ICONDR.EQ.0)GOTO 120
              C     THERE IS SOME MULTIPLIER THAT IS +VE
              C
0095                    CALL DROP(ICONDR)
0096                     CALL RBAR(C(1,ICONAD),QTLZ,ICONDR,M)
0097                     LENY=SINNER(QTLZ(Q+1),QTLZ(Q+1),M-Q)
0098                     NOPS=NOPS+M-Q
0099                     IF(LENY.LT.TOLDEP)GOTO 110
0100                    GOTO 98
0101          120       WRITE(6,125)
0102          125       FORMAT(' ',5X,'NO FEASIBLE PT. FOUND - ALL IM < 0.0')
0103                  WRITE(6,126)(IM(J),J=1,Q)
0104          126     FORMAT(' ',5X,'INFEASIBILITY MULTIPLIERS ARE',(/9D14.7))
0105                    IRC = 1
0106                    GOTO 1000
              C
              C     FEAS PT IS FOUND & HENCE STORE EVERYTHING
              C
0107          130       DO 140 I = 1,M
0108                    FX(I) = X(I)
0109                    DO 140 J= 1,M
0110          140       FQTLIN(J,I) = QTLINV(J,I)
0111                    FQ = Q
0112                    IF(Q.EQ.0) GOTO 160
0113                    DO 150 I=1,Q
0114                    FNQ(I) = NQ(I)
0115                    DO 150 J=1,I
0116          150       FR(J,I) = R(J,I)
0117          160       IRC = 0
0118          1000      CONTINUE
0119                  WRITE(IUNIT6,71)Q,(NQ(I),I=1,Q)
0120          71      FORMAT(' ',I7,' - IN BASIS ',30I3)
0121                    RETURN
0122                    END
```

```
FORTRAN IV G LEVEL  21                  STPLEN              DATE = 80105

0001                        SUBROUTINE STPLEN(TAU,ICONAD)
0002                    IMPLICIT REAL*8(A-H,O-Z)
              COMMON C1
0003                  COMMON /C01/PROJV,LENY,QTLZ
0004                  REAL * 8 PROJV(30),LENY,QTLZ(30)
              COMMON C1 END
              COMMON 05
0005                  COMMON /C05/C,B,XLAMDA,ICONOK
0006                  REAL*8 C(30,90),B(90),XLAMDA(90)
0007                  INTEGER*2 ICONOK(90)
              COMMON 05 END
              COMMON 09
0008                  COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINOPR,FSOKFG,TYPG
                     1,    ISEED,ISOPT,IAOPT,IDOPT
0009                  INTEGER M,P,Q,K,ON,OFF,ISIZ,KINOPR,FSOKFG,TYPG
              COMMON 09 END
              COMMON 12
0010                    COMMON /C12/NQ
0011                    INTEGER NQ(30)
              COMMON 12 END
              COMMON 15
0012                    COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMIN
              COMMON 15 END
              COMMON 21
0013                  COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                     1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
              COMMON 21 END
              COMMON 22
0014                  COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0015                  REAL*8 TITLE(15)
0016                  INTEGER PRTOPT
              COMMON 22 END
0017                  TAU = 1.D70
0018                  ICONAD = 0
0019                  DO 20 I=1,K
0020                  IF(ICONOK(I).EQ.0)GOTO 20
0021                  IF(Q.EQ.0)GOTO 15
              C   CHECK IF IT IS IN THE BASIS
0022                  DO 10 J = 1,Q
0023                  IF(NQ(J).EQ.I)GOTO 20
0024           10     CONTINUE
0025           15     DENOM = SINNER(C(1,I),PROJV,M)
0026                     NOPS = NOPS + M
0027                  IF(DENOM.LE.TOLDEP)GOTO 20
0028                  RAT = XLAMDA(I)/DENOM
0029                  IF(XLAMDA(I).GE.-TOLAM.AND.XLAMDA(I).LT.0.D0)RAT=0.D0
0030                     NOPS = NOPS + 1
0031                  IF(RAT.GE.TAU)GOTO 20
0032                  TAU = RAT
0033                  ICONAD = I
0034           20     CONTINUE
              C
0035                    IF(PRTOPT.GT.1)WRITE(6,125)TAU,ICONAD
0036           125      FORMAT(' ',5X,'TAU =',E14.9,'   ICONAD = ',I5)
0037                     RETURN
0038                     END
```

```
FORTRAN IV G LEVEL  21                  FLET                    DATE = 80105

0001                SUBROUTINE FLET(IPALG,IRC,IPDPRI)
0002                   IMPLICIT REAL*8(A-H,O-Z)
                COMMON 01
0003                COMMON /C01/PROJV,LENY,QTLZ
0004                REAL * 8 PROJV(30),LENY,QTLZ(30)
                COMMON 01 END
                COMMON 04
0005                 COMMON /C04/Q1R
0006                 REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0007                 EQUIVALENCE(Q1R(1,1),Q1(1,1))
0008                 EQUIVALENCE(Q1R(1,3),R(1,1))
                COMMON 04 END
                COMMON 05
0009                 COMMON /C05/C,B,XLAMDA,ICONOK
0010                 REAL*8 C(30,90),B(90),XLAMDA(90)
0011                 INTEGER*2 ICONOK(90)
                COMMON 05 END
                COMMON 09
0012                COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                   1,    ISEED,ISOPT,IAOPT,IDOPT
0013                INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                COMMON 09 END
                COMMON 11
0014                  COMMON /C11/X
0015                  REAL*8 X(30)
                COMMON 11 END
                COMMON 13
0016                 COMMON /C13/LM,IM
0017                 REAL*8 LM(30),IM(30)
                COMMON 13 END
                COMMON 14
0018                    COMMON /C14/F,GRAD
0019                    REAL*8 F,GRAD(30)
                COMMON 14 END
                COMMON 15
0020                    COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
                COMMON 15 END
                COMMON 17
0021                COMMON/C17 /IUNIT5,IUNIT6,IUNIT8
                COMMON 17 END
                COMMON 21
0022                COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                  1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                COMMON 21 END
                COMMON 22
0023                 COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0024                 REAL*8 TITLE(15)
0025                 INTEGER PRTOPT
                COMMON 22 END
0026                 IRC = 0
0027                   CALL LAMDA
0028           7     CALL GRA
0029           9      CALL RBAR(GRAD,QTLZ,Q+1,M)
0030                  CALL PROJ
0031                 IF(LENY.LE.TOLPR )GOTO 70
0032                      CALL STPLEN(TAU,ICONAD)
```

```
0033                    IF(ICONAD.EQ.0)TAU = 1.D0
                  C   ISTEP = 1 - FULL STEP, 0-PARTIAL STEP
                  C
0034                  ISTEP = 0
0035                  IF(TAU.GT.1.0)TAU = 1.D0
0036                    IF(TAU .EQ.1.D0) ISTEP = 1
0037                   IF(PRTOPT.GT.1)WRITE(6,121)TAU
0038           121       FORMAT(' ',5X,'STEP LENGTH,TAU=',F14.7)
0039                  DO 30 I=1,M
0040           30     X(I) = X(I) - TAU * PROJV(I)
0041                   NOPS=NOPS+M
0042                    IF(PRTOPT.GT.1)WRITE(6,1202)(X(I),I=1,M)
0043           1202     FORMAT(' ',' VECTOR X AFTER STEP IS',(/9D14.7))
0044                  NSTEPS = NSTEPS + 1
0045                    IF(NSTEPS.LT.MXSTPS)GOTO 35
0046                    WRITE(6,31)MXSTPS
0047           31       FORMAT(' ',5X,'NO. OF STEPS EXCEED MAX OF',I5)
0048                    IRC = 1
0049                    GOTO 100
0050           35       CONTINUE
0051                  IF(ISTEP.EQ.1)GOTO 40
0052                  NPSTEP = NPSTEP + 1
0053                    CALL UPRIN(-1000-NSTEPS)
0054                   CALL RBAR(C(1,ICONAD),QTLZ,1,M)
0055                  CALL ADD(ICONAD)
0056           5     CALL RENCHK(IRC)
0057                 IF(IRC.EQ.0) GOTO 7
0058                  WRITE(IUNIT6,8) IRC
0059           8      FORMAT(' ','**** REINVERSION FAILED, IRC = ',I5)
0060                  GOTO 100
0061           40     NFSTEP = NFSTEP + 1
0062                    CALL UPRIN(1000+NSTEPS)
0063                  CALL GRA
0064           70       CONTINUE
0065                  CALL RBAR(GRAD,QTLZ,1,Q)
0066                  CALL SOLVR(LM,QTLZ,1,Q)
0067           4      CALL MXMIN(LM,Q,IDOPT ,-1,TOLMUL,ICONDR,VALUE)
0068                  IF(ICONDR.EQ.0)GOTO 100
0069                  IF(IPALG.EQ.1)GOTO 80
0070           50      CALL STPEDG(ICONDR,DIAG)
0071                  IF(ICONDR.EQ.0)GOTO 100
0072           80     CONTINUE
0073                    CALL DROP(ICONDR)
0074                   CALL LAMDA
0075                  GOTO 9
0076           100     CALL LAMDA
0077                   CALL GRA
0078                   RETURN
0079                  END
```

```
FORTRAN IV G LEVEL  21                NSGNST                DATE = 80105

0001                    SUBROUTINE NSGNST(I,DIAG)
0002                     IMPLICIT REAL*8(A-H,O-Z)
0003                    REAL*8 TEMP(30)
                  COMMON C4
0004                    COMMON /C04/Q1R
0005                    REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0006                    EQUIVALENCE(Q1R(1,1),Q1(1,1))
0007                    EQUIVALENCE(Q1R(1,3),R(1,1))
                  COMMON C9
0008                   COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                     1,    ISEED,ISOPT,IAOPT,IDOPT
0009                    INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON C9 END
                  COMMON 21
0010                   COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                     1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  COMMON 22
0011                    COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0012                    REAL*8 TITLE(15)
0013                    INTEGER PRTOPT
                  COMMON 22 END
0014                    TEMP(I) =1.D0/R(I,I)
0015                    IP1 = I+1
0016                    IF(I.EQ.Q)GOTO 65
0017                    DO 60 J=IP1,Q
0018                        NOPS = NOPS + J - I + 1
0019          60        TEMP(J) =-SINNER(R(I,J),TEMP(I),J-I)/R(J,J)
0020          65        DIAG    = SINNER(TEMP(I),TEMP(I),Q-I+1)
0021                        NOPS = NOPS + Q - I + 2
0022                       IF(PRTOPT.GT.1)WRITE(6,67)I,DIAG
0023          67           FORMAT(' ',5X,'NSGNST - I,DIAG',I5,D20.10)
0024                      RETURN
0025                      END
```

```
FORTRAN IV G LEVEL  21                  STPEDG              DATE = 80105

0001                    SUBROUTINE STPEDG(ICONDR,DIAGMX)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                C
                C    THIS SR CAN BE USED TO FIND MAX LM(I)**2/STPEDG
                C     FOR STEEPEST EDGE
                C   I.E. FIND CONSTRAINT TO DROP FOR STEEPEST EDGE
                C    ICONDR = 0 NONE AVAILABLE TO DROP
                C    ELSE, DROP ICONDR CONSTRAINT
                C
                C   METHOD:
                C        CHOOSE AMONG -VE LM
                C     FOR THE CORRESPONDING CONS. FIND CORRESPONDING ROW IN
                C      R(-1)
                C     FIND INNER PRODUCT OF THAT  TO GET STPEDG
                C     NOW FIND THE MAX OF LM(I)**2/STPEDG
                COMMON C9
0003                 COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    1,     ISEED,ISOPT,IAOPT,IDOPT
0004                 INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                COMMON C9 END
                COMMON 13
0005                  COMMON /C13/LM,IM
0006                  REAL*8 LM(30),IM(30)
                COMMON 13 END
                COMMON 15
0007                     COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
                COMMON 15 END
                COMMON 21
0008                COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                   1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                COMMON 21 END
                COMMON 22
0009                 COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0010                 REAL*8 TITLE(15)
0011                 INTEGER PRTOPT
                COMMON 22 END
0012          50     RATMAX =-1.D70
0013                 ICONDR = 0
0014                  DIAGMX = 0.D0
0015                  IF(Q.EQ.0)GOTO 1000
0016                 DO 70 I=1,Q
0017                 IF(LM(I).GT.TOLMUL)GOTO 70
0018                 CALL NSGNST(I,DIAG)
0019                 RAT = LM(I)*LM(I)/DIAG
0020                   NOPS = NOPS + 2
0021                 IF(RAT.LT.RATMAX)GOTO 70
0022                 RATMAX = RAT
0023                 DIAGMX = DIAG
0024                 ICONDR = I
0025           70    CONTINUE
0026                     IF(PRTOPT.GT.1)WRITE(6,67)ICONDR,DIAGMX
0027           67        FORMAT(' ',5X,'STPEDG - ICONDR,DIAGMX',I5,D20.10)
0028         1000     RETURN
0029                 END
```

```
FORTRAN IV G LEVEL  21                    DUAL                    DATE = 80105

0001                  SUBROUTINE DUAL(IRC)
0002                     IMPLICIT REAL*8(A-H,O-Z)
0003                    REAL*8 UPF
                    COMMON C1
0004                      COMMON /C01/PROJV,LENY,QTLZ
0005                      REAL * 8 PROJV(30),LENY,QTLZ(30)
                    COMMON 01 END
                    COMMON 05
0006                       COMMON /C05/C,B,XLAMDA,ICONOK
0007                       REAL*8 C(30,90),B(90),XLAMDA(90)
0008                       INTEGER*2 ICONOK(90)
                    COMMON 05 END
                    COMMON 08
0009                        COMMON /C08/   FXLINR
0010                        REAL*8 FXLINR(30,32),FX(30),LINV(30,30),FR(30,30)
0011                        EQUIVALENCE(FXLINR(1,1),FX(1))
0012                        EQUIVALENCE(FXLINR(1,2),LINV(1,1))
0013                        EQUIVALENCE(FXLINR(1,3),FR(1,1))
                    COMMON 08 END
                    COMMON 09
0014                      COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                         1,    ISEED,ISOPT,IAOPT,IDOPT
0015                       INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    COMMON C9 END
                    COMMON 10
0016                         COMMON /C10  /UNCONX,UNCONF
0017                         REAL*8 UNCONX(30),UNCONF
                    COMMON 10 END
                    COMMON 11
0018                         COMMON /C11/X
0019                         REAL*8 X(30)
                    COMMON 11 END
                    COMMON 13
0020                        COMMON /C13/LM,IM
0021                        REAL*8 LM(30),IM(30)
                    COMMON 13 END
                    COMMON 14
0022                           COMMON /C14/F,GRAD
0023                           REAL*8 F,GRAD(30)
                    COMMON 14 END
                    COMMON 15
0024                           COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
                    COMMON 15 END
                    COMMON 17
0025                      COMMON/C17 /IUNIT5,IUNIT6,IUNIT8
                    COMMON 17 END
                    COMMON 21
0026                      COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                         1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                    COMMON 21 END
                    COMMON 22
0027                       COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0028                       REAL*8 TITLE(15)
0029                       INTEGER PRTOPT
                    COMMON 22 END
                    C   THIS SR IS FOR DUAL ALGORITHM.
```

```
C   IRC IS THE RETURN CODE - 0 -EVERYTHING IS OK
C     1 MEANS COULD NOT SOLVE THE DUAL PROBLEM OR NO FEASIBLE
C     POINT EXISTS.
C   METHODOLOGY:
C    1) FIND A STARTING OPTIMAL POINT (UNCONS MIN)
C    2)FIND A VIOLATED CONS TO ADD( IAOPT)
C    3) CHECK IF THE CONS IS LIN. DEP.
C    4) IF IT IS L.DEP. FIND A CONS TO DROP
C    5) GO TO 3
C    6) IF IT IS L.IND. FIND STEP LENGTH- FULL AND PARTIAL
C    7) TAKE STEP
C    8) IF IT IS FULL STEP, ADD THE CONS. AND GOTO 2)
C    9) IF PARTIAL, DROP THE CONS.  WHICH FORCED A PARTIAL STEP
C   10) GOTO 6.
C
C   DATE AUGUST 2,1979
C
0030            IRC = 0
C
C    COPY THE UNCONS. MIN
C
0031              F = UNCONF
0032              DO 10 I=1,M
0033       10     X(I) = UNCONX(I)
0034              CALL INILQL
C   SET Q TO 0
0035                Q = 0
0036       40     CALL RENCHK(IRC)
0037              IF(IRC.NE.0)GOTO 1000
0038       45     CALL MXMIN(XLAMDA,K,IAOPT ,-1,TOLAM,ICONAD,VALUE)
0039              IF(Q.NE.M)LM(Q+1)=0.D0
0040       50     CONTINUE
0041              IF(ICONAD.EQ.0)GOTO 900
0042       60     CALL RBAR(C(1,ICONAD),QTLZ,1,M)
0043              CALL PROJ
0044              IF(Q.EQ.0)GOTO 110
C             CALL MULT(GRAD,LM,1)
C             CALL MULT(C(1,ICONAD),IM,1)
0045              CALL SOLVR(IM,QTLZ,1,Q)
0046              IF(LENY.GT.TOLDEP)GOTO 90
0047               CALL DSTPLN(T1,ICONDR)
0048              IF(ICONDR.NE.0)GOTO 80
0049              WRITE(IUNIT6,71)
0050       71     FORMAT(' ','**** NO FEASIBLE POINT EXISTS ****')
0051              IRC = 1
0052              GOTO 1000
0053       80     CONTINUE
0054                   CALL DROP(ICONDR)
0055                   DO 85 I=1,Q
0056       85          LM(I) = LM(I) - T1 * IM(I)
0057                   LM(Q+1) = T1
0058                 NOPS = NOPS + Q
0059              GOTO 60
0060       90     IF(Q.EQ.0)GOTO 110
0061              CALL DSTPLN(T1,ICONDR)
0062       110     T3 = SINNER(QTLZ(Q+1),QTLZ(Q+1),M-Q)
```

```
0063                T2 = -XLAMDA(ICONAD)/T3
0064                   NOPS = NOPS + M - Q+ 1
0065               T=T2
0066               ISTEP = 1
0067               IF(Q.EQ.0.OR.ICONDR.EQ.0)GOTO 115
0068               IF(T1.GE.T)GOTO 115
0069               T=T1
0070               ISTEP = 0
                C   ISTEP = 1 FULL STEP, ADD
                C   ISTEP = 0 PARTIAL STEP, DROP
                C
0071            115    CONTINUE
0072                   DO 120 I=1,M
0073            120    X(I) = X(I) + T * PROJV(I)
0074                   F = F + T * T3 *(LM(Q+1) + 0.5 * T)
0075                    IF(PRTOPT.GT.1)WRITE(6,701)T1,T2,T
0076            701     FORMAT(' ',5X,'T1,T2,T=',3E16.9)
0077                  IF(PRTOPT.GT.1)  WRITE(6,702)(X(I),I=1,M)
0078            702     FORMAT(' ',5X,'X VECTOR',(/9D14.7))
0079                     IF(Q.EQ.0)GOTO 705
0080                   DO 703 I=1,Q
0081            703    LM(I)=LM(I) - IM(I) * T
0082                   IF(PRTOPT.GT.2)WRITE(6,704)(LM(I),I=1,Q)
0083            704    FORMAT(' ',5X,'PREDICTED LM',(/9D14.7))
0084            705    CONTINUE
0085                   LM(Q+1) = LM(Q+1) + T
0086                    NOPS = NOPS + M + Q +3
0087                  NSTEPS = NSTEPS + 1
0088                   IF(NSTEPS.LT.MXSTPS)GOTO 135
0089                   WRITE(6,131)MXSTPS
0090            131    FORMAT(' ',5X,'NO. OF STEPS EXCEED MAX OF',I5)
0091                   IRC = 1
0092                   GOTO 1000
0093            135    CONTINUE
0094                  IF(ISTEP.EQ.0)GOTO 130
0095                  CALL UPRIN(1000+NSTEPS)
0096                  CALL ADD(ICONAD)
0097                  NFSTEP = NFSTEP + 1
0098                  GOTO 40
0099            130    CALL UPRIN(-1000-NSTEPS)
0100                  CALL DROP(ICONDR)
0101                   LM(Q+1) = LM(Q+2)
0102                     XLAMDA(ICONAD) = SINNER(C(1,ICONAD),X,M) - B(ICONAD)
0103                     NOPS = NOPS + M
0104                  NPSTEP = NPSTEP + 1
0105                  GOTO 60
0106            900    UPF = F
0107                   CALL GRA
0108                   NOPS = NOPS - 2*M*(M+2)-1
0109                   IF(DABS(UPF-F).GT.TOLAM)WRITE(6,909)F,UPF
0110            909    FORMAT(' ',5X,'F , UPF ARE DIFFERENT',2F16.7)
0111            1000    RETURN
0112                    END
```

```
FORTRAN IV G LEVEL  21                  DSTPLN              DATE = 80105

0001                    SUBROUTINE DSTPLN(T1,ICONDR)
0002                      IMPLICIT REAL*8(A-H,O-Z)
            COMMON 09
0003                    COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                       1,    ISEED,ISOPT,IAOPT,IDOPT
0004                    INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
            COMMON 09 END
            COMMON 13
0005                     COMMON /C13/LM,IM
0006                     REAL*8 LM(30),IM(30)
            COMMON 13 END
            COMMON 15
0007                       COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
            COMMON 15 END
            COMMON 21
0008                    COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                       1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
            COMMON 21 END
            COMMON 22
0009                    COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0010                    REAL*8 TITLE(15)
0011                    INTEGER PRTOPT
            COMMON 22 END
0012                     ICONDR = 0
0013                     T1 = 1.D70
0014                     IF(Q.EQ.0)       RETURN
0015                     DO 70 I=1,Q
0016                     IF(IM(I).LE.TOLMUL)GOTO 70
0017                        IF(LM(I).LT.0.D0)LM(I)=0.D0
0018                     RAT = LM(I)/IM(I)
0019                       NOPS = NOPS + 1
0020                     IF(RAT.GT.T1)GOTO 70
0021                     T1 = RAT
0022                     ICONDR = I
0023          70         CONTINUE
0024                       IF(PRTOPT.GT.1)WRITE(6,71)T1,ICONDR
0025          71         FORMAT(' ',5X,'DSTPLN - T1,ICONDR',E20.10,I9)
0026                     RETURN
0027                     END

FORTRAN IV G LEVEL  21                  UPRIN               DATE = 80105

0001                     SUBROUTINE UPRIN(ICON)
0002                       IMPLICIT REAL*8(A-H,O-Z)
            COMMON 19
0003                    COMMON /C19/IPRIN,IPTR,IPMAX,NBRK
0004                    INTEGER*2 IPRIN(5000),IPTR,IPMAX,NBRK(15)
            COMMON 19 END
0005                    IF(IPTR.GE.IPMAX)GOTO 10
0006                    IPTR=IPTR+1
0007                    IPRIN(IPTR)=ICON
0008                    RETURN
0009          10        WRITE(6,11)IPMAX
0010          11        FORMAT(' ',5X,'PTR HAS REACHED MAX OF ',I5)
0011                    RETURN
0012                    END
```

```
FORTRAN IV G LEVEL  21                  PRIMAL                  DATE = 80105

0001                    SUBROUTINE PRIMAL(IALG,IRC,IPDPRI)
0002                      IMPLICIT REAL*8(A-H,O-Z)
                  COMMON 03
0003                      COMMON /C03   /QTLINV
0004                      REAL*8  QTLINV(30,30)
                  COMMON 03 END
                  COMMON 04
0005                      COMMON /C04/Q1R
0006                      REAL*8 Q1R(30,32),Q1(30,30),R(30,30)
0007                      EQUIVALENCE(Q1R(1,1),Q1(1,1))
0008                      EQUIVALENCE(Q1R(1,3),R(1,1))
                  COMMON 04 END
                  COMMON 05
0009                      COMMON /C05/C,B,XLAMDA,ICONOK
0010                      REAL*8 C(30,90),B(90),XLAMDA(90)
0011                      INTEGER*2 ICONOK(90)
                  COMMON 05 END
                  COMMON 06
0012                      COMMON /C06/FF,FNQ,FQ
0013                      INTEGER FNQ(30),FQ
0014                      REAL*8 FF
                  COMMON 06 END
                  COMMON 07
0015                       COMMON /C07   /FQTLIN
0016                       REAL*8 FQTLIN(30,30)
                  COMMON 07 END
                  COMMON 08
0017                       COMMON /C08   /FXLINR
0018                       REAL*8 FXLINR(30,32),FX(30),LINV(30,30),FR(30,30)
0019                       EQUIVALENCE(FXLINR(1,1),FX(1))
0020                       EQUIVALENCE(FXLINR(1,2),LINV(1,1))
0021                       EQUIVALENCE(FXLINR(1,3),FR(1,1))
                  COMMON 08 END
                  COMMON 09
0022                     COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                        1,    ISEED,ISOPT,IAOPT,IDOPT
0023                      INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON 09 END
                  COMMON 11
0024                        COMMON /C11/X
0025                        REAL*8 X(30)
                  COMMON 11 END
                  COMMON 12
0026                        COMMON /C12/NQ
0027                        INTEGER NQ(30)
                  COMMON 12 END
                  C   THIS SR IS FOR PRIMAL AND PRIMAL PART OF PRIMAL DUAL
                  C   THE FUNCTION OF THIS SR IS TO INVOKE APPROPRIATE PRIMAL
                  C   ROUTINE AND SET UP NECESSARY STUFF FOR PRIMAL DUAL
                  C
                  C    IALG IS THE PRIMAL ALGORITHM TO BE RUN
                  C   1 - FLETCHER, 2 - FLETCHER WITH STEEPEST EDGE
                  C   3 - GOLDFARB, 4 - GOLDFARB WITH STEEPEST EDGE
                  C   5 - GOLDFARB S.T. IT WONT VIOLATE ANY CONS. ALREDY DROPPED
                  C   6 - GOLDFARB  COMBO OF 4 & 5
                  C
```

```
      C     IRC IS THE RETURN CODE , 0 - EVERYTHING OK,1 -OTHERWISE
      C
      C    IPDPRI IS 0 - PRIMAL, 1 - PRIMAL DUAL
      C
0026            IRC = 0
0029            IF(IPDPRI.EQ.1)GOTO 40
0030            DO 10 I=1,K
0031       10   ICONOK(I) = 1
0032            DO 20 I = 1,M
0033            X(I) = FX(I)
0034            DO 20 J=1,M
0035       20   QTLINV(J,I) = FQTLIN(J,I)
0036            F = FF
0037            Q = FQ
0038            IF(Q.EQ.0)GOTO 70
0039            DO 30 I=1,Q
0040            NQ(I)=FNQ(I)
0041            DO 30 J=1,I
0042            R(J,I) = FR(J,I)
0043       30   CONTINUE
0044            GOTO 70
0045       40      CALL UPRIN(2000)
0046            DO 50 I=1,K
0047       50   ICONOK(I) = 0
0048            IF(Q.EQ.0)GOTO 70
0049            DO 60 I=1,Q
0050       60   ICONOK(NQ(I)) = 1
0051       70   GOTO(1,1,3,3,3,3),IALG
0052       1    CALL FLET(IALG,IRC,IPDPRI)
0053            GOTO 100
0054       3    IPALG = IALG - 2
0055            CALL GOLD(IPALG,IRC,IPDPRI)
0056       100     IF(IPDPRI.EQ.1)CALL UPRIN(-2000)
0057               RETURN
0058            END
```

```
FORTRAN IV G LEVEL  21                ACTCHK              DATE = 80105

0001                  SUBROUTINE ACTCHK(IRC)
0002                    IMPLICIT REAL*8(A-H,O-Z)
               COMMON 05
0003                  COMMON /C05/C,B,XLAMDA,ICONOK
0004                  REAL*8 C(30,90),B(90),XLAMDA(90)
0005                  INTEGER*2 ICONOK(90)
               COMMON 05 END
               COMMON 09
0006                COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                   1,    ISEED,ISOPT,IAOPT,IDOPT
0007                INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
               COMMON 09 END
               COMMON 15
0008                  COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
               COMMON 15 END
               COMMON 12
0009                  COMMON /C12/NQ
0010                  INTEGER NQ(30)
               COMMON 12 END
               COMMON 22
0011                  COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0012                  REAL*8 TITLE(15)
0013                  INTEGER PRTOPT
               COMMON 22 END
               C     THIS SR WILL CHECK IF ACTIVE CONSTRAINTS ARE SATISFIED
               C     IRC IS THE RETURN CODE , SET TO 0 IF ALL ACTIVE CONS. ARE
               C     SATISFIED AT X, ELSE IT IS SET TO THE CONSTRAINT NUMBER
               C      THAT IS NOT SATISFIED
               C
0014                   IRC =0
0015                   IF(Q.EQ.0)RETURN
0016                   DO 10 I=1,Q
0017                   IF(DABS(XLAMDA(NQ(I))).LE.TOLAM)GOTO 10
0018                   IRC = I
0019                   WRITE(6,11)NQ(I),XLAMDA(NQ(I)),TOLAM
0020            11     FORMAT(' ',5X,'ACT. CONS',I5,' LAMDA - ',F16.7,' TOL',F16.7)
0021                   RETURN
0022            10     CONTINUE
0023                   RETURN
0024                   END
```

```
FORTRAN IV G LEVEL  21                  RENCHK              DATE = 80105

0001                     SUBROUTINE RENCHK(IRC)
0002                    IMPLICIT REAL*8(A-H,O-Z)
            COMMON 17
0003                COMMON/C17/IUNIT5,IUNIT6,IUNIT8
            COMMON 17 END
            C       IRC = 0 RETURN CODE 0 - EVERYTHING OK
            C           >= 0 THEN A CONSTRAINT IS STILL VIOLATED AND REINVERSION
            C         CANNOT HELP
            C
            C     METHOD:
            C       CALL ACTCHK
            C       IF EVERYTHING OK RETURN
            C      ELSE, PERFORM XCOR
            C      CALL LAMDA
            C      CALL ACTCHK
            C      IF EVERYTHING OK RETURN
            C      ELSE, PERFORM REIN
            C      CALL XCOR
            C      CALL ACTCHK
            C      IF EVERYTHING OK, RETURN
            C      ELSE, RETURN, REINVERSION FAILED
            C
0004                  CALL LAMDA
0005                  CALL ACTCHK(IRC)
0006                  IF(IRC.EQ.0)GOTO 45
0007                  CALL XCOR
0008                  CALL LAMDA
0009                  CALL ACTCHK(IRC)
0010                  IF(IRC.EQ.0)GOTO 45
0011                  CALL REIN  (IRC)
0012                  IF(IRC.NE.0)GOTO 45
0013                  CALL XCOR
0014                  CALL LAMDA
0015                  CALL ACTCHK(IRC)
0016                  IF(IRC.EQ.0) GOTO 45
0017                  WRITE(IUNIT6,201)IRC
0018           201    FORMAT(' ',' ****REINVERSION FAILED -IRC = ',I5)
0019          45      RETURN
0020                  END
```

```
FORTRAN IV G LEVEL  21                  PRDUAL              DATE = 80105

0001                        SUBROUTINE PRDUAL(IALGO,IRC,IPDPRI)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                  COMMON C1
0003                    COMMON /C01/PROJV,LENY,QTLZ
0004                    REAL * 8 PROJV(30),LENY,QTLZ(30)
                  COMMON 01 END
                  COMMON 05
0005                     COMMON /C05/C,B,XLAMDA,ICONOK
0006                     REAL*8 C(30,90),B(90),XLAMDA(90)
0007                     INTEGER*2 ICONOK(90)
                  COMMON 05 END
                  COMMON 09
0008                    COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                     1,     ISEED,ISOPT,IAOPT,IDOPT
0009                     INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                  COMMON 09 END
                  COMMON 10
0010                       COMMON /C10/UNCONX,UNCONF
0011                       REAL*8 UNCONX(30),UNCONF
                  COMMON 10 END
                  COMMON 11
0012                       COMMON /C11/X
0013                       REAL*8 X(30)
                  COMMON 11 END
                  COMMON 13
0014                     COMMON /C13/LM,IM
0015                     REAL*8 LM(30),IM(30)
                  COMMON 13 END
                  COMMON 14
0016                        COMMON /C14/F,GRAD
0017                        REAL*8 F,GRAD(30)
                  COMMON 14 END
                  COMMON 15
0018                        COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
                  COMMON 15 END
                  COMMON 17
0019                    COMMON/C17/IUNIT5,IUNIT6,IUNIT8
                  COMMON 17 END
                  COMMON 21
0020                    COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCGR
                     1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                  COMMON 21 END
                  COMMON 22
0021                     COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0022                     REAL*8 TITLE(15)
0023                     INTEGER PRTOPT
                  COMMON 22 END
0024                          IRC = 0
0025                          F = UNCONF
0026                         DO 10 I=1,M
0027           10            X(I) = UNCONX(I)
0028                         CALL INTLQL
0029                         Q = 0
0030           40            CALL LAMDA
0031           45            CALL MXMIN(XLAMDA,K,IAOPT,-1,TOLAM,ICONAD,VALUE)
0032                         IF(ICONAD.EQ.0)GOTO 1000
```

```
0033                    IF(Q.EQ.M)GOTO 105
                C     CALC PROJ. OF NORMAL FOR ICONAD
0034             50     CALL RBAR(C(1,ICONAD),QTLZ,Q+1,M)
0035                    CALL PROJ
0036                    IF(LENY.LE.TOLPR)GOTO 100
                C        CALC FULL DUAL STEP LENGTH
0037                    T3 = SINNER(QTLZ(Q+1),QTLZ(Q+1),M-Q)
0038                    NOPS=NOPS+M-Q
0039                    T2 = -XLAMDA(ICONAD)/T3
0040                  IF(PRTOPT.GT.1)WRITE(6,121)T2
0041             121    FORMAT(' ',5X,'FULL STEP , T2 =',F14.7)
0042                    DO 60 I=1,M
0043             60     X(I)=X(I) + T2 * PROJV(I)
0044                  IF(PRTOPT.GT.1)WRITE(6,1202)(X(I),I=1,M)
0045             1202   FORMAT(' ',' VECTOR X AFTER STEP IS',(/9D14.7))
0046                    NOPS = NOPS + M + 1
0047                    NSTEPS = NSTEPS + 1
0048                    NFSTEP = NFSTEP + 1
0049                   CALL UPRIN(1000+NSTEPS)
0050                  CALL RBAR(C(1,ICONAD),QTLZ,1,Q)
0051                    CALL ADD(ICONAD)
0052                    CALL RENCHK(IRC)
0053                    IF(IRC.NE.0)GOTO 1000
0054                    CALL GRA
                C       CALC LAGRANGE MULTIPLIERS TO SEE IF PRIMAL TO BE CALLED
0055                  CALL RBAR(GRAD,QTLZ,1,Q)
0056                  CALL SOLVR(LM,QTLZ,1,Q)
0057                    CALL MXMIN(LM,Q,IDOPT,-1,TOLMUL,ICONDR,VALUE)
0058                    IF(ICONDR.EQ.0)GOTO 45
0059                    CALL PRIMAL(IALGO,IRC,IPDPRI)
0060                    IF(IRC.EQ.0)GOTO 40
0061                    WRITE(6,70)IRC
0062             70     FORMAT(' ',5X,'*** PRIMAL FAILED - IRC=',I5)
0063                    GOTO 1000
0064             100    CALL GRA
                C    CALC LM & IM - DROP THE ONE CORRESPONDING TO THE MIN RATIO
0065                    CALL RBAR(GRAD,QTLZ,1,Q)
0066                    CALL SOLVR(LM,QTLZ,1,Q)
0067             105    CALL RBAR(C(1,ICONAD),QTLZ,1,Q)
0068                    CALL SOLVR(IM,QTLZ,1,Q)
0069                    CALL DSTPLN(T1,ICONDR)
0070                    IF(ICONDR.EQ.0)GOTO 120
0071                    CALL DROP(ICONDR)
0072                    GOTO 50
0073             120    WRITE(6,130)
0074             130    FORMAT(' ',5X,'NO FEAS PT EXISTS -PD')
0075             1000   CALL GRA
0076                  RETURN
0077                  END
```

```
FORTRAN IV G LEVEL  21                  GOLD                  DATE = 80105

0001                  SUBROUTINE GOLD(JPRIM,IRC,IPDPRI)
0002                     IMPLICIT REAL*8(A-H,O-Z)
                    COMMON 01
0003                      COMMON /C01/PROJV,LENY,QTLZ
0004                      REAL * 8 PROJV(30),LENY,QTLZ(30)
                    COMMON 01 END
                    COMMON 05
0005                      COMMON /C05/C,B,XLAMDA,ICONOK
0006                      REAL*8 C(30,90),B(90),XLAMDA(90)
0007                      INTEGER*2 ICONOK(90)
                    COMMON 05 END
                    COMMON 09
0008                      COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                   1,     ISEED,ISOPT,IAOPT,IDOPT
0009                      INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                    COMMON 09 END
                    COMMON 11
0010                        COMMON /C11/X
0011                        REAL*8 X(30)
                    COMMON 11 END
                    COMMON 12
0012                        COMMON /C12/NQ
0013                        INTEGER NQ(30)
                    COMMON 12 END
                    COMMON 13
0014                      COMMON /C13/LM,IM
0015                      REAL*8 LM(30),IM(30)
                    COMMON 13 END
                    COMMON 14
0016                        COMMON /C14/F,GRAD
0017                        REAL*8 F,GRAD(30)
                    COMMON 14 END
                    COMMON 15
0018                        COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAH,TOLMUL
                    COMMON 15 END
                    COMMON 17
0019                      COMMON/C17 /IUNIT5,IUNIT6,IUNIT8
                    COMMON 17 END
                    COMMON 21
0020                      COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                   1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
                    COMMON 21 END
                    COMMON 22
0021                      COMMON /C22/  TITLE,NPAGE,NLINE,MXLINE,PRTOPT
0022                      REAL*8 TITLE(15)
0023                      INTEGER PRTOPT
                    COMMON 22 END
0024                      REAL*8 B1(30),LM1(30),B2(30),LM2(30)
0025                      INTEGER QM1,NNG(30),NG(30),QP1
0026                1         CALL LAMDA
0027                2     CALL GRA
0028                4     NDROPH = 0
0029                      IF(Q.EQ.0)GOTO 200
0030                      DO 5 I=1,Q
0031                5     NG(I) = 1
0032                        CALL RBAR(GRAD,QTLZ,1,Q)
```

```
0033                    CALL SOLVR(LM,QTLZ,1,Q)
0034          6       CALL MXMIN(LM,Q,IDOPT,-1,TOLMUL,ICONDR,VALUE)
0035                  IF(ICONDR.EQ.0)GOTO 200
0036          15      GOTO(40,10,20,10),JPRIM
0037          10      CALL STPEDG(ICONDR,DIAG)
0038                     IF(ICONDR.EQ.0)GOTO 95
0039          17      GOTO(40,40,20,30),JPRIM
0040          20      CALL NSGNST(ICONDR,DIAG)
0041          30      BEE = LM(ICONDR) / DIAG
0042                     NOPS = NOPS + 1
0043                  B1(NDROPH + 1) = BEE
0044                  IF(NDROPH.EQ.0)GOTO 40
                C
                C   UPDATE BEES
                C
0045                  DO 35 I=1,NDROPH
                C     CALCULATE IM FOR DROPPED CONSTRAINTS
0046                   CALL RBAR(C(1,NNG(I)),QTLZ,ICONDR,Q)
0047                   CALL SOLVR(IM,QTLZ,ICONDR,Q)
                C     FIND N(ICONDR)* N(NNG(I))
0048           35     B1(I) = B1(I) + IM(ICONDR) * BEE
0049                     NOPS = NOPS + NDROPH
0050           40     NDROPH = NDROPH + 1
0051                  B1(NDROPH) =  BEE
0052                  NNG(NDROPH) = NQ(ICONDR)
0053                  QM1 = Q-1
0054                  IF(QM1.EQ.0)GOTO 65
0055                  DO 60 I=1,QM1
0056                  IF(I.LT.ICONDR)GOTO 50
0057                  LM1(I)=LM(I+1)
0058                  NG(I) = NG(I+1)
0059                  GOTO 60
0060           50     LM1(I) = LM(I)
0061           60     CONTINUE
0062           65      CONTINUE
0063                  CALL DROP(ICONDR)
0064                  IF(Q.EQ.0)GOTO 200
0065                  CALL RBAR(GRAD,QTLZ,1,Q)
0066                  CALL SOLVR(LM,QTLZ,1,Q)
0067                  DO 70 I=1,Q
0068                      LM2(I)=LM(I)
0069                  IF(NG(I).EQ.0)GOTO 67
                C     CHECK IF LM IS LESS THAN BEFORE
0070                  IF(LM(I).LT.LM1(I))GOTO 70
0071           63     NG(I)=0
0072           67     LM(I)=1.D70
0073           70     CONTINUE
0074                  CALL MXMIN(LM,Q,IDOPT,-1,TOLMUL,ICONDR,VALUE)
0075                  IF(ICONDR.GT.0)GOTO 15
0076           95     GOTO(200,200,100,100),JPRIM
0077           100    IF(Q.EQ.0) GOTO 200
0078                      DO 103 I=1,Q
0079           103        LM(I)=LM2(I)
0080                      GOTO 110
0081           105     CALL RBAR(GRAD,QTLZ,1,Q)
0082                   CALL SOLVR(LM,QTLZ,1,Q)
```

```
FORTRAN IV G LEVEL  21                 GOLD                DATE = 80105

0083          110    CALL MXMIN(LM,Q,IDOPT,-1,TOLMUL,ICONDR,VALUE)
0084                 IF(ICONDR.EQ.0)GOTO 200
0085          115    CALL NSGNST(ICONDR,DIAG)
0086                 BEE = LM(ICONDR) / DIAG
0087                    NOPS = NOPS + 1
0088                 DO 120 I=1,NDROPH
0089                 CALL RBAR(C(1,NNG(I)),QTLZ,ICONDR,Q)
0090                 CALL SOLVR(IM,QTLZ,ICONDR,Q)
0091                 B2(I) = B1(I)+BEE * IM(ICONDR)
0092                    NOPS = NOPS + 1
0093                 IF(B2(I).LE.TOLMUL)GOTO 120
0094                 IF(PRTOPT.GT.2)WRITE(6,112)ICONDR,NNG(I),IM(ICONDR),B2(I),B1(I)
                    1,BEE
0095          112     FORMAT(' ','CANT DROP',I5,'TH INDEX CONS-WILL VIOL',I5,
                    1'IM,B2,B1,BEE',4D16.7)
              C
              C   THIS WILL VIOLATE ITH CONSTRAINT, THEREFORE DONT DROP THIS ONE
              C
0096                  LM(ICONDR) = 1.D70
0097                  GOTO 110
0098          120     CONTINUE
              C
              C    DROPPING THIS WILL NOT VIOLATE A PREVIOUSLY DROPPED CONS,
              C   HENCE, IT IS OK TO DROP
              C
0099                  DO 130 I=1,NDROPH
0100          130     B1(I)=B2(I)
0101                  NDROPH = NDROPH + 1
0102                  NNG(NDROPH)=NQ(ICONDR)
0103                  CALL DROP(ICONDR)
0104                  IF(Q.GT.0)GOTO 105
0105          200      QP1=J+1
0106                  CALL RBAR(GRAD,QTLZ,QP1,M)
0107                  CALL PROJ
0108                  IF(LENY.LE.TOLPR)GOTO 275
0109                  CALL STPLEN(TAU,ICONAD)
0110                  ISTEP = 1
0111                  IF(TAU.GT.1.D0)TAU=1.D0
0112                  IF(TAU.EQ.1.D0)ISTEP = 0
0113                  IF(PRTOPT.GT.1)WRITE(6,121)TAU
0114          121       FORMAT(' ',5X,'STEP LENGTH,TAU=',F14.7)
0115                  DO 210 I=1,M
0116          210     X(I)=X(I) - TAU * PROJV(I)
0117                   IF(PRTOPT.GT.1)WRITE(6,1202)(X(I),I=1,M)
0118          1202     FORMAT(' ',' VECTOR X AFTER STEP IS',(/9D14.7))
0119                    NOPS = NOPS + M
0120                  NSTEPS=NSTEPS + 1
0121                  IF(NSTEPS.LT.MXSTPS)GOTO 215
0122                  WRITE(IUNIT6,212)MXSTPS
0123          212     FORMAT(' ',5X,'MAX STEPS EXCEEDED',I7)
0124                  IRC = 1
0125                 GOTO 300
0126          215     IF(ISTEP.EQ.1)GOTO 220
0127                  NFSTEP = NFSTEP + 1
0128                    CALL UPRIN(1000+NSTEPS)
0129                    IF(NDROPH.GT.0)GOTO 1
```

```
0130                    CALL GRA
0131          275      CONTINUE
0132                    GOTO 300
0133          220     NPSTEP = NPSTEP + 1
0134                   CALL UPRIN(-1000-NSTEPS)
0135                   CALL RBAR(C(1,ICONAD),GTLZ,1,M)
0136                  CALL ADD(ICONAD)
0137          235     CALL RENCHK(IRC)
0138                  IF(IRC.EQ.0)GOTO 2
0139          300      CONTINUE
0140                   RETURN
0141                   END
```

```
FORTRAN IV G LEVEL  21                  CHECK                DATE = 80105

0001                SUBROUTINE CHECK
0002                   IMPLICIT REAL*8(A-H,O-Z)
              C THIS ROUTINE IS MODIFIED ON 8/9/79 .
              C   MAJOR CHANGES ARE THAT IT RETURNS MFLAG AS A SIGNED 5 DIGIT
              C     AS  S  D1 D2 D3 D4 D5
              C    S  = - IF THE ALG. DID NOT GET THE CORRECT FUNC. VAL. AT OPT
              C       = BLANK OR + OTHERWISE
              C   D1  = 1 ALWAYS
              C   D2  = 0 X IS CORRECT AT OPT. 1 OTHERWISE
              C   D3  = 0, NO. OF CONS. IN FINAL BASIS CORRECT, 1- INCORRECT
              C   D4  = 0, FINAL BASIS IS DIFFERENT FROM CORRECT ONE, 1-INCORRE
              C   D5  = 0, ABS(LAMDA) IS < EPS FOR CORRECT FINAL BASIS,1-OTHERW
0003                   INTEGER*2 LOGIC(90)
              C   LOGIC ARRAY IS TEMPORARY FOR THIS RTN. ONLY
              COMMON 05
0004                   COMMON /C05/C,B,XLAMDA,ICONOK
0005                   REAL*8 C(30,90),B(90),XLAMDA(90)
0006                   INTEGER*2  ICONOK(90)
              COMMON 05 END
              COMMON 09
0007                  COMMON /C09/M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
                     1,    ISEED,ISOPT,IAOPT,IDOPT
0008                   INTEGER M,P,Q,K,ON,OFF,ISIZ,KINDPR,FSOKFG,TYPG
              COMMON 09 END
              COMMON 11
0009                     COMMON /C11/X
0010                     REAL*8 X(30)
              COMMON 11 END
              COMMON 12
0011                     COMMON /C12/NQ
0012                     INTEGER NQ(30)
              COMMON 12 END
              COMMON 14
0013                      COMMON /C14/F,GRAD
0014                      REAL*8 F,GRAD(30)
              COMMON 14 END
              COMMON 15
0015                      COMMON /C15/TOLDCM,TOLDEP,TOLINV,TOLPR,TOLAM,TOLMUL
              COMMON 15 END
              COMMON 17
0016                  COMMON/C17 /IUNIT5,IUNIT6,IUNIT8
              COMMON 17 END
              COMMON 18
0017                   COMMON /C18 /SOLNX,SOLNF,SOLNQ
0018                   REAL*8 SOLNX(30),SOLNF
0019                   INTEGER SOLNQ
              COMMON 18 END
              COMMON 21
0020                  COMMON/C21/IAG,NSTEPS,NADDS,NDROPS,NOPS,NSQRT,NREIN,NXCOR
                     1,NFSTEP,NPSTEP,ICC,NTIME,MXSTPS,MXREIN
              COMMON 21 END
0021                  IMF=0
0022                  NMISS=0
0023                  SUM=0.
0024                  SUMX=0.
0025                   DO 5 I=1,SOLNQ
```

```
FORTRAN IV G LEVEL  21                CHECK              DATE = 80105

0026        5       LOGIC(I)=1
0027                CALL UPRIN( 9000)
0028                MFLAG=10000
0029        10      DO 20 I=1,M
0030                SUM=SUM+(X(I)-SOLNX(I))**2
0031                SUMX=SUMX+SOLNX(I)**2
0032                IF(DABS(X(I)-SOLNX(I)).LE.1.D-2*DABS(SOLNX(I)))  GOTO 20
0033                IMF=1
0034                CALL UPRIN(I )
0035                 ICON = - DABS(X(I)-SOLNX(I)) * 100000
0036                CALL UPRIN( ICON)
0037        20      CONTINUE
0038                IF(IMF.NE.1)GOTO21
0039                CALL UPRIN( 9001)
0040        21      CONTINUE
0041                SUM=DSQRT(SUM)
0042                SUMX=DSQRT(SUMX)
0043                RAT=SUM/SUMX
0044                IRAT=RAT*100000
0045                CALL UPRIN( IRAT)
0046                CALL UPRIN( 9002)
0047                IF(RAT.GT.TOLPR)WRITE(IUNIT6, 30 )SUM,SUMX,RAT
0048        30      FORMAT(' ',5X,'RELATIVE ERROR= ',F10.4,' / ',F10.4,' = ',F10.4)
0049                IF (IMF.EQ.1) GOTO 80
0050        45      IF (Q.EQ.0.AND.SOLNQ.EQ.0) GOTO 249
0051                IF(Q.EQ.0) GOTO 155
0052                DO 48 I = 1,Q
0053                 LOGIC(NQ(I))=0
0054                IF(NQ(I).LE.SOLNQ)GOTO 48
0055                WRITE(6,53)NQ(I)
0056        53      FORMAT('0',10X,'CONSTRAINT # ',I5,'  MUST NOT BE IN FINAL BASIS')
0057                NMISS = NMISS + 1
0058                CALL UPRIN( NQ(I))
0059        48      CONTINUE
0060        155     IF(SOLNQ.EQ.0) GOTO 169
0061                DO 47 I=1,SOLNQ
0062                IF(LOGIC(I).NE.1)GOTO 47
0063        159     FORMAT('0',10X,'CONS.# ',I5,' LAMDA',F16.7,' MISSING IN FB')
0064                 NMISS = NMISS + 1
0065                CALL UPRIN( -I)
0066                WRITE(6,159)I,XLAMDA(I)
0067        47      CONTINUE
0068        169      IF(NMISS.LE.0)GOTO 158
0069                 MFLAG=MFLAG+10
0070                 CALL UPRIN( 9003)
0071        158      IMF=0
0072                  IF(SOLNQ.EQ.0)GOTO 249
0073                DO 50 I=1,SOLNQ
0074                IF (DABS(XLAMDA(I)).LE.TOLAM)GOTO 50
0075        140     WRITE(6,150)I,XLAMDA(I)
0076                 ICON = -DABS(XLAMDA(I) ) * 100000
0077                CALL UPRIN( ICON)
0078                IMF=1
0079        50      CONTINUE
0080                 IF(IMF.EQ.0)GOTO 249
0081                 CALL UPRIN( 9004)
```

```
FORTRAN IV G LEVEL  21                CHECK              DATE = 80105

0082                 MFLAG=MFLAG+1
0083          150    FORMAT(/15X,'BASIS CONSTRAINT ',I4,' IS NOT ACTIVE -LAMDA',F10.6)
0084                 GOTO 249
0085          80     WRITE (6,90)
0086          90     FORMAT('0',/15X,'SORRY  THE  VECTOR X IS NOT WHAT IT SHOULD BE',/
                    1,15X,'IT SHOULD BE ',5X,'IT IS'     )
0087                 DO 100 I=1,M
0088                 NLINE=NLINE+1
              C      IF (NLINE.GT.50) CALL HEADER
0089          100    WRITE(IUNIT6, 110)SOLNX(I),X(I)
0090          110    FORMAT(' ',5X,F15.4,F15.4)
0091                 MFLAG=MFLAG+1000
0092                 GOTO 45
0093          249    IF (Q.EQ.SOLNQ) GOTO 248
0094          120     WRITE (6,130) SOLNQ,Q
0095          130    FORMAT('0',/15X,'NUMBER OF CONS. IN BASIS IS NOT SAME',/,5X,'SHOUL
                    1D BE ',5X,'AND IS',/2I10)
0096                 MFLAG = MFLAG + 100
0097                 CALL UPRIN( SOLNQ)
0098                 CALL UPRIN( -Q)
0099                 CALL UPRIN( 9000)
0100          248    IF (DABS(F-SOLNF).GT.1.D-5  *DABS(SOLNF)) MFLAG = -MFLAG
0101                 IF(DABS(F-SOLNF).GT.1.D-5  *DABS(SOLNF)) WRITE(6,70)SOLNF,F
0102          70     FORMAT('0',/15X,'SORRY, OPTIMUM VALUES ARE DIFF.',/15X,'SHOULD BE'
                    1,F15.6,5X,'BUT IS',F15.6)
0103                      ICC = MFLAG
0104                 RETURN
0105                 END
```

```
ADDR1 ADDR2   STMT   SOURCE STATEMENT

                 1 SINNER   CSECT
       C0000     2          USING *,15
       C000C     3          ST    14,12(13)
                 4          SAVE  (1,5)
                 5+         DS    0H
       C0018     6+         STM   1,5,24(13)                    SAVE REGISTERS
       C0070     7  N 14,=X'FDFFFFFF'
                 8  SPM 14
       00001     9          LA    4,1
       00001    10          LA    5,1
C0008           11          TM    8(1),128
       C0036    12          BO    START
       C000C    13          L     4,12(,1)
       00000    14          L     4,0(,4)
C000C           15          TM    12(1),128
       C0036    16          BO    START
       C0010    17          L     5,16(1)
       00000    18          L     5,0(,5)
       C0003    19 START    SLA   4,3
       00003    20          SLA   5,3
                21          SDR   0,0
                22          SDR   2,2
                23          SDR   4,4
       00000    24          LM    1,3,0(1)
       00000    25          L     3,0(3)
       00000    26 LOOP     LD    2,0(,1)
       00000    27          LD    4,0(,2)
                28          MDR   2,4
                29          ADR   0,2
                30          AR    1,4
                31          AR    2,5
       0004C    32          BCT   3,LOOP
       C000C    33  L 14,12(13)
                34  SPM 14
                35          RETURN (1,5)
       C0018    36+         LM    1,5,24(13)                    RESTORE THE REG
                37+         BR    14                            RETURN
                38          END
                39                =X'FDFFFFFF'
```

REFERENCES

1. Bartels, R. H., Golub, G. H., and Saunders, M. A. (1970).

   "Numerical techniques in mathematical programming", Nonlinear programming, Academic Press, N.Y.

2. Beale, E. M. L. (1959).

   "On quadratic programming", Naval research log. quarterly 6, pp. 227-243.

3. Biggs, M. C. (1975)

   "Constrained optimization using recursive quadratic programming: some alternative sub-problem formulations", Toward global optimization, eds. L. C. W. Dixon and G. P. Szego, North Holland Publishing Co., Amsterdam.

4. Boot, J. C. G. (1964)

   Quadratic programming, algorithms - anamolies - applications, North Holland Publishing Co., Amsterdam.

5. Bunch, J. W., and Kaufman, L. (1977)

   "Indefinite quadratic programming", Computing science technical report 61, Bell Labs., N.J.

6. Daniel, J. W., Graggs, W. B., Kaufman, L. and Stewart, G. W. (1976)

   "Reorthogonalization and stable algorithms for updating the Gram - Schmidt QR factorizations", Math. of Comp., 30 (136), Oct. 1976, pp. 772-795.

7. Dantzig, G. B. (1963)

   Linear programming and extensions, Princeton University press, Princeton, N.J.

8. Dembo, R. S. (1976)

   "A set of geometric programming test problems and their solutions", Math. Programming, 10, pp. 192-213.

9. Dorn, W. S. (1960)

   "Duality in quadratic programming", Quarterly Applied mathematics, 18, pp. 155-162.

10. Fletcher, R. (1970)

"The calculation of feasible points for linearly constrained optimization problems", UKAEA research report, AERE R. 6354.

11. Fletcher, R. (1970)

"A Fortran subroutine for general quadratic programming", UKAEA research report, AERE R. 6370.

12. Fletcher, R. (1971)

"A general quadratic programming algorithm", J. Inst. Math. Applications, 7, pp. 76-91.

13. Gill, P. E., Golub, G. H., Murray, W. and Saunders, M. A. (1974)

"Methods for modifying matrix factorizations", Math. Comp., v. 28, pp. 505-535.

14. Gill, P. E., Murray, W., and Saunders, M. A. (1975)

"Methods for computing and modifying the LDV factors of a matrix", Math. Comp., v. 29, pp. 1051-1077.

15. Goldfarb, D. (1969)

"Extension of Davidson's variable metric method for maximation under linear inequality and equality constraints", SIAM Journal of applied math. 17, pp. 739-764.

16. Goldfarb, D. (1972)

"Extension of Newton's method and simplex method for solving quadratic programs", Numerical methods for nonlinear optimization, ed. F. Lootsma, Academic press, London.

17. Goldfarb, D. (1975)

"Matrix factorizations in optimization of nonlinear functions subject to linear constraints", Math. Programming, 10(1975), pp. 1-31.

18. Goldfarb, D. and Reid, J. K. (1977)

"A practicable steepest edge simplex algorithm", Math. Programming, v. 12, pp. 361-371.

19. Goldfarb, D. (1977)

Steepest edge and related strategies for primal gradient projection methods for quadratic programming. Correspondence.

20. Goldfarb, D. (1979)

Updating of factors for steepest edge algorithms in quadratic programming. Correspondence.

21. Goncalves, A. S. (1972)

"A primal-dual method for quadratic programming with bounded variables', Numerical methods for nonlinear optimization, ed. F. Lootsma, Academic Press, New York.

22. Grigoriadis, M. D. and Ritter, K. (1969)

"A parametric method for semidefinite quadratic programs", SIAM J. control, v. 7, no. 4, pp. 559-577.

23. Han, S-P. (1975)

"A globally convergent method for nonlinear programming", Report no. 75-257 (Dept. of Computer science, Cornell University)

24. Han, S-P. (1976)

"Superlinearly convergent variable metric algorithms for general nonlinear programming problems" Math. Programming. v. 11, pp. 263-282.

25. Himmelblau, D. M. (1973)

Applied nonlinear programming, McGraw Hill.

26. Idnani, A. U. (1973)

"Extensions of Newton's method algorithms for solving convex quadratic programming problems - a computational experience", Master's thesis, Computer science dept., City College of New York.

27. Lenard, M. (1979)

"Active set strategies comparison for nonlinear programs", Math. Programming, v. 16, pp. 81-97.

28. Murray, W. (1970)

"An algorithm to find a local minimum of an indefinite quatratic program", Presented at the seventh symposium on Math. Programming.

29. Powell, M. J. D. (1977)

"Constrained optimization by a variable metric method",

30. Powell, M. J. D. (1977)

"The convergence of variable metric methods for nonlinearly constrained optimization calculations", Presented at Nonlinear Programming-3 symposium at Madison, Wisconsin.

31. Powell, M. J. D. (1977)

"A fast algorithm for nonlinearly constrained optimization calculations" Presented at the 1977 Dundee conference on numerical analysis.

32. Ritter, K. (1968)

"A method for solving maximum problems depending on parameters", Unternehmensforschung, v. 6, pp. 149-166.

33. Rosen, J. B. (1960)

"The gradient projection method for nonlinear programming, Part 1. Linear constraints", SIAM Journal of applied math. v. 8.

34. Rosen, J. B. and Suzuki, S. (1965)

"Construction of nonlinear programming test problems", Communications of the ACM, Feb. 1965, pp. 113.

35. Schitkowski, K. (1979)

"A numerical comparison of 13 nonlinear programming codes with randomly generated test problems", To appear in Numerical optimization of dynamic systems, eds. L. C. W. Dixon and G. P. Szego, North Holland Publishing co.

36. Schittkowski, K. (1969)

"The construction of degenerate, ill-conditioned and indefinite nonlinear programming problems and their usage to test optimization programs"

37. Theil, H. and Van de Panne, C. (1960)

"Quadratic programming as an extension of conventional quadratic maximization", Management Sciences, v. 7, pp. 1-20.

38. Van de Panne, C. and Whinston, A. (1964)

"The simplex and dual method for quadratic programming", Operations research quarterly, v. 15, pp. 422-441.

39. Van de Panne, C. (1975)

Methods for linear and quadratic programming, North Holland Pub. Co.

40. Wolfe, P. (1959)

"The simplex method for quadratic programming", Econometrica 27, pp. 382-398.