SOLUTION OF DEFINITE QUADRATIC PROGRAMMING PROBLEMS

by

Gregory Donald Shubert†

TR78-341

Department of Computer Science
Cornell University
Ithaca, NY 14853

†This research is supported in part by the National Science Foundation under grant ENG 77-02647.

SOULTION OF DEFINITE QUADRATIC PROGRAMMING PROBLEMS

by

Gregory Donald Shubert

78-341

Department of Computer Science
Cornell University
Ithaca, NY 14853

Abstract

An algorithm for solving the definite quadratic programming problem is presented. An implementation of this algorithm in FORTRAN is discussed. Numerical tests of this algorithm and a similar one not using the positive definiteness property show the former to be more stable. This algorithm is particularly suited for numerical methods for solving general nonlinear programming problems or minimax problems.

BIOGRAPHICAL SKETCH

Gregory D. Shubert was born December S, 1953 in Pontiac, Illinois. He graduated from Saunemin High School in 1972. He attended Illinois Wesleyan University in Bloomington, Illinois as an Illinois State Scholar and a National Merit Scholar. He concentrated in physics and mathematics, and did research in analog computation and astrophysics. He graduated Summa Cum Laude with a Bachelor of Arts degree in physics in 1976. During the summer of 1977 he worked on the development of programs for MINPACK at Argonne National Laboratory. He is a member of Phi Kappa Phi, an Associate Fellow of the British Interplanetary Society, and a Charter Member of the National Space Institute.

DEDICATION

To my dear friend and honored colleague,

Andrew F. Rex

ACKNOWLEOGEMENTS

This rosearch is supported in part by the National Science Foundation under grant ENG 77-02647.

TABLE OF CONTENTS

1. INTRODUCTION

We present an algorithm for solving a definite quadratic programming problem of the following form:

$$\begin{aligned} &\text{Minimize } a^T x + 1/2\, x^T Ax \\ &\text{subject to} \quad Bx = b \\ &\qquad\qquad\quad\ Cx <= c \end{aligned} \tag{DQP}$$

where A is an N-by-N symmetric and positive definite matrix,
B is an L-by-N matrix, and
C is an M-by-N matrix.

The motivation of this work is based on the fact that many recent nonlinear optimization techniques rely heavily on an efficient solution of this kind of quadratic programming problem. For instance, a general nonlinear programming problem can be effectively tackled by solving a sequence of such quadratic programming problems. In this case the positive definiteness of matrix A is usually maintained in order to achieve global convergence. The algorithm presented here takes advantage of the positive definiteness of matrix A and is readily adaptable to use in solving nonlinear programming or minimax problems [1,2].

In our algorithm the DQP is first reduced by some numerically stable processes such as Cholesky's and the QR decomposition of matrices to a solvable linear complementarity problem. This, in turn, is solved by Lemke's method.

In Section 2, we describe the Kuhn-Tucker conditions which must be satisfied by a solution to the DQP and show how a Kuhn-Tucker point and a solution to a linear complementarity problem (LCP) are related. In the next section, we show how to reduce the OPQ to a LCP, using Cholesky's and the QR decomposition of matrices. We describe Lemke's method in Section 4. In Section S, we discuss how the Kuhn-Tucker point and Lagrange multiplier can be recovered from the vectors obtained from the LCP. Section 6 describes the logical structure of the algorithms, the representations for some of the variables, numerical tolerances chosen for stopping criteria, and changes in the algorithm needed to handle the special cases of no equality or inequality constraints. The methods of generating test data and computational results are given in Section 7, and a summary is given in Section 8.

2. BASIC APPROACH

In this section we describe the Kuhn-Tucker conditions that must be satisfied by a solution to the definite quadratic programming problem (DQP). Then we show how a Kuhn-Tucker point and a solution to a linear complementarity problem (LCP) are related.

THEOREM 1.

A point x^* in R^N is a solution to the DQP if and only if there exists v^* in R^L and u^* in R^M such that the triple (x^*, v^*, u^*) satisfies the following conditions:

$$
\begin{aligned}
Ax + B^T v + C^T u + a &= 0 \\
-Bx + b &= 0 \\
-Cx + c &>= 0 \\
u &>= 0 \\
u^T (c - Cx) &= 0
\end{aligned}
\tag{2.1}
$$

For a proof of this theorem, readers are referred to [4]. Conditions (2.1) are known as the Kuhn-Tucker conditions for the DQP. If a triple (x^*, v^*, u^*) satisfies the Kuhn-Tucker conditions, then x^* is called a Kuhn-Tucker point of the DQP, and (v^*, u^*) are called the Lagrange multipliers of the Kuhn-Tucker point x^* with respect to the equality and inequality constraints, respectively.

If we define

$$w = c - Cx, \tag{2.2}$$

then the Kuhn-Tucker conditions become

$$\begin{bmatrix} A & B^T & C^T \\ -B & 0 & 0 \\ -C & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \\ u \end{bmatrix} + \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ w \end{bmatrix}$$

$$u >= 0,\ w >= 0,\ u^T w = 0. \qquad (2.3)$$

Because of Theorem 1 we are content with finding vectors x^*, u^*, v^* and w^* that satisfy (2.3). This can be done through a relationship between condition (2.3) and a linear complementarity problem. This relationship, a basis for our algorithm is contained in the following theorem.

THEOREM 2.

Let A be a positive definite and B have full row rank. If x^* is a Kuhn-Tucker point of the DQP, if (v^*, u^*) are the Lagrange multipliers, and if $w^* = c - Cx$, then the vectors u^* and w^* solve the linear complementarity problem:

$$w = q + Du \qquad (2.4)$$

$$w >= 0, \quad u >= 0, \quad u^T w = 0$$

where

$$q = c + [C\ 0]H^{-1} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$D = [C\ 0]H^{-1} \begin{bmatrix} C^T \\ 0 \end{bmatrix}$$

$$H = \begin{bmatrix} A & B^T \\ -B & 0 \end{bmatrix}.$$

Conversely, if u^* and v^* solve (2.4) and if x^* and v^* solve the following system of linear equations:

$$H \begin{bmatrix} x \\ v \end{bmatrix} = - \begin{bmatrix} a + C^T u^* \\ b \end{bmatrix} \qquad (2.5)$$

then x^* is a Kuhn-Tucker point of the DQP and (v^*, u^*) are its Lagrange multipliers.

PROOF.

We first prove the nonsingularity of the matrix H by showing that $H \begin{bmatrix} x \\ y \end{bmatrix} = 0$ implies $x = 0$ and $y = 0$. Substituting for H, we have $Ax + B^T y = 0$ and $Bx = 0$. Premultiplying the first equation by x^T, we have $x^T Ax + x^T B^T y = 0$. Since $Bx = 0$, $x^T B^T = 0$, and we have $x^T Ax = 0$. Since A is positive definite, this implies $x = 0$. Returning to the equation $Ax + B^T y = 0$ and using the fact that $x = 0$, we have $B^T y = 0$. Since B has full row rank, this implies $y = 0$. Thus H^{-1} exists.

From (2.3) we have

$$H \begin{bmatrix} x \\ v \end{bmatrix} = - \begin{bmatrix} a \\ b \end{bmatrix} - \begin{bmatrix} C^T \\ 0 \end{bmatrix} u,$$

$$- [\, C \;\; 0 \,] \begin{bmatrix} x \\ v \end{bmatrix} + c = w,$$

$$u >= 0, \quad w >= 0, \quad u^T w = 0.$$

Taking the first equation immediately above and premultiplying by H^{-1}, we solve for x and v. Substituting this in the second equation gives us (2.4).

Conversely, if u^* and w^* solve (2.4) and x^* and v^* solve (2.5), then obviously x^*, v^*, u^*, and w^* satisfy (2.3), which is equivalent to the Kuhn-Tucker condition (2.1). The proof is then complete.

QED.

3. REDUCTION TO A LINEAR COMPLEMENTARITY PROBLEM

From the results of Section 2 we can reduce the DQP to a linear complementarity problem simply by generating matrix O and vector q of (2.4). Since the Cholesky decomposition of a positive definite matrix and the QR decomposition of a general matrix are numerically stable, they are frequently used in carrying out this reduction. First, we exploit the special structure of the DQP and the positive definiteness of matrix A to obtain an LU decomposition of matrix H.

Theorem 3.

Let A be positive definite and let $A = LL^T$ be its Cholesky decomposition. Let an N-by-L matrix G be constructed by

$$LG = B^T. \tag{3.1}$$

and let matrix G have a QR decomposition

$$QG = \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{3.2}$$

where Q is an N-by-N orthogonal matrix and R is an L-by-L upper triangular matrix. Then

$$H = \begin{bmatrix} L & 0 \\ -G^T & R^T \end{bmatrix} \begin{bmatrix} L^T & G \\ 0 & R \end{bmatrix} \tag{3.3}$$

is an LU decomposition of H.

PROOF.

$$\begin{bmatrix} L & 0 \\ -G^T & R^T \end{bmatrix} \begin{bmatrix} L^T & G \\ 0 & R \end{bmatrix} = \begin{bmatrix} LL^T & LG \\ -G^TL^T & R^TR - G^TG \end{bmatrix}$$

It sufficos to show $R^TR = G^TG$. This is obvious because

$$G^TG = [\ R^T, 0\ ]\ QQ^T \begin{bmatrix} R \\ 0 \end{bmatrix} = [\ R^T\ 0\ ] \begin{bmatrix} R \\ 0 \end{bmatrix} = R^TR$$

QED.

The computations of matrix D and vector q are given in Theorems 4 and S respectively. We note here that a very important advantage of our algorithm is that, even with the presence of rounding errors, the positive semidefiniteness of the matrix D is always ensured. Note also that the positive semidefiniteness is essential for the effectiveness of Lemke's method for solving the linear complementarity problem.

THEOREM 4.

Let A be positive definite and let $A = LL^T$ be its Cholesky decomposition. Let E be an N-by-M matrix satisfying

$$LE = C^T \tag{3.4}$$

If Q is the orthogonal matrix defined in Theorem 3, and if

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = QE \tag{3.S}$$

where Z_1 is an L-by-M matrix and Z_2 is an (N-L)-by-M matrix, and if D is the matrix defined in Theorem 2, then

$$D = Z_2^TZ_2 \tag{3.6}$$

PROOF.

Recall the definition of D from Theorem 2:

$$D = [\ C, 0\ ]\ H^{-1} \begin{bmatrix} C^T \\ 0 \end{bmatrix}$$

From (3.3) we also have

$$H = \bar{L}\bar{U}$$

where

$$\bar{L} = \begin{bmatrix} L & 0 \\ -G^T & R^T \end{bmatrix}$$

$$\bar{U} = \begin{bmatrix} L^T & G \\ 0 & R \end{bmatrix}$$

By a direct calculation, we have

$$\bar{L}^{-1} = \begin{bmatrix} L^{-1} & 0 \\ -K^T & R^{-1T} \end{bmatrix}$$

$$\bar{U}^{-1} = \begin{bmatrix} L^{-1T} & K \\ 0 & R^{-1} \end{bmatrix}$$

where K is an N-by-L matrix satisfying

$$L^TK + GR^{-1} = 0.$$

Because $QG = \begin{bmatrix} R \\ 0 \end{bmatrix}$, from (3.2), we then have

$$K = -L^{-1T}Q^TI_{N,L} \tag{3.7}$$

where $I_{N,L}$ consists of the first L columns of the identity matrix I_N.

Therefore,

$$D = [\ C,\ 0\ ]\ \bar{U}^{-1}\bar{L}^{-1} \begin{bmatrix} C^T \\ 0 \end{bmatrix}$$

$$= [\ C,\ 0\ ] \begin{bmatrix} L^{-1T} & K \\ 0 & R^{-1} \end{bmatrix} \begin{bmatrix} L^{-1} & 0 \\ -K^T & R^{-1T} \end{bmatrix} \begin{bmatrix} C^T \\ 0 \end{bmatrix}$$

$$= [\, C,\ 0 \,] \begin{bmatrix} L^{-1T}L^{-1} - KK^T & KR^{-1} \\ -R^{-1}K^T & R^{-1}R^{-1T} \end{bmatrix} \begin{bmatrix} C^T \\ 0 \end{bmatrix}$$

$$= C\,(L^{-1T}L^{-1} - KK^T)\,C^T$$

Using (3.7) for K and K^T, we have

$$0 = C\,(L^{-1T}L^{-1} - L^{-1T}Q^T I_{N,L} I_{N,L}^T Q L^{-1})\,C^T$$

$$= CL^{-1T}Q^T \quad (I_N - I_{N,L} I_{N,L}^T)\; QL^{-1}C^T$$

From (3.4) and (3.5), we have

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = QE = QL^{-1}C^T$$

Thus,

$$0 = [\, Z_1^T,\ Z_2^T \,]\; (I_N - I_{N,L} I_{N,L}^T) \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}$$

$$= [\, Z_1^T,\ Z_2^T \,] \begin{bmatrix} 0 & 0 \\ 0 & I_{N,L} \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}$$

$$= Z_2^T Z_2$$

QED.

THEOREM 5.

$$q = c - Z_1^T d + Z_2^T z_2 \tag{3.8}$$

where

$R^T d = b$

$Le = a$

$Qe = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$. z_1 is an L-vector, and z_2 is an (N-L)-vector

PROOF.

From (2.4), q is defined to be

$$q = c + [\,C,\ 0\,]\ H^{-1} \begin{bmatrix} a \\ b \end{bmatrix}$$

Substituting for $H^{-1} = \bar{U}^{-1}\ \bar{L}^{-1}$, we have

$$q = c + [\,C,\ 0\,] \begin{bmatrix} L^{-1T} & K \\ 0 & R^{-1} \end{bmatrix} \begin{bmatrix} L^{-1} & 0 \\ -K^T & R^{-1T} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$= c + [\,CL^{-1T},\ CK\,] \begin{bmatrix} L^{-1}a \\ -K^T a + R^{-1T} b \end{bmatrix}$$

$$= c + CL^{-1T}L^{-1}a - CKK^T a + CKR^{-1T}b$$

Using (3.7) for K and K^T, we have

$$q = c + CL^{-1T}L^{-1}a - CL^{-1T}Q^T I_{N,L} I_{N,L}^T QL^{-1}a$$

$$+ CKR^{-T}b$$

$$q = c + CL^{-1T}Q^T\ (I_N - I_{N,L}I_{N,L}^T)\ QL^{-1}a$$

$$- CL^{-1T}Q^T I_{N,L} R^{-1T} b$$

By our assumptions, $d = R^{-1T}b$

Also, since by (3.4) and (3.5)

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = QE = QL^{-1}C^T,$$

$$Z_1 = I_{N,L}^T QL^{-1}C^T$$

Thus,

$$Z_1^T d = CL^{-1T}Q^T I_{N,L} R^{-1T} b$$

Now, from (3.4),

$$QL^{-1}a = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

Note that

$$[\ Z_1^T,\ Z_2^T\ ] = CL^{-1T}Q^T$$

Thus,

$$\begin{aligned} CL^{-1T}Q^T\ (I_N - I_{N,L}I_{N,L}^T)\ QL^{-1}a \\ &= [\ Z_1^T,\ A_2^T\ ] \begin{bmatrix} 0 & 0 \\ 0 & I_{N,L} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \\ &= Z_2^T z_2 \end{aligned}$$

Therefore,

$$q = c - Z_1^T d + Z_2^T z_2$$

QEO.

Thus, the reduction of the definite quadratic programming problem to a linear complementarity problem (2.4) proceeds as follows:

1.	$A = LL^T$	Solve for L by Cholesky decomposition
2.	$LG = B^T$	Solve for G by forward substitution

3. $LE = C^T$ Solve for E by forward substitution

4. $Le = a$ Solve for e by forward substitution

5. $QG = \begin{bmatrix} R \\ 0 \end{bmatrix}$ Solve for Q and R using QR decomposition

6. $\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = QE$ Compute Z_1 and Z_2

7. $\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = Qe$ Compute z_1 and z_2

8. $R^T d = b$ Solve for d by forward substitution

9. $D = Z_2^T Z_2$ Compute D

10. $q = c - Z_1^T d + Z_2^T z_2$ Compute q

4. LEMKE'S METHOD

Lemke's method is used to solve the linear complementarity problem (2.4), which is restated as follows:

Given an M-by-M matrix D and an M-vector q, find M-vectors u and w satisfying

$$\begin{aligned} & w = q + Du \\ & w >= 0, \quad u >= 0, \quad u^T w = 0 \end{aligned} \qquad \text{(LCP)}$$

A point (w, u) is feasible if it satisfies w = q + Du and if w >= 0 and u >= 0. A point is complementary if $u^T w = 0$. The linear complementarity problem is to find a complementary feasible point.

Lemke's method begins by adding a scalar variable u', called the dummy variable, to the system of equations in the LCP, and by modifying matrix D by adding an (M+1)st column d of positive components to it, so that we have

$$w = q + D' \begin{bmatrix} u \\ u' \end{bmatrix} \qquad (4.1)$$

where D' = [D, d]. Now we have 2M + 1 variables: w, u, and u'. These variables come in two flavors: basic and nonbasic. The basic variables are the M variables on the left hand side of (4.1); the nonbasic variables are the M + 1 variables on the right hand side of (4.1). Initially, the basic variables are the components of w; the nonbasic variables are u' and the components of u.

Suppose q >= 0. Then we set u = 0 and u' = 0 and w = q; and the LCP is solved. And since u' = 0, the addition of this dummy variable doesn't alter the problem.

Since some components of q may be negative, we try to state the LCP in different terms, by expressing some variables in terms of others in a process called pivoting. If $D'_{i,j} \neq 0$, we may pivot on (4.1) by solving the ith equation in the system for variable u_j, then replacing u_j in each of the remaining M equations.

$$w_i = q_i + \mathrm{SUM}_{k=1}^{M+1} \; D'_{i,k} u_k$$

$$w_i = q_i + D'_{i,j} u_j + \mathrm{SUM}_{\substack{k=1 \\ k \neq j}}^{M+1} \; D'_{i,k} u_k$$

$$u_j = (w_i - q_i - \mathrm{SUM}_{\substack{k=1 \\ k \neq j}}^{M+1} \; D'_{i,k} u_k) \; / \; D'_{i,j}$$

Substituting this for u_j in (4.1) we get (4.2).

$$w^t = q^t + D'^t u^t \tag{4.2}$$

This equation expresses the same problem as (4.1) in a similar form. But the basic and nonbasic sets have changed. The pair (w_i, u_j) have been exchanged; w_1 is now nonbasic, and u_j is now basic.

A basic point $(w^t, u^t) = (q^t, 0)$ is obtained from (4.2) by setting all nonbasic variables u^t to zero. A feasible basic point is a basic form (4.2) whose basic point is feasible, that is, both w^t and u^t are nonnegative. A feasible pivot is a pivot from one feasible basic form to another.

The first pivot is (w_r, u') where w_r is chosen in the following way. The pivoting column is the (M+1)st column, corresponding to variable u'. Since $u = 0$, the equation for each component of the w vector is simple:

$$w_i = q_i + O_{i,M+1}u' \tag{4.3}$$

We increase u' from zero, but only enough to make w positive. Thus we choose a pivoting row r that minimizes $q_r / O_{r,M+1}$, for all negative q_r. This quantity is actually largest in absolute value.

After the first pivot, q^t is positive. Also, u' is now on the left hand side of the equation (4.2) and is a basic variable, and w_r has become nonbasic. But u_r is also a nonbasic variable and this violates complementarity. The next pivoting column must be the r-th, corresponding to u_r.

Now we exchange u_r with one of the basic variables. Let us imagine that the situtation is the following:

$$w = q + O' \begin{bmatrix} u \\ u' \end{bmatrix}$$

$$\begin{array}{ccccccccccccc}
w & & q & & \vdots & \vdots & + & \vdots & \vdots & \vdots & \vdots & & u \\
w & & q & & \vdots & \vdots & - & \vdots & \vdots & \vdots & \vdots & & u \\
\cdot & & \cdot & & \vdots & \vdots & + & \vdots & \vdots & \vdots & \vdots & & \cdot \\
\cdot & = & \cdot & + & \vdots & \vdots & - & \vdots & \vdots & \vdots & \vdots & & \cdot \\
\cdot & & \cdot & & \vdots & \vdots & + & \vdots & \vdots & \vdots & \vdots & & \cdot \\
w & & q & & \vdots & \vdots & + & \vdots & \vdots & \vdots & \vdots & & \cdot \\
 & & & & & & & & & & & & u
\end{array}$$

column r

In the pivoting column there are two negative elements, denoted by minus signs. The other elements are positive. As we increase u_r, the values of the elements in w corresponding to these negative elements will decrease. At some point, a component of w in some row will become zero. This we choose as our pivoting row. If two or more w's become zero at once, we choose the dummy variable, if it is among them, in order to make it nonbasic again.

Several events can cause this pivoting sequence to stop. If q has become negative, due to rounding errors, then there is no way, setting u = 0, that w >= 0. We note that this will never occur in exact arithmetic. If there are no negative elements in the pivot column, then we can not increase u_r and simultaneously decrease some w to zero, so we can not pivot. In this case the problem has no feasible solution. If either of these two cases arise, the pivoting sequence stops.

If the dummy variable u' is pivoted back into the right hand side and becomes nonbasic again, then we can set all the nonbasic variables to zero, and solve the problem. Also, if the value of the element in the q-vector corresponding to the dummy variable is zero, then the value of the dummy variable doesn't matter. Again, we set all the nonbasic variables to zero, and are done.

Lemke [3] shows that if O is positive semidefinite, then the pivoting sequence is guaranteed to stop, either with a solution, or with an indication that a feasible solution does not exist.

5. FINDING THE KUHN-TUCKER POINT AND MULTIPLIER

In the reduction from the OQP to the linear complementarity problem, we essentially eliminate vectors x and v from system (2.3). In this section we discuss how the Kuhn-Tucker point x^* and the Lagrange multiplier v^* can be effectively recovered from the vectors u^* and w^*, which are obtained from solving the linear complementarity problem (2.4).

THEOREM 6.

If u^* and w^* solve the linear complementarity problem then the Kuhn-Tucker point x^* and the Lagrange multiplier v^* can be obtained by solving

$$L^T x = -Q^T \begin{bmatrix} -d \\ Z_2 u^* + z_2 \end{bmatrix}$$

$$Rv = -(Z_1 u^* + z_1 + d)$$

PROOF.

From (2.5) in Theorem 2, the Kuhn-Tucker point x^* and the Lagrange multiplier v^* solve the system

$$H \begin{bmatrix} x \\ v \end{bmatrix} = - \begin{bmatrix} a + C^T u^* \\ b \end{bmatrix}$$

Since, from (3.3),

$$H = \begin{bmatrix} L & 0 \\ -G^T & R^T \end{bmatrix} \begin{bmatrix} L^T & G \\ 0 & R \end{bmatrix}$$

and, from the proof of Theorem 4,

$$\begin{bmatrix} L & 0 \\ -G^T & R^T \end{bmatrix}^{-1} = \begin{bmatrix} L^{-1} & 0 \\ -K^T & R^{-1T} \end{bmatrix}$$

where

$$K = -L^{-1T}Q^T I_{N,L}$$

we have

$$\begin{bmatrix} L^T & G \\ 0 & R \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} = - \begin{bmatrix} L^{-1} & 0 \\ -K^T & R^{-1T} \end{bmatrix} \begin{bmatrix} a + C^T u^* \\ b \end{bmatrix}$$

or

$$L^T x + Gv = -L^{-1}a - L^{-1}C^T u^*$$

$$Rv = K^T(a + C^T u^*) - R^{-1T} b$$

Note that, from (3.4), (3.5), and (3.8),

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = QL^{-1}C^T$$

$$R^T d = b$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = QL^{-1}a$$

Thus

$$Rv = -I_{N,L}^T QL^{-1}a - I_{N,L}^T QL^{-1}C^T u^* - R^{-1T}b$$

$$= -z_1 - Z_1 u^* - d$$

To solve for x, we have

$$L^T x = -Gv - L^{-1}a - L^{-1}C^T u^*$$

Solving the above equation for v and substituting yields

$$L^T x = GR^{-1}(z_1 + Z_1 u^* + d) - Q^T(QL^{-1}a + QL^{-1}C^T u^*)$$

From (3.2), we have

$$QG = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

So

$$GR^{-1} = Q^T I_{N,L}$$

Substituting yields

$$L^T x = Q^T I_{N,L}(z_1 + Z_1 u^* + d) - Q^T \left(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} Z_1 u^* \\ Z_2 u^* \end{bmatrix} \right)$$

$$= -Q^T \begin{bmatrix} -d \\ z_2 + Z_2 u^* \end{bmatrix}$$

QEO.

In summary, we compute x^* and v^* from the vector u^* as follows:

1. $L^T x = -Q^T \begin{bmatrix} -d \\ Z_2 u^* + z_2 \end{bmatrix}$ Solve for x by forward substitution

2. $RV = -(Z_1 u^* + z_1 + d)$ Solve for v by forward substitution

6. IMPLEMENTING THE ALGORITHM

In this section we describe the logical structure of the algorithms, the particular representations for some of the variables, numerical tolerances chosen for stopping criteria, and changes in the algorithm needed to handle the special cases of no equality or inequality constraints.

The algorithms are implemented in FORTRAN. Various subroutines are arranged in the following hierarchical structure:

```
          OEFQP              -- solves definite quadratic
         /  |  \                programming problems
   OECOMP   |   \            -- performs decompositions
          LEMKE  \           -- solves LCP by Lemke's method
                 EVALU       -- evaluates the results

          OECOMP
         /  |   \
   CHOLES   |    \           -- Cholesky decomposition
          FORWAR  \          -- Forward substitution
                  QR         -- QR decomposition

          LEMKE
         /     \
     CHOOSE     \            -- Chooses a pivot pair
               PIVOT         -- Performs a pivot
```

In order to test these subroutines, a driver and associated routines are useful:

```
          MAIN
         / |  |\
      GEN  |  | \            -- Generates test data
       OEFQP  |  \           -- Solves OQP
           PRINT  \          -- Prints out results
                 REPORT      -- Reports on error codes
```

Generally, a vector or matrix is represented in a straightforward fashion as a one-or two-dimensional array. In several instances we choose a different representation.

After matrix A has been decomposed using Cholesky's decomposition into a lower triangular matrix L, L is stored in condensed form in a one-dimensional array. Since A is N-by-N, L has $(N^2 + N)/2$ nonzero elements, and the one-dimensional array need only be this long. We use a scheme that maps successive columns of L into the one-dimensional array, and a function INDEX(i,j) that returns the appropriate position in the one-dimensional array for the element $L_{i,j}$.

In the QR decomposition of matrix G, we have

$$QG = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Matrix Q is formed by the product of several simpler orthogonal matrices Q(1), Q(2), and so on. These matrices may be expressed in terms of a single vector. These vectors are stored along the diagonal and in the lower triangle of G (which is overwritten). The upper triangular matrix R is then stored in the upper triangle of G and its diagonal elements are stored in an auxiliary vector.

Although Lemke's method is guaranteed to halt after a finite number of pivots, a large number of pivots may be quite expensive. The user can set a maximum on the number of Lemke iterations in the integer variable LIMIT. A default value of LIMIT = M + 5 (where M is the number of inequality constraints) is used.

The stopping criteria in Lemke's method (discussed in Section 4) are changed slightly to allow for rounding error, by comparisons with user-defined tolerances stored in a vector TOL.

Successful termination occurs if:

1. The dummy variable becomes nonbasic; or

2. The value of the component of q corresponding to the dummy variable is smaller than TOL(1).

Otherwise, iteration continues as long as:

1. All components of q are larger than -TOL(1); and

2. Any element in the pivot column in the O matrix is less than -TOL(2); and

3. The pivot elements are large enough -- the absolute value of the pivot element in the O matrix is greater than or equal to TOL(3) times the infinity norm of the pivot column.

Default values of 10^{-4}, 10^{-6}, and 10^{-8} for TOL(1),TOL(2), and TOL(3), respectively, are used.

Several special cases may occur in defining the constraints: there may be no equality constraints; there may be no inequality constraints; or there may be neither equality nor inequality constraints. In these cases the vectors and matrices b, B, c, and C would have zero rows. Dimensions of zero are not permitted in FORTRAN. Instead, we use dimensions of 1, and use logical variables EQCON and NQCON to signal the presence of equality or inequality constraints, respectively.

In Theorems 4 and 5 a matrix and a vector with the following form were defined:

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \qquad \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad \begin{array}{l} \text{L rows} \\ \text{N-L rows} \end{array}$$

where L is the number of equality constraints and N is the number of unknowns. In addition, the first matrix has M columns, where M is the number of inequality constraints. If there are no equality constraints ($L = 0$), Z_1 and z_1 do not exist. If there are no inequality constraints ($M = 0$), the matrix on the left does not exist. And if $L = M = 0$, only z_2 exists.

In addition, G, a matrix derived from B, the matrix of equality constraints, is factored using QR decomposition. If there are no equality constraints ($L = 0$), then matrix Q does not exist.

These special cases result in slightly different expressions of Theorem 6, which follow in a straightforward manner from the details in the paragraphs above.

An integer variable IFLAG is used as a parameter to communicate information about errors among the various subroutines. Generally, an error causes an immediate return from that subroutine and the subroutines that called it, back to the main procedure. A separate subroutine, REPORT, can be called with IFLAG as an argument to print a suitable error message, thus eliminating the need for error messages to be printed in any of the computational subroutines.

7. COMPUTATIONAL RESULTS

In this section we discuss the method used for generating test data and some of the results of these tests.

Since there seem to be no widely accepted standard test cases for definite quadratic programs, we devised a method of generating test matrices. All the matrices and vectors used in a definite quadratic program were generated using a random number generator. The signs of the numbers were distributed evenly between plus and minus. The magnitudes of the numbers were distributed logarithmically over a specified range. A logarithmic distribution was chosen so that there would be about as many numbers in the smallest decade as in the largest and intermediate decades. In the tables below, the endpoints of the ranges are given. A larger range means the sizes of the numbers may vary more widely.

We compared our algorithm against a similar one that uses Lemke's method, but that does not take advantage of the positive definiteness of matrix A. A scheme not using this advantage would carry out the reduction to the linear complementarity problem by using the Gauss-Jordan method to eliminate the vectors x and v from the following system directly:

$$\begin{bmatrix} a \\ b \\ b \end{bmatrix} + \begin{bmatrix} A & B^T & C^T \\ -B & 0 & 0 \\ -C & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ w \end{bmatrix} \tag{7.1}$$

The differences in the space requirements of these two methods are largely due to the amount of space needed to perform this reduction. These are summarized as follows:

Approximate Space Requirements

Using Gauss-Jordan elimination:	$(L + M + N)^2 + LN + MN + M^2 + N^2$
Taking advantage of special structure	$2LN + 2MN + M^2 + 3/2\ N^2$

As an example of the space savings, for a large problem involving 50 unknowns, 15 equality constraints, and 20 inequality constraints, the difference amounts to about 33 thousand bytes, if double precision variables are used. This advantage is somewhat offset by the fact that our algorithm is longer and more complex. When compiled under FORTRAN H, the difference in the size of the object modules produced was about 7 thousand bytes (11 versus 4).

All timings cited here are for programs compiled under FORTRAN H Extended with full optimization, and run on an IBM 370/168 under CMS, and average timings are given in milliseconds.

To evaluate numerical stability the solutions are substituted into the original problem and the residuals are calculated. Given the need to compare vectors of small numbers, we decided a good measure of residual size is the average negative logarithm of the absolute values of the components of the residual vector, a refinement of simply averaging the negatives of the exponents of the residuals.

1. $a + Ax + B^T v + C^T u = 0$
2. $b - Bx = 0$ $\qquad\qquad N > 0$

In the tables below, we refer to our algorithm as A, and the alternate algorithm as B.

For comparison, the machine epsilon for double precision on the

IBM 370 (the smallest number which can be added to one to yield a result greater than one) is 2^{-56} or 1.4×10^{-17}. Its negative logarithm is 16.86. A residual in the tables of 14.9 is on the order of about 100 times the machine epsilon, or a loss of 2 significant digits.

Some results are:

N = 3 L = 1 M = 1 L + M + N = 5

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-1.0	1.5	15.4	15.3	14.8	15.4		
-1.5	1.5	15.6	15.6	16.3	16.3	8	8
-2.5	1.0	16.1	16.4	15.4	15.7		

N = 5 L = 0 M = 3 L + M + N = 8

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-1.0	1.5	14.8	12.7	---	---		
-1.5	1.0	15.5	14.5	---	---	8	8
-1.5	1.5	15.3	14.7	---	---		

N = 8 L = 0 M = 0 L + M + N = 8

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-1.0	1.5	11.3	11.3	---	---		
-1.5	1.0	12.9	12.9	---	---	8	8
-1.5	1.5	12.8	12.4	---	---		

N = 5 L = 4 M = 0 L + M + N = 9

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-2.0	1.5	13.6	11.6	14.6	11.9		
-1.5	2.0	13.7	12.8	14.5	13.8	8	8
-1.5	2.5	13.9	13.3	15.4	13.3		

N = 10 L = 3 M = 5 L + M + N = 18

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-0.1	1.5	14.3	13.4	14.5	14.1		
-0.5	1.0	13.9	12.4	14.0	12.6		
-1.0	1.5	13.3	10.9	15.4	11.2	12	10
-1.0	1.5	13.1	12.4	13.6	12.5		
-1.5	1.0	13.7	10.9	13.4	11.2		
-1.5	1.5	12.8	10.4	14.5	10.6		

N = 15 L = 5 M = 5 L + M + N = 25

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-1.0	1.5	13.2	10.8	13.4	11.0		
-1.5	1.0	14.0	11.7	14.3	12.2	18	13
-1.5	1.5	10.6	5.8	12.1	5.9		

N = 20 L = 10 M = 5 L + M + N = 35

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-1.0	1.5	12.3	8.1	12.6	7.8		
-1.5	1.0	11.2	6.4	12.6	7.0		
-1.5	1.5	8.7	2.0	11.0	1.8	35	19
-2.0	2.5	9.4	2.9	10.0	3.7		
-2.5	2.0	9.9	0.8	9.7	1.5		
-3.0	3.0	6.0	fail	7.2	fail		

N = 35 L = 15 M = 30 L + M + N = 60

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-1.0	1.5	6.7	-1.6	8.7	-0.9		
-1.5	1.0	9.1	1.8	10.4	2.3		
-1.0	1.0	9.6	3.7	10.8	3.9	97	49
-1.0	0.5	10.0	2.2	11.3	2.1		
-0.5	1.0	10.2	4.9	11.5	5.3		

N = 50 L = 15 M = 20 L + M + N = 85

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-0.5	1.0	9.0	fail	8.6	fail		
0.0	0.5	10.5	4.2	10.6	4.2	240	110
-0.5	0.5	9.2	1.8	9.4	1.9		

Our algorithm, A, consistently shows better stability than algorithm B, as evidenced by the higher ratings in the residual tables. The differences become more pronounced in the larger problems, sometimes amounting to 5 and 6 orders of magnitude in the size of the residuals. In several cases, our algorithm succeeded in finding a solution while algorithm B failed. In one case, in the N = 35 table, algorithm B's "solution" is not a solution at all, because the residuals are on the order of 100 (whose negative logarithm is -2). For some problems, not shown here, both algorithms failed for a large random number range, because the random matrix A generated was very nearly singular.

For small problems, the timings are about the same. For the large problems, our algorithm runs about twice as slowly as algorithm B.

The number of Lemke iterations needed to solve the various problems is given in the table below.

Problem size N	L	M	Lemke iterations Min.	Max.	Ave.
3	1	1	0	2	1.3
5	0	3	2	4	3.0
8	0	0	0	0	0.0
5	4	0	0	0	0.0
10	3	5	0	5	3.2
15	5	5	3	5	4.0
20	10	5	2	5	3.8
35	15	10	7	9	7.6
50	15	20	10	10	10.0

In summary, taking advantage of special structure and positive definiteness gains a significant measure of improved numerical stability. Our implementation of this algorithm uses less space for array storage, but runs more slowly than an algorithm not using these advantages.

8. CONCLUSIONS

The algorithm developed in this thesis is for effectively solving a definite quadratic programming problem. It takes advantage of the special structure of the problem and positive definiteness of the matrix. Our computational results indicate that it is less prone to rounding error accumulation than a similar method not taking advantage of the special structure.

9. REFERENCES

1. Han, S.P. Superlinearly convergent variable metric algorithms for general nonlinear programming problems. Mathematical Programming. 11 (1976) 263-282.

2. Han, S.P. Variable metric methods for minimizing a class of nondifferentiable functions. Computer Science Department, Technical Report TR77-322.

3. Lemke, C.E. On complementary pivot theory. Pages 95-114 of Mathematics of Decision Sciences. G.B. Dantzig and A.F. Veinott, Jr. (eds.), Vol. 11 of Lectures in Applied Mathematics. American Mathematical Society, 1968.

4. Mangasarian, Olvi L. Nonlinear Programming. New York: McGraw-Hill, 1969.

N = 50 L = 15 M = 20 L + M + N = 85

Random range		Residual 1 A	Residual 1 B	Residual 2 A	Residual 2 B	Time (ms) A	Time (ms) B
-0.5	1.0	9.0	fail	8.6	fail		
0.0	0.5	10.5	4.2	10.6	4.2	240	110
-0.5	0.5	9.2	1.8	9.4	1.9		

Our algorithm, A, consistently shows better stability than algorithm B, as evidenced by the higher ratings in the residual tables. The differences become more pronounced in the larger problems, sometimes amounting to 5 and 6 orders of magnitude in the size of the residuals. In several cases, our algorithm succeeded in finding a solution while algorithm B failed. In one case, in the N = 35 table, algorithm B's "solution" is not a solution at all, because the residuals are on the order of 100 (whose negative logarithm is -2). For some problems, not shown here, both algorithms failed for a large random number range, because the random matrix A generated was very nearly singular.

For small problems, the timings are about the same. For the large problems, our algorithm runs about twice as slowly as algorithm B.

The number of Lemke iterations needed to solve the various problems is given in the table below.

Problem size N	L	M	Lemke iterations Min.	Max.	Ave.
5	1	1	0	2	1.3
5	0	3	2	4	3.0
8	0	0	0	0	0.0
5	4	0	0	0	0.0
10	3	5	0	5	3.2
15	5	5	3	5	4.0
20	10	5	2	5	3.8
35	15	10	7	9	7.6
50	15	20	10	10	10.0

In summary, taking advantage of special structure and positive definiteness gains a significant measure of improved numerical stability. Our implementation of this algorithm uses less space for array storage, but runs more slowly than an algorithm not using these advantages.

8. CONCLUSIONS

The algorithm developed in this thesis is for effectively solving a definite quadratic programming problem. It takes advantage of the special structure of the problem and positive definiteness of the matrix. Our computational results indicate that it is less prone to rounding error accumulation than a similar method not taking advantage of the special structure.

9. REFERENCES

1. Han, S.P. Superlinearly convergent variable metric algorithms for general nonlinear programming problems. Mathematical Programming. 11 (1976) 263-282.

2. Han, S.P. Variable metric methods for minimizing a class of nondifferentiable functions. Computer Science Department, Technical Report TR77-322.

3. Lemke, C.E. On complementary pivot theory. Pages 95-114 of Mathematics of Decision Sciences. G.B. Dantzig and A.F. Veinott, Jr. (eds.), Vol. 11 of Lectures in Applied Mathematics. American Mathematical Society, 1968.

4. Mangasarian, Olvi L. Nonlinear Programming. New York: McGraw-Hill, 1969.

APPENDIX 1

Driver and support programs

MAIN

GEN

RANDOM

URAND

PRINT

REPORT

```
C**********
C
C  THIS PROGRAM SOLVES A QUADRATIC PROGRAMMING PROBLEM OF
C   THE FOLLOWING TYPE:
C
C               T            T
C  MINIMIZE   A *X + (1/2)*X *AMAT*X
C
C  SUBJECT TO   BMAT*X =  B
C         AND   CMAT*X <= C
C
C   WHERE  AMAT IS N-BY-N, POSITIVE DEFINITE AND SYMMETRIC
C                         MATRIX
C          BMAT IS L-BY-N MATRIX
C          CMAT IS M-BY-N MATRIX
C          A IS AN N-VECTOR       B IS AN L-VECTOR
C          C IS AN M-VECTOR
C
C  THE UNKNOWNS ARE:
C
C   U   LAGRANGE MULTIPLIERS CORRESPONDING TO INEQUALITY
C        CONSTRAINTS
C   V   LAGRANGE MULTIPLIERS CORRESPONDING TO EQUALITY
C        CONSTRAINTS
C   W   DEFINED TO BE C - CMAT*X
C   X   KUHN-TUCKER POINT, WHICH SOLVES QUADRATIC
C        PROGRAMMING PROBLEM
C
C  THE INPUT DATA ARE:
C
C   A, B, C, AMAT, BMAT, CMAT   THE MATRICES AND VECTORS OF
C                               THE QUADRATIC PROGRAMMING
C                               PROBLEM STATED ABOVE.  AMAT
C                               AMAT MUST BE SYMMETRIC AND
C                               POSITIVE DEFINITE.
C
C   L, M, N                     THE SIZES OF THE MATRICES
C                               AND VECTORS
C
C  STORAGE MANAGEMENT:
C
C   THE INPUT AND OUTPUT VARIABLES, AND OTHER WORKING
C    STORAGE MUST BE PROPERLY DIMENSIONED ACCORDING TO THE
C    COMMENT ABOVE THE DECLARATION SECTION.  THE VALUES OF
C    THE VALUES OF L, M, AND N MUST CORRESPOND TO THESE
C    DEFINITIONS.
C
C  LIMITS AND TOLERANCES:
C
C   LIMIT   UPPER BOUND ON THE NUMBER OF LEMKE ITERATIONS
C            USED TO SOLVE THE LINEAR COMPLEMENTARITY
C            PROBLEM.  DEFAULT IS M + 5, WHERE M = NUMBER
C            OF INEQUALITY CONSTRAINTS.
C
```

```
C   EPS     CHOLESKY DECOMPOSITION OF AMAT CONTINUES AS LONG
C             AS LONG AS ALL DIAGONAL ELEMENTS ARE GREATER IN
C             ABSOLUTE VALUE THAN EPS.
C             DEFAULT VALUE IS 1.0D-12.
C
C   TOL     A VECTOR OF 3 SMALL POSITIVE NUMBERS USED IN
C             LEMKE AND CHOOSE, IN THE LINEAR COMPLE-
C             MENTARITY PROBLEM.  COMPUTATION CONTINUES
C             AS LONG AS:
C
C       (1)  Q-VECTOR (IN L.C. PROB.) >=0, FOR FEASIBILITY
C             OF SOLUTION.  COMPUTATIONALLY, AS LONG AS ALL
C             Q(I) >= -TOL(1)
C
C       (2)  ANY ELEMENT IN PIVOT COLUMN < -TOL(2).
C
C       (3)  PIVOT ELEMENTS ARE LARGE ENOUGH:  THE ABSOLUTE
C             VALUE OF THE PIVOT ELEMENT >= TOL(3)*INFINITY
C             NORM OF THE PIVOT COLUMN.
C
C            DEFAULT VALUES ARE TOL(1) = 1.0D-4,
C                               TOL(2) = 1.0D-6,
C                               TOL(3) = 1.0D-8.
C
C  ERROR INFORMATION:
C
C   IFLAG    INFORMATION ABOUT THE SUCCESS OR FAILURE OF THE
C             DEFINITE QUADRATIC PROGRAMMING PROBLEM SOLVER.
C
C             IFLAG =  0      SUCCESSFUL RETURN
C                   =  1- 2   FAILURE, INCONSISTENT INTEGER
C                              PARAMETERS, L, M, N, MP1, LMNP1
C                   = 11-18   FAILURE, ERRORS IN ROUTINE
C                              DECOMP
C                   = 21-25   FAILURE, ERRORS IN ROUTINE
C                              LEMKE
C                   = 31      FAILURE, ERRORS IN ROUTINE
C                              EVALU
C                   = 41      FAILURE, ERRORS IN ROUTINE
C                              PRINT
C
C
C  IN THIS PROGRAM, THE PROBLEM IS FIRST REDUCED BY USING
C   CHOLESKY'S AND QR DECOMPOSITION TO A LINEAR COMPLEMEN-
C   TARITY PROBLEM, WHICH IS THEN SOLVED BY LEMKE'S METHOD.
C
C  SUBROUTINE GEN IS CALLED TO GENERATE THE INPUT DATA.
C
C  SUBROUTINE PRINT IS CALLED TO PRINT OUT THE ORIGINAL
C   PROBLEM AND ITS SOLUTION.
C
C
C  MAIN CALLS SUBROUTINES GEN, DEFQP, PRINT, AND REPORT;
C
```

```
C        CALLS NO FUNCTION SUBROUTINES OR INTRINSIC FUNCTIONS
C
C
C   SHIH PING HAN
C   GREGORY D. SHUBERT
C   CORNELL UNIVERSITY
C   1978 APRIL
C
C**********
      INTEGER IFLAG, ITER, IWORK, L, LIMIT, LM2, LMNP1, M,
     1        MM2, MP1, N, NL, NM2
      LOGICAL EQCON, NQCON
      DOUBLE PRECISION A, ACOPY, AMAT, B, BCOPY, BMAT, C,
     1                 CMAT, DMAT, EMAT, EPS, GMAT, PI,
     2                 Q, RDIAG, TOL, TMAT, U, V, W, X
C
C   DIMENSIONS
C              A(N), ACOPY(N), AMAT(N,N), B(L), BCOPY(L)
C              BMAT(L,N), C(M), CMAT(M,N), DMAT(M,MP1),
C              EMAT(N,M), GMAT(N,L), IWORK(LMNP1), PI(L),
C              Q(M), RDIAG(L), TOL(3), TMAT(NL), U(M), V(L),
C              W(M), X(N)
C
      DIMENSION A(15), ACOPY(15), AMAT(15,15), B(5), BCOPY(5
     C ),
     1          BMAT(5,15), C(5), CMAT(5,15), DMAT(5,6),
     2          EMAT(15,5), GMAT(15,5), IWORK(26), PI(5),
     3          Q(5), RDIAG(5), TOL(3), TMAT(120), U(5), V(5
     C ),
     4          W(5), X(15)
      COMMON /SIZE/ N
      COMMON /LIMITS/ TOL, EPS, LIMIT
      L = 5
      M = 5
      N = 15
      NL = 120
      WRITE (6,1000) L, M, N
      LM2 = L - 2
      MM2 = M - 2
      NM2 = N - 2
      MP1 = M + 1
      LMNP1 = L + M + N + 1
C
C   SET TOLERANCES AND LIMITS
C
      LIMIT = M + 5
      EPS = 1.0D-12
      TOL(1) = 1.0D-4
      TOL(2) = 1.0D-6
      TOL(3) = 1.0D-8
C
C   CHECK FOR THE ABSENCE OF EQUALITY OR INEQUALITY
C    CONSTRAINTS
C
```

```
      EQCON = .TRUE.
      NQCON = .TRUE.
      IF (L .EQ. 0) EQCON = .FALSE.
      IF (M .EQ. 0) NQCON = .FALSE.
  100 CONTINUE
C
C  SET UP THE QUADRATIC PROGRAMMING PROBLEM
C
      CALL GEN(A,B,C,AMAT,BMAT,CMAT,EQCON,NQCON,L,M,N)
C
C  SOLVE THE QUADRATIC PROGRAMMING PROBLEM
C
      IF (.NOT. EQCON) L = 1
      IF (.NOT. NQCON) M = 1
      ITER = 0
      CALL DEFQP(A,B,C,AMAT,BMAT,CMAT,DMAT,ACOPY,BCOPY,Q,
     1           GMAT,EMAT,TMAT,U,V,W,X,PI,RDIAG,L,M,N,NL,
     2           MP1,LMNP1,IWORK,EQCON,NQCON,ITER,IFLAG)
      WRITE (6,1100) ITER
      WRITE (6,1200) IFLAG
      CALL REPORT(IFLAG)
      IF (IFLAG .NE. 0) GOTO 100
C
C  PRINT THE SOLUTION OF THE QUADRATIC PROGRAMMING PROBLEM
C
      CALL PRINT(A,B,C,U,V,W,X,AMAT,BMAT,CMAT,EQCON,NQCON,L,
     1           M,N,IFLAG)
      IF (IFLAG .NE. 0) CALL REPORT(IFLAG)
C
C  REPEAT RUN FOR A NEW PROBLEM
C
      GOTO 100
 1000 FORMAT (/32H NUMBER OF EQUALITY CONSTRAINTS:, 2X, I5,
     1        /34H NUMBER OF INEQUALITY CONSTRAINTS:, I5,
     2        /20H NUMBER OF UNKNOWNS:, 14X, I5/)
 1100 FORMAT (//30H NUMBER OF LEMKE ITERATIONS = ,I5)
 1200 FORMAT (9H IFLAG = ,I3)
C
C  LAST CARD OF MAIN PROCEDURE
C
      END
```

```
      SUBROUTINE GEN(A,B,C,AMAT,BMAT,CMAT,EQCON,NQCON,L,M,N)
      INTEGER L, M, N
      LOGICAL EQCON, NQCON
      DOUBLE PRECISION A, AMAT, B, BMAT, C, CMAT
      DIMENSION A(N), AMAT(N,N), B(L), BMAT(L,N), C(M),
     1          CMAT(M,N)
C**********
C
C  GEN
C
C   GENERATES THE MATRICES AND VECTORS DEFINING THE
C     QUADRATIC PROGRAMMING PROBLEM
C
C   THE USER WILL WANT TO SUPPLY HIS/HER OWN GEN ROUTINE,
C     WHICH WILL READ OR COMPUTE THE QUANTITIES A, B, C,
C     AMAT, BMAT, AND CMAT.
C
C
C  CALLS TO GEN:
C
C   GEN (A, B, C, AMAT, BMAT, CMAT, EQCON, NQCON, L, M, N)
C
C   WHERE
C
C     A, B,     VECTORS IN THE QUADRATIC PROGRAMMING PROBLEM
C     C           OF LENGTHS N, L, AND M, RESPECTIVELY.
C
C     AMAT,    N-BY-N VECTOR,
C     BMAT,    L-BY-N VECTOR,
C     CMAT,    M-BY-N VECTOR, IN THE QUADRATIC PROGRAMMING
C                               PROBLEM.
C
C     EQCON,   LOGICAL VARIABLES WHICH ARE TRUE IF THERE ARE
C     NQCON     EQUALITY CONSTRAINTS OR INEQUALITY CONSTRAINTS
C               RESPECTIVELY, AND ARE FALSE OTHERWISE.
C
C     L, M,    INTEGER VARIABLES INDICATING DIMENSIONS OF THE
C     N         VARIOUS MATRICES AND VECTORS.
C
C
C  GEN CALLS NO SUBROUTINES;
C
C       CALLS FUNCTION SUBROUTINE RANDOM;
C
C       CALLS INTRINSIC FUNCTION MINO.
C
C
C  GREGORY D. SHUBERT
C  CORNELL UNIVERSITY
C  1978 APRIL
C
C**********
      INTEGER I, IJ, IM1, IY, J, K
      INTEGER MINO
```

```
      OCUBLE PRECISION AHIGH, ALOW, SUM, ZERO
      OOUBLE PRECISION RANOCM
      OATA ZERO /0D0/
C
C  THE VECTORS A, B, C, ANO MATRICES BMAT, CMAT, ANO UMAT,
C   HERE CONTAIN POSITIVE ANO NEGATIVE NUMBERS OISTRIBUTEO
C   UNIFORMLY ON AN EXPONENTIAL SCALE BETWEEN ALOW ANO
C   AHIGH.
C
C  MATRIX AMAT IS COMPUTEO BY TAKING THE PROOUCT OF UMAT-
C   TRANSPOSE ANO UMAT, WHERE UMAT IS UPPER TRIANGULAR.
C
C  REAC SEEO FOR RANOOM NUMBER GENERATOR
C
      REAO (5,1000) IY
      WRITE (6,1100) IY
      IF (IY .EQ. 0) STOP
C
C  REAO UPPER, LOWER LIMITS FOR RANOOM NUMBER OISTRIBUTION
C
      REAO (5,1200) ALOW, AHIGH
      WRITE (6,1300) ALOW, AHIGH
      OO 120 I = 1, N
         OO 110 J = I, N
            AMAT(I,J) = RANOCM(IY,AHIGH,ALOW)
  110       CONTINUE
  120    CONTINUE
C
C  PRINT OUT OIAGONAL ELEMENTS OF UMAT
C
      WRITE (6,1400)
      WRITE (6,1500) (I, AMAT(I,I), I = 1, N)
C                   T
C  COMPUTE AMAT = UMAT *UMAT
C
      OC 160 I = 1, N
         IM1 = I - 1
         IF (I .EQ. 1) GOTO 150
         OO 140 J = 1, IM1
            SUM = ZERO
            IJ = MINO(I,J)
            OC 130 K = 1, IJ
               SUM = SUM + AMAT(K,I)*AMAT(K,J)
  130          CONTINUE
            AMAT(I,J) = SUM
  140       CONTINUE
  150    CONTINUE
  160    CONTINUE
      OC 180 I = 1, N
         SUM = ZERO
         OO 170 J = 1, I
            SUM = SUM + AMAT(J,I)**2
            AMAT(J,I) = AMAT(I,J)
  170       CONTINUE
```

```
         AMAT(I,I) = SUM
  180    CONTINUE
C
C  COMPUTE A, B, C, BMAT, CMAT
C
      DO 200 I = 1, N
         A(I) = RANDOM(IY,AHIGH,ALOW)
  200    CONTINUE
      IF (.NOT. EQCCN) GOTO 220
      DO 210 I = 1, L
         B(I) = RANDOM(IY,AHIGH,ALOW)
  210    CONTINUE
  220 CONTINUE
      IF (.NOT. NQCCN) GOTO 240
      DO 230 I = 1, M
         C(I) = RANDOM(IY,AHIGH,ALOW)
  230    CONTINUE
  240 CONTINUE
      IF (.NOT. EQCCN) GOTO 270
      DO 260 I = 1, L
         DO 250 J = 1, N
            BMAT(I,J) = RANDOM(IY,AHIGH,ALOW)
  250       CONTINUE
  260    CONTINUE
  270 CONTINUE
      IF (.NOT. NQCCN) GOTO 300
      DO 290 I = 1, M
         DO 280 J = 1, N
            CMAT(I,J) = RANDOM(IY,AHIGH,ALOW)
  280       CONTINUE
  290    CONTINUE
  300 CONTINUE
      RETURN
 1000 FORMAT (I12)
 1100 FORMAT (1H1/, 6H IY = , I12)
 1200 FORMAT (2D20.12)
 1300 FORMAT (18H  ALOW AND AHIGH: , 1PD12.4, 1H , 1PD12.4)
 1400 FORMAT (//26H DIAGONAL ELEMENTS OF UMAT,
     1          /13H             T,
     2          /18H AMAT = UMAT *UMAT/)
 1500 FORMAT (6(I8, 1PD12.4))
C
C  LAST CARD OF SUBROUTINE GEN
C
      END
```

```
      DOUBLE PRECISION FUNCTION RANDOM(IY,AHIGH,ALOW)
      INTEGER IY
      DOUBLE PRECISION AHIGH, ALOW
C**********
C
C  RANDCM
C
C   PRODUCES POSITIVE AND NEGATIVE NUMBERS X, SUCH THAT THE
C    NATURAL LOGARITHMS OF |X| ARE UNIFORMLY DISTRIBUTED
C    BETWEEN ALOW AND AHIGH.
C
C
C  RANDOM CALLS NO SUBROUTINES;
C
C          CALLS FUNCTION SUBROUTINE URAND;
C
C          CALLS INTRINSIC FUNCTIONS DEXP AND DSIGN.
C
C
C  GREGORY D. SHUBERT
C  CORNELL UNIVERSITY
C  1977 NOVEMBER
C
C**********
      DOUBLE PRECISION HALF
      DOUBLE PRECISION URAND
      DOUBLE PRECISION DEXP, DSIGN
      DATA HALF /5D-1/
      RANDOM = DSIGN(DEXP(URAND(IY)*(AHIGH - ALOW) + ALOW),
     1               URAND(IY) - HALF)
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION URAND(IY)
      INTEGER IY
C**********
C
C   URAND IS A UNIFORM RANDOM NUMBER GENERATOR BASED ON
C     THEORY AND SUGGESTIONS GIVEN IN D. E. KNUTH (1969),
C     VOL. 2.  THE INTEGER IY SHOULD BE INITIALIZED TO AN
C     ARBITRARY INTEGER PRIOR TO THE FIRST CALL TO URAND.
C     THE CALLING PROGRAM SHOULD NOT ALTER THE VALUE OF IY
C     BETWEEN SUBSEQUENT CALLS TO URAND.  VALUES OF URAND
C     WILL BE RETURNED IN THE INTERVAL (0,1).
C
C
C     FORSYTHE, MALCOLM, AND MOLER, 1977
C     COMPUTER METHODS FOR MATHEMATICAL COMPUTATIONS, PP. 246
C
C**********
      INTEGER IA, IC, ITWO, M, M2, MIC
      INTEGER IDINT
      DOUBLE PRECISION HALF, HALFM, S, SIX, THREE
      DOUBLE PRECISION DATAN, DFLOAT, DSQRT
      DATA M2 /0/, ITWO /2/, HALF /5D-1/, THREE /3D0/,
     1     SIX /6D0/
      IF (M2 .NE. 0) GOTO 20
C
C     IF FIRST ENTRY, COMPUTE MACHINE INTEGER WORD LENGTH
C
      M = 1
   10 M2 = M
      M = ITWO*M2
      IF (M .GT. M2) GOTO 10
      HALFM = M2
C
C     COMPUTE MULTIPLIER AND INCREMENT FOR LINEAR
C     CONGRUENTIAL METHOD
C
      IA = 8*IDINT(HALFM*DATAN(1.D0)/8.D0) + 5
      IC = 2*IDINT(HALFM*(HALF - DSQRT(THREE)/SIX)) + 1
      MIC = (M2 - IC) + M2
C
C     S IS THE SCALE FACTOR FOR CONVERTING TO FLOATING POINT
C
      S = HALF/HALFM
C
C     COMPUTE NEXT RANDOM NUMBER
C
   20 IY = IY*IA
C
C     THE FOLLOWING STATEMENT IS FOR COMPUTERS WHICH DO NOT
C     ALLOW INTEGER OVERFLOW ON ADDITION
C
      IF (IY .GT. MIC) IY = (IY - M2) - M2
C
      IY = IY + IC
```

```
C
C     THE FOLLOWING STATEMENT IS FOR COMPUTERS WHERE THE
C     WORD LENGTH FOR ADDITION IS GREATER THAN FOR
C     MULTIPLICATION
C
      IF (IY/2 .GT. M2) IY = (IY - M2) - M2
C
C     THE FOLLOWING STATEMENT IS FOR COMPUTERS WHERE INTEGER
C     OVERFLOW AFFECTS THE SIGN BIT
C
      IF (IY .LT. 0) IY = (IY + M2) + M2
      URAND = DFLOAT(IY)*S
      RETURN
      END
```

```
      SUBROUTINE PRINT(A,B,C,U,V,W,X,AMAT,BMAT,CMAT,EQCON,
     1                 NQCON,L,M,N,IFLAG)
      INTEGER L, M, N
      LOGICAL EQCON, NQCON
      DOUBLE PRECISION A, AMAT, B, BMAT, C, CMAT, U, V, W, X
      DIMENSION A(N), AMAT(N,N), B(L), BMAT(L,N), C(M),
     1     CMAT(M,N), U(M), V(L), W(M), X(N)
C**********
C
C   PRINT
C
C    PRINTS OUT THE SOLUTION TO THE QUADRATIC PROGRAMMING
C      PROBLEM
C
C
C   CALLS TO PRINT:
C
C    PRINT (A, B, C, U, V, W, X, AMAT, BMAT, CMAT, EQCON,
C           NQCON, L, M, N)
C
C    WHERE
C
C     A, B,     VECTORS IN THE QUADRATIC PROGRAMMING PROBLEM OF
C     C          LENGTHS N, L, AND M, RESPECTIVELY.
C
C     U,        M-VECTOR,    LAGRANGE MULTIPLIERS (INEQUALITY
C                             CONSTRAINTS)
C     V,        L-VECTOR,    LAGRANGE MULTIPLIERS (EQUALITY
C                             CONSTRAINTS)
C     W,        M-VECTOR,    C - CMAT*X
C     X         N-VECTOR     SOLUTION VECTOR, FROM QUADRATIC
C                             PROGRAMMING PROBLEM
C
C     AMAT,     N-BY-N MATRIX,
C     BMAT,     L-BY-N MATRIX,
C     CMAT      M-BY-N MATRIX, ALL IN THE QUADRATIC PROGRAMMING
C                              PROBLEM.
C
C     EQCON,    LOGICAL VARIABLES WHICH ARE TRUE IF THERE ARE
C     NQCON      EQUALITY CONSTRAINTS OR INEQUALITY CONSTRAINTS
C                 RESPECTIVELY, AND ARE FALSE OTHERWISE.
C
C     L, M,     INTEGER VARIABLES INDICATING DIMENSIONS OF THE
C     N          VARIOUS MATRICES AND VECTORS.
C
C     IFLAG     INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C                INFORMATION REGARDING THE SUCCESS OR FAILURE
C                OF PRINT.
C
C                IFLAG =  0  SUCCESSFUL RETURN
C                      = 41  NOT ENOUGH SPACE IN ACOPY, N > 100
C
C
C   PRINT CALLS NO SUBROUTINES, FUNCTION SUBROUTINES, OR
```

```
C          INTRIKSIC FUNCTIONS.
C
C
C  GREGORY C. SHUBERT
C  CORNELL UNIVERSITY
C  1978 APRIL
C
C**********
      INTEGER I, J
      OCUBLE PRECISION ATEMP(100), SUM
C
C  CHECK SPACE REQUIREMENTS, N <= 100
C
      IFLAG = 41
      IF (N .GT. 10C) RETURN
      IFLAG = 0
C
C  PRINT CUT A, B, C, AMAT, BMAT, CMAT
C
C     WRITE (6,1000)
C     WRITE (6,16C0) (I, A(I), I = 1, N)
C     IF (.NOT. EQCCN) GOTO 100
C     WRITE (6,11C0)
C     WRITE (6,16C0) (I, B(I), I = 1, L)
C 100 CCNTINUE
C     IF (.NOT. NCCCN) GOTO 110
C     WRITE (6,1200)
C     WRITE (6,1600) (I, C(I), I = 1, M)
C 110 CONTINUE
C     WRITE (6,1300)
C     WRITE (6,17C0) (I = 1, N)
C     OC 12C I = 1, N
C        WRITE (6,1800) I
C        WRITE (6,1900) (AMAT(I,J), J = 1, N)
C 120 CONTINUE
C     IF (.NCT. EQCON) GOTO 140
C     WRITE (6,1400)
C     WRITE (6,17C0) (I, I = 1, N)
C     OO 130 I = 1, L
C        WRITE (6,1800) I
C        WRITE (6,1900) (BMAT(I,J), J = 1, N)
C 130    CONTINUE
C 140 CONTINUE
C     IF (.NCT. NCCCN) GOTO 160
C     WRITE (6,1500)
C     WRITE (6,1700) (I, I = 1, N)
C     OO 150 I = 1, M
C        WRITE (6,1800) I
C        WRITE (6,1900) (CMAT(I,J), J = 1, N)
C 150    CONTINUE
C 160 CCNTINUE
C
C  PRINT THE SOLUTICN VECTORS U, V, W, X
C
```

```
      IF (.NOT. NQCCN) GCTO 170
      WRITE (6,2200)
      WRITE (6,1600) (I, U(I), I = 1, M)
  170 CONTINUE
      IF (.NCT. ECCCN) GCTO 180
      WRITE (6,2300)
      WRITE (6,1600) (I, V(I), I = 1, L)
  180 CONTINUE
      IF (.NCT. NCCON) GOTO 190
      WRITE (6,2400)
      WRITE (6,1600) (I, W(I), I = 1, M)
  190 CCNTINUE
      WRITE (6,2500)
      WRITE (6,1600) (I, X(I), I = 1, N)
C
C  CHECK THE RESULT BY KUHN-TUCKER CONOITION:
C
C                            T          T
C  1.   A + AMAT*X + BMAT *V + CMAT *U =  0
C  2.                        B - BMAT*X =  0
C  3.                   W = C - CMAT*X >= 0
C  4.           FOR ALL I,   U(I)*W(I) =  0
C  5.                                U >= 0
C
C  1ST CONOITION
C
      OC 250 I = 1, N
         SUM = A(I)
         OO 200 J = 1, N
            SUM = SUM + AMAT(I,J)*X(J)
  200       CCNTINUE
         IF (.NCT. EQCON) GOTO 220
         OO 210 J = 1, L
            SUM = SUM + BMAT(J,I)*V(J)
  210       CCNTINUE
  220    CONTINUE
         IF (.NOT. NQCON) GOTC 240
         OO 230 J = 1, M
            SUM = SUM + CMAT(J,I)*U(J)
  230       CCNTINUE
  240    CONTINUE
         ATEMP(I) = SUM
  250    CONTINUE
      WRITE (6,2600)
      WRITE (6,1600) (I, ATEMP(I), I = 1, N)
C
C  2NO CGNOITION
C
      IF (.NOT. EQCON) GOTO 280
      OO 270 I = 1, L
         SUM = B(I)
         OO 260 J = 1, N
            SUM = SUM - BMAT(I,J)*X(J)
  260       CONTINUE
```

```
         ATEMP(I) = SUM
  270    CONTINUE
      WRITE (6,2700)
      WRITE (6,1600) (I, ATEMP(I), I = 1, L)
C
C  3RO CCNOITION
C
  280 CCNTINUE
      IF (.NCT. NQCCN) GOTO 320
      OO 300 I = 1, M
         SUM = C(I)
         OO 290 J = 1, N
            SUM = SUM - CMAT(I,J)*X(J)
  290       CONTINUE
         ATEMP(I) = SUM
  300    CONTINUE
      WRITE (6,2800)
         WRITE (6,1600) (I, ATEMP(I), I = 1, M)
C
C  4TH CONOITION
C
      OC 310 I = 1, M
         ATEMP(I) = U(I)*W(I)
  310    CONTINUE
      WRITE (6,2900)
      WRITE (6,1600) (I, ATEMP(I), I = 1, M)
  320 CONTINUE
      RETURN
C1000 FCRMAT (///9H VECTCR A/)
C1100 FORMAT (///9H VECTOR B/)
C1200 FCRMAT (///9H VECTOR C/)
C1300 FCRMAT (///9H MATRIX A/)
C1400 FORMAT (///9H MATRIX B/)
C1100 FORMAT (///9H MATRIX C/)
 1600 FCRMAT (6(I8,1P012.4))
C1700 FCRMAT (10(I8,4X))
C1800 FORMAT (/5H RCW ,I3)
C1900 FORMAT (1P10012.4)
 2200 FORMAT (///38H LAGRANGE MULTIPLIERS CORRESPONOING TO,
     1         24H INEQUALITY CCNSTRAINTS:,
     2         26H VECTOR U (SHOULO BE >= 0)/)
 2300 FCRMAT (///38H LAGRANGE MULTIPLIERS CCRRESPCNOING TO,
     1         31H EQUALITY CONSTRAINTS: VECTOR V/)
 2400 FORMAT (///39H VECTOR W = C - CMAT*X (SHOULO BE >= 0)
     1         /)
 2500 FCRMAT (///28H KUHN-TUCKER POINT: VECTOR X/)
 2600 FCRMAT (///26H VERIFICATION OF SOLUTION:,
     1           /34X, 1HT, 9X, 1HT,
     2           /37H KT-CONOITION 1: A + AMAT*X + BMAT *V,
     3            14H + CMAT *U = 0/)
 2700 FORMAT (///32H KT-CONOITION 2:  B - BMAT*X = 0/)
 2800 FCRMAT (///37H KT-CONOITION 3:  W = C - CMAT*X >= 0/)
 2900 FORMAT (///31H KT-CONOITION 4:  U(I)*W(I) = 0/)
C
```

```
C  LAST CARD OF SUBROUTINE PRINT
C
      ENO
```

```
      SUBROUTINE REPORT(IFLAG)
      INTEGER IFLAG
C**********
C
C  REPORT
C
C   PRINTS APPROPRIATE RETURN MESSAGES
C
C
C  CALLS TO REPORT:
C
C   REPORT (IFLAG)
C
C   WHERE
C
C     IFLAG   INTEGER VARIABLE WHICH CONTAINS INFORMATION
C              REGARDING THE SUCCESS OR FAILURE OF OEFQP
C              OR PRINT.
C
C
C  REPORT CALLS NO SUBROUTINES; FUNCTION SUBROUTINES OR
C          INTRINSIC FUNCTICNS.
C
C
C  GREGORY O. SHUBERT
C  CORNELL UNIVERSITY
C  1978 APRIL
C
C**********
      INTEGER LIMIT
      DOUBLE PRECISION EPS, TOL(3)
      COMMON /LIMITS/ TOL, EPS, LIMIT
C
C  ISOLATE THE ROUTINE IN WHICH THE ERROR OCCURRED
C  DESCRIBE THE ERROR
C
      IF (IFLAG .NE. 0) GOTO 100
      WRITE (6,1000)
      RETURN
  100 CONTINUE
      IF (IFLAG .NE. 1) GOTO 110
      WRITE (6,2000)
      WRITE (6,1010)
      WRITE (6,1011)
      RETURN
  110 CONTINUE
      IF (IFLAG .NE. 2) GOTO 120
      WRITE (6,2000)
      WRITE (6,1020)
      WRITE (6,1011)
      RETURN
  120 CONTINUE
      IF (IFLAG .NE. 11) GOTO 130
      WRITE (6,2100)
```

```
      WRITE (6,1010)
      WRITE (6,1011)
      RETURN
  130 CONTINUE
      IF (IFLAG .NE. 12) GOTO 140
      WRITE (6,2100)
      WRITE (6,1120)
      WRITE (6,1011)
      RETURN
  140 CONTINUE
      IF (IFLAG .NE. 13) GOTO 150
      WRITE (6,2200)
      WRITE (6,1120)
      WRITE (6,1011)
      RETURN
  150 CONTINUE
      IF (IFLAG .NE. 14) GOTO 160
      WRITE (6,2200)
      WRITE (6,1140)
      WRITE (6,1141) EPS
      RETURN
  160 CONTINUE
      IF (IFLAG .NE. 15) GOTO 170
      WRITE (6,2300)
      WRITE (6,1120)
      WRITE (6,1011)
      RETURN
  170 CONTINUE
      IF (IFLAG .NE. 16) GOTO 180
      WRITE (6,2300)
      WRITE (6,1160)
      RETURN
  180 CONTINUE
      IF (IFLAG .NE. 17) GOTO 190
      WRITE (6,2300)
      WRITE (6,1170)
      RETURN
  190 CONTINUE
      IF (IFLAG .NE. 18) GOTO 200
      WRITE (6,2400)
      WRITE (6,1180)
      RETURN
  200 CONTINUE
      IF (IFLAG .NE. 21) GOTO 210
      WRITE (6,2500)
      WRITE (6,1210)
      WRITE (6,1011)
      RETURN
  210 CONTINUE
      IF (IFLAG .NE. 22) GOTO 220
      WRITE (6,2500)
      WRITE (6,1220) LIMIT
      RETURN
  220 CONTINUE
```

```
      IF (IFLAG .NE. 23) GOTO 230
      WRITE (6,2600)
      WRITE (6,1230) TOL(1)
      WRITE (6,1231)
      RETURN
  230 CONTINUE
      IF (IFLAG .NE. 24) GOTO 240
      WRITE (6,2600)
      WRITE (6,1240) TOL(2)
      WRITE (6,1231)
      RETURN
  240 CONTINUE
      IF (IFLAG .NE. 25) GOTO 250
      WRITE (6,2600)
      WRITE (6,1250) TOL(3)
      WRITE (6,1231)
      RETURN
  250 CONTINUE
      IF (IFLAG .NE. 31) GOTO 260
      WRITE (6,2700)
      WRITE (6,1120)
      WRITE (6,1011)
      RETURN
  260 CONTINUE
      IF (IFLAG .NE. 41) GOTO 270
      WRITE (6,2800)
      WRITE (6,1410)
      RETURN
  270 CONTINUE
      WRITE (6,2900)
      RETURN
 1000 FORMAT (14H NORMAL RETURN/)
 1010 FORMAT (24H MP1 NOT EQUAL TO M + 1.)
 1011 FORMAT (38H CHECK ASSIGNMENTS TO THESE VARIABLES.)
 1020 FORMAT (34H LMNP1 NOT EQUAL TO L + M + N + 1.)
 1120 FORMAT (30H NL NOT EQUAL TO (N**2 + N)/2.)
 1140 FORMAT (39H DIAGONAL ELEMENT OF MATRIX AMAT < EPS.,
     1        /42H  PROBABLE CAUSE: MATRIX AMAT NOT POSITIVE,
     2        /33H  DEFINITE WITHIN ROUNDING ERROR.)
 1141 FORMAT (7H EPS = , 1PD12.5)
 1160 FORMAT (36H M < 1.  M = COLUMNS IN MATRIX XMAT.,
     1        /19H  CHECK DIMENSIONS.)
 1170 FORMAT (37H DIAGONAL ELEMENT OF LMAT EQUAL TO 0.,
     1        /35H  PROBABLE CAUSE: AMAT NOT POSITIVE,
     2         10H DEFINITE.)
 1180 FORMAT (37H MATRIX GMAT PROBABLY RANK DEFICIENT.,
     1        /42H  PROBABLE CAUSE: EQUALITY CONSTRAINTS NOT,
     2         22H LINEARLY INDEPENDENT.)
 1210 FORMAT (24H NP1 NOT EQUAL TO N + 1.)
 1220 FORMAT (37H NUMBER OF LEMKE ITERATIONS > LIMIT, , I5)
 1230 FORMAT (39H Q(I) < -TOL(1) FOR SOME I;   TOL(1) = ,
     1         1PD12.5)
 1231 FORMAT (40H LINEAR COMPLEMENTARITY PROBLEM POSSIBLY,
     1         17H HAS NO SOLUTION.)
```

```
 1240 FORMAT (42H PIVOT COLUMN TOC LARGE:   Y(I) >= -TOL(2),
     1         11H FOR ALL I:, /11H  TOL(2) = , 1PO12.5)
 1250 FORMAT (40H PIVOT TOO SMALL;   |Y(PIVOT)| < TOL(3)*,
     1         18HINFINITY NCRM OF Y, /11H  TOL(3) = ,
     2         1PO12.5)
 1410 FORMAT (39H NUMBER OF UNKNCWNS > SIZE OF WORKSPACE,
     1          6H ATEMP, /20H  REOIMENSION ATEMP.)
 2000 FORMAT (/16H ERROR IN OEFCP.)
 2100 FORMAT (/17H ERROR IN OECOMP.)
 2200 FORMAT (/17H ERROR IN CHCLES.)
 2300 FORMAT (/17H ERROR IN FCRWAR.)
 2400 FORMAT (/13H ERROR IN QR.)
 2500 FORMAT (/16H ERROR IN LEMKE.)
 2600 FORMAT (/17H ERROR IN CHCOSE.)
 2700 FORMAT (/16H ERROR IN EVALU.)
 2800 FORMAT (/16H ERROR IN PRINT.)
 2900 FORMAT (/17H UNOEFINEO ERROR.)
C
C  LAST CARO OF SUBROUTINE REPORT
C
      END
```

APPENDIX 2

Definite quadratic program solver

DEFQP

DECOMP

CHOLES

FORWAR

QR

INDEX

LEMKE

CHOOSE

PIVOT

EVALU

```
      SUBROUTINE CEFQP(A,B,C,AMAT,BMAT,CMAT,OMAT,ACOPY,
     1                   BCOPY,Q,GMAT,EMAT,TMAT,U,V,W,X,PI,
     2                   RCIAG,L,M,N,NL,MP1,LMNP1,IWORK,
     3                   EQCON,NQCON,ITER,IFLAG)
      INTEGER IFLAG, ITER, IWORK, L, LMNP1, M, MP1, N, NL
      LOGICAL EQCCN, NCCON
      OCUBLE PRECISION A, ACOPY, AMAT, B, BCOPY, BMAT, C,
     1                   CMAT, CMAT, EMAT, GMAT, PI, Q,
     2                   ROIAG, TMAT, U, V, W, X
      OIMENSION A(N), ACOPY(N), AMAT(N,N), B(L), BCOPY(L),
     1          BMAT(L,N), C(M), CMAT(M,N), OMAT(M,MP1),
     2          EMAT(N,M), GMAT(N,L), IWORK(LMNP1),
     3          PI(L), Q(M), ROIAG(L), TMAT(NL), U(M), V(L),
     4          W(M), X(N)
C**********
C
C  CEFQP
C
C   SOLVES THE QUAORATIC PRCGRAMMING PROBLEM
C
C                  T              T
C     MINIMIZE   A *X + (1/2)*X *AMAT*X
C
C     SUBJECT TO   BMAT*X =  B
C                  CMAT*X <= C
C
C  IN THIS PROGRAM, THE PROBLEM IS FIRST REOUCEO BY USING
C   CHOLESKY'S ANC QR OECOMPCSITICN TO A LINEAR CCMPLEMEN-
C   TARITY PROBLEM, WHICH IS THEN SOLVEO BY LEMKE'S METHOO.
C
C  THE LINEAR CCMPLEMENTARITY PROBLEM IS OF THE FORM:
C
C    W = Q + OMAT*U
C                                     T
C    SUBJECT TO   W >=0,  U >= 0,  W *U = 0
C
C
C  INPUT PARAMETERS ARE A, B, C, AMAT, BMAT, CMAT, L, M, N,
C   NL, MP1, LMNP1, EQCON, NEQCON.
C
C  WORKING SPACE PARAMETERS ARE OMAT, ACOPY, BCOPY, Q, GMAT,
C   EMAT, PI, ROIAG, TMAT, IWORK.
C
C  OUTPUT PARAMETERS ARE U, V, W, X, ITER, IFLAG.
C
C
C  CALLS TO OEFQP:
C
C   OEFQP (A, B, C, AMAT, BMAT, CMAT, OMAT, ACOPY, BCOPY, Q,
C          GMAT, EMAT, TMAT, U, V, W, X, PI, ROIAG, L, M, N,
C          NL, MP1, LMNP1, IWORK, EQCON, NQCON, ITER, IFLAG)
C
C   WHERE
C
```

```
C     A,B,     VECTCRS IN THE QUAORATIC PROGRAMMING PRCBLEM OF .
C     C         LENGTHS N, L, ANO M, RESPECTIVELY.
C
C     BMAT,    L-BY-N MATRIX,
C     CMAT,    M-BY-N MATRIX,
C     OMAT,    M-BY-MP1 MATRIX,
C     AMAT     N-BY-N MATRIX, IN THE QUAORATIC PROGRAMMING
C                               PROBLEM
C
C     ACOPY,   COPIES OF VECTORS A, B, ANO C, RESPECTIVELY.
C     BCCPY,    ON CUTPUT A, B, ANO C ARE UNCHANGED, BUT THE
C     Q         SPACE USEO BY THESE COPIES MAY BE USEO
C               ELSEWHERE.
C                                         T        T
C     GMAT,    COPIES OF MATRICES BMAT , CMAT , ANO THE LOWER
C     EMAT,     TRIANGLE OF AMAT, RESPECTIVELY.  ON OUTPUT,
C     TMAT      BMAT, CMAT, ANO AMAT ARE UNCHANGEO, BUT THE
C               SPACE USEO BY THESE COPIES MAY BE USEO
C               ELSEWHERE.
C
C     U,       M-VECTOR,     LAGRANGE MULTIPLIERS (INEQUALITY
C                             CONSTRAINTS)
C     V,       L-VECTCR,     LAGRANGE MULTIPLIERS (EQUALITY
C                             CCNSTRAINTS)
C     W,       M-VECTCR,     C - CMAT*X
C     X        N-VECTOR.     SCLUTICN VECTOR
C
C     PI,      TWC L-VECTORS USEO IN SUBROUTINES OECOMP ANO
C               EVALU
C     ROIAG
C
C     L, M,    INTEGER VARIABLES INOICATING OIMENSICNS OF THE
C     N, MP1    VARIOUS MATRICES ANO VECTORS.  MP1 = M + 1.
C     LMNP1     LMNP1 = L + M + N + 1.
C
C     IWORK    INTEGER LMNP1-VECTOR USEO AS SCRATCH SPACE.
C
C     EQCON,   LOGICAL VARIABLES WHICH ARE TRUE IF THERE ARE
C     NQCON     EQUALITY CONSTRAINTS OR INEQUALITY CONSTRAINTS
C               RESPECTIVELY, ANO ARE FALSE OTHERWISE.
C
C     ITER     INTEGER VARIABLE.  ON OUTPUT IT GIVES THE
C               NUMBER OF LEMKE ITERATIONS PERFORMEO.
C
C     IFLAG    INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C               INFORMATION REGAROING THE SUCCESS OR FAILURE
C               OF CEFQP.
C
C               IFLAG =  0      SUCCESSFUL RETURN
C                     =  1- 2  FAILURE, INCONSISTENT INTEGER
C                                PARAMTERS L, M, N, MP1, LMNP1
C                     = 11-18  FAILURE, ERRORS IN ROUTINE
C                                DECOMP
C                     = 21-25  FAILURE, ERRORS IN ROUTINE
```

```
C                             LEMKE
C                    = 31    FAILURE, ERRORS IN RCUTINE
C                             EVALU
C
C
C  OTHER VARIABLES USED IN DEFQP:
C
C   LIMIT   UPPER BOUND ON THE NUMBER OF LEMKE ITERATIONS
C            USED TO SOLVE THE LINEAR CCMPLEMENTARITY
C            PROBLEM.
C
C   EPS     CHOLESKY DECCMPCSITION OF AMAT CONTINUES AS LONG
C            AS ALL DIAGONAL ELEMENTS ARE GREATER IN
C            ABSOLUTE VALUE THAN EPS.
C
C   TOL     A VECTOR OF 3 SMALL POSITIVE NUMBERS USED IN
C            LEMKE AND CHOOSE, IN THE LINEAR COMPLEMENTARITY
C            PRCBLEM.  COMPUTATION CONTINUES AS LCNG AS:
C
C       (1)  Q-VECTOR (IN L.C. PRCB.) >=0, FOR FEASIBILITY
C             OF SOLUTION.  COMPUTATICNALLY, AS LONG AS ALL
C             Q(I) >= -TCL(1)
C
C       (2)  ANY ELEMENT IN PIVOT COLUMN < -TOL(2).
C
C       (3)  PIVOT ELEMENTS ARE LARGE ENOUGH:  THE ABSOLUTE
C             VALUE OF THE PIVOT ELEMENT >= TOL(3)*INFINITY
C             NCRM CF THE PIVOT COLUMN.
C
C
C  DEFQP CALLS SUBROUTINES CECOMP, LEMKE, AND EVALU;
C
C        CALLS FUNCTION SUBRCUTINE INDEX;
C
C        CALLS NO INTRINSIC FUNCTIONS.
C
C
C  SHIH PING HAN
C  GREGORY D. SHUBERT
C  CORNELL UNIVERSITY
C  1977 NOVEMBER
C
C**********
      INTEGER I, I1, J, LIMIT, LPN
      INTEGER INDEX
      DOUBLE PRECISION EPS, CNE, TOL(3)
      COMMON /LIMITS/ TOL, EPS, LIMIT
      DATA CNE /1D0/
C
C  CHECK FOR ABSENCE OF EQUALITY CR INEQUALITY CONSTRAINTS
C
      IF (.NCT. ECCON) L = 0
      IF (.NCT. NCCON) M = 0
      LPN = L + N
```

```
C
C  CHECK INTEGER INPUT PARAMETERS
C
      IFLAG = 1
      IF (MP1 .NE. M + 1) RETURN
      IFLAG = 2
      IF (LMNP1 .NE. L + M + N + 1) RETURN
C
C  COPY MATRICES AND VECTORS
C
C  COPY AMAT INTO TMAT (LOWER TRIANGLE ONLY)
C
      DO 110 I = 1, N
         DO 100 J = 1, I
            I1 = INDEX(I,J)
            TMAT(I1) = AMAT(I,J)
 100        CONTINUE
 110     CONTINUE
C           T
C  COPY BMAT  INTO GMAT.
C
      IF (.NOT. EQCON) GOTO 140
      DO 130 I = 1, N
         DO 120 J = 1, L
            GMAT(I,J) = BMAT(J,I)
 120        CONTINUE
 130     CONTINUE
 140  CONTINUE
C           T
C  COPY CMAT  INTO EMAT.
C
      IF (.NOT. NQCON) GOTO 170
      DO 160 I = 1, N
         DO 150 J = 1, M
            EMAT(I,J) = CMAT(J,I)
 150        CONTINUE
 160     CONTINUE
 170  CONTINUE
C
C  COPY A INTO ACOPY
C
      DO 180 I = 1, N
         ACOPY(I) = A(I)
 180     CONTINUE
C
C  COPY B INTO BCOPY
C
      IF (.NOT. EQCON) GOTO 200
      DO 190 I = 1, L
         BCOPY(I) = B(I)
 190     CONTINUE
 200  CONTINUE
C
C  COPY C INTO C
```

```
C
      IF (.NCT. NCCCN) GOTO 220
      00 210 I = 1, M
         Q(I) = C(I)
  210    CONTINUE
  220 CONTINUE
C
C  OECOMPOSE THE VARIOUS MATRICES TC CONVERT TO THE LINEAR
C   CCMPLEMENTARITY PROBLEM:  W = Q + OMAT*U
C
      IF (.NCT. ECCCN) L = 1
      IF (.NOT. NQCON) M = 1
      CALL CECOMP(TMAT,GMAT,EMAT,ACOPY,BCOPY,Q,OMAT,PI,
     1            ROIAG,L,M,MP1,N,NL,EQCON,NQCON,EPS,IFLAG)
      IF (IFLAG .NE. 0) RETURN
C
C  FILL LAST (MP1-ST) COLUMN OF OMAT WITH 1'S
C
      ITER = 0
      IF (.NCT. NCCCN) GCTO 310
      00 300 I = 1, M
         OMAT(I,MP1) = ONE
  300 CONTINUE
C
C  SOLVE LINEAR CCMPLEMENTARITY PROBLEM USING LEMKE'S METHOO
C
      CALL LEMKE(OMAT,Q,W,U,M,MP1,TCL,LIMIT,IWORK(1),
     1           IWCRK(MP1),ITER,IFLAG)
      IF (IFLAG .NE. 0) RETURN
  310 CONTINUE
C
C  USE THE SCLUTICN U TO FINO X ANO V.
C
      IF (.NCT. ECCCN) L = 1
      IF (.NOT. NQCCN) M = 1
      CALL EVALU(TMAT,GMAT,PI,ROIAG,EMAT,BCOPY,ACOPY,U,
     1           V,X,L,M,N,NL,EQCON,NQCON,IFLAG)
      RETURN
C
C  LAST CARO OF SUBROUTINE OEFQP
C
      END
```

```
      SUBROUTINE CECCMP(TMAT,GMAT,EMAT,A,B,Q,OMAT,PI,ROIAG,
     1                  L, M, MP1,N,NL,EQCON,NQCON,EPS,
     2                  IFLAG)
      INTEGER IFLAG, L, M, MP1, N, NL
      LOGICAL ECCCN, NQCCN
      OOUBLE PRECISION A, B, OMAT, EMAT, EPS, GMAT, PI,
     1                 Q, ROIAG, TMAT
      OIMENSION A(N), B(L), OMAT(M,MP1), EMAT(N,M),
     1          GMAT(N,L), PI(L), Q(M), ROIAG(L), TMAT(NL)
C**********
C
C  OECCMP
C
C   TRANSFORMS THE CUAORATIC PROGRAMMING PROBLEM INTO A
C   LINEAR CCMPLEMENTARITY PROBLEM:  W = Q + OMAT*U
C                                               T
C     SUBJECT TO:  W >= 0   U >= 0   W U = 0
C
C
C  CALLS TO CECCMP:
C
C   OECOMP (TMAT, GMAT, EMAT, A, B, Q, OMAT, PI, ROIAG, L,
C           M, MP1, N, NL, ECCON, NQCON, EPS, IFLAG)
C
C   WHERE
C
C     TMAT    THE NON-ZERC HALF OF AN N-BY-N LOWER TRIANGULAR
C              MATRIX STOREO AS AN NL-VECTOR.  ON INPUT TMAT
C              CONTAINS THE LOWER TRIANGLE OF SYMMETRIC
C              MATRIX AMAT.  CN OUTPUT IT CONTAINS TMAT,
C              WHERE
C                        T
C              TMAT*TMAT  = AMAT
C                                                        T
C     GMAT    AN N-BY-L MATRIX.  ON INPUT IT CCNTAINS BMAT .
C              ON OUTPUT IT CCNTAINS MATRIX R IN ITS UPPER
C              TRIANGLE ANO THE REFLECTORS Q(K) OF MATRIX Q
C              BELCW.  Q*GMAT = R.
C
C                                                        T
C     EMAT    AN N-BY-M MATRIX.  ON INPUT IT CONTAINS CMAT .
C              ON OUTPUT IT CCNTAINS ZMAT1 AND ZMAT2 (L ANO
C              N-L RCWS RESPECTIVELY).
C
C     A       AN N-VECTOR.  CN INPUT IT CONTAINS THE A-VECTOR
C              FROM THE QUACRATIC PROGRAMMING PROBLEM.  CN
C              OUTPUT ITS FIRST L ELEMENTS ARE Z1 ANO THE
C              REMAINING N-L ELEMENTS ARE Z2.
C
C     B       AN L-VECTOR.  ON INPUT IT CONTAINS THE B-VECTOR
C              FROM THE QUADRATIC PROGRAMMING PROBLEM.  CN
C              OUTPUT IT CONTAINS VECTOR O USEO LATER IN
C              CALCULATING X.
C
```

```
C     Q        AN M-VECTOR.  CN INPUT IT CONTAINS THE C-VECTOR
C               FROM THE QUADRATIC PROGRAMMING PROBLEM.  CN
C               OUTPUT IT CONTAINS THE VECTOR Q USED IN THE
C               LINEAR COMPLEMENTARITY PROBLEM.
C
C     OMAT     AN M-BY-MP1 MATRIX.  ON OUTPUT IT CONTAINS THE
C               MATRIX OMAT USED IN THE LINEAR COMPLEMENTARITY
C               PROBLEM.
C
C     PI       AN L-VECTOR CONTAINING SCALARS WHICH OCCUR IN
C               THE REFLECTORS Q(K).
C
C     RDIAG    AN L-VECTOR CONTAINING THE DIAGONAL ELEMENTS OF
C               THE R MATRIX (THE REST OF WHICH IS IN GMAT).
C
C     L,M,     INTEGER VARIABLES GIVING THE SIZES OF THE
C     MP1,      VARIOUS MATRICES AND VECTORS.
C     M,NL      NL = (N**2 + N)/2.  MP1 = M + 1.
C
C     EQCON,   LOGICAL VARIABLES, WHICH ARE TRUE IF THERE ARE
C               EQUALITY OR INEQUALITY CONSTRAINTS,
C               RESPECTIVELY, AND FALSE OTHERWISE.
C
C     EPS      VARIABLE CONTAINING A SMALL POSITIVE NUMBER
C               WHICH IS USED IN SUBROUTINES CHOLES AND QR AS
C               A LOWER BOUND ON THE SIZE OF ELEMENTS IN
C               MATRICES BEING MANIPULATED.
C
C     IFLAG    INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C               INFORMATION REGARDING THE SUCCESS OR FAILURE
C               OF DECOMP.
C
C               IFLAG =  0     SUCCESSFUL RETURN
C                     = 11,12  FAILURE, INCONSISTENT INTEGER
C                                PARAMETERS N, NL, M, MP1
C                     = 13,14  FAILURE, ERRORS IN ROUTINE
C                                CHOLES
C                     = 15-17  FAILURE, ERRORS IN ROUTINE
C                                FORWAR
C                     = 18     FAILURE, ERRORS IN ROUTINE QR
C
C
C  DECOMP CALLS SUBROUTINES CHOLES, FORWAR, AND QR;
C
C         CALLS NO FUNCTION SUBROUTINES;
C
C         CALLS INTRINSIC FUNCTION MINO.
C
C
C  SHIH PING HAN
C  GREGORY C. SHUBERT
C  CORNELL UNIVERSITY
C  1977 NOVEMBER
C
```

```
C**********
      INTEGER I, IM, J, JM1, K, LP1
      INTEGER MINO
      DOUBLE PRECISION SUM, ZERO
      DATA ZERO /0D0/
C
C  CHECK FOR ABSENCE OF EQUALITY OR INEQUALITY CONSTRAINTS
C
      IF (.NOT. EQCON) L = 0
      IF (.NOT. NQCON) M = 0
C
C  CHECK INTEGER INPUT PARAMETERS
C
      IFLAG = 11
      IF (MP1 .NE. M + 1) RETURN
      IFLAG = 12
      IF (NL .NE. (N**2 + N)/2) RETURN
      IFLAG = 0
C
C  FORM THE CHOLESKY DECOMPOSITION OF AMAT (STORED IN TMAT):
C                         T
C   AMAT = TMAT*TMAT .  OVERWRITE TMAT ONTO AMAT.
C
      CALL CHOLES(TMAT,NL,N,EPS,IFLAG)
      IF (IFLAG .NE. 0) RETURN
C                                        T
C  SOLVE FOR GMAT:  TMAT*GMAT = BMAT , OVERWRITE GMAT ONTO
C      T
C   BMAT.
C                                        T
C  SOLVE FOR EMAT:  TMAT*EMAT = CMAT , OVERWRITE EMAT ONTO
C      T
C   CMAT.
C
C  SOLVE FOR E:  TMAT*E = A, OVERWRITE E ONTO A.
C
      IF (.NOT. EQCON) GOTO 100
      CALL FORWAR(TMAT,GMAT,NL,N,L,IFLAG)
      IF (IFLAG .NE. 0) RETURN
  100 CONTINUE
      IF (.NOT. NQCON) GOTO 110
      CALL FORWAR(TMAT,EMAT,NL,N,M,IFLAG)
      IF (IFLAG .NE. 0) RETURN
  110 CONTINUE
      CALL FORWAR(TMAT,A,NL,N,1,IFLAG)
      IF (IFLAG .NE. 0) RETURN
C
C  FORM THE QR DECOMPOSITION OF GMAT.  OVERWRITE QMAT, R
C   ONTO GMAT, AND STORE THE DIAGONAL ELEMENTS OF R IN
C   RDIAG.
C
      IF (.NOT. EQCON) GOTO 230
      CALL QR(GMAT,PI,RDIAG,N,L,EPS,IFLAG)
      IF (IFLAG .NE. 0) RETURN
```

```
C
C           | ZMAT1 |
C  COMPUTE  |-------| = QMAT*EMAT, WHERE ZMAT1 HAS L ROWS AND
C           | ZMAT2 |                       ZMAT2 HAS N-L ROWS.
C
C   OVERWRITE ZMAT1 AND ZMAT2 ONTO EMAT.
C
C           | Z1 |
C  COMPUTE  |----| = QMAT*E, WHERE Z1 HAS L ELEMENTS AND
C           | Z2 |                   Z2 HAS N-L ELEMENTS.
C
C   OVERWRITE Z1 AND Z2 ONTO E.
C
C  QMAT = Q(IM)*Q(IM-1)* ... *Q(1), WHERE IM = MIN(N-1,L).
C                          T
C   Q(K) = I - V(K)*V(K) /PI(K)
C                              T
C   Q(K)*X = X - V(K)*(V(K) *X)/PI(K)
C
C   AND THE NON-ZERO ELEMENTS (K, K+1, ... N) OF V(K) ARE
C   STORED IN GMAT(K,K) THROUGH GMAT(N,K).
C
      IM = MIN0(N-1,L)
      IF (IM .EQ. 0) GOTO 190
      DO 180 K = 1, IM
C
C  IF PI(K) = 0, THERE IS NO REFLECTION
C
         IF (PI(K) .EQ. ZERO) GOTO 150
C
C  FOR EACH COLUMN X OF EMAT AND FOR E, COMPUTE
C             T
C   SUM = V(K) *X/PI(K)
C
         IF (.NOT. NQCON) GOTO 150
         DO 140 J = 1, M
            SUM = ZERO
            DO 120 I = K, N
               SUM = SUM + GMAT(I,K)*EMAT(I,J)
 120           CONTINUE
            SUM = SUM/PI(K)
C
C  COMPLETE THE UPDATE OF EACH X:  X = X - V(K)*SUM
C
            DO 130 I = K, N
               EMAT(I,J) = EMAT(I,J) - GMAT(I,K)*SUM
 130           CONTINUE
 140        CONTINUE
 150     CONTINUE
         SUM = ZERO
         DO 160 J = K, N
            SUM = SUM + GMAT(J,K)*A(J)
 160        CONTINUE
         SUM = SUM/PI(K)
```

```
         DO 170 J = K, N
            A(J) = A(J) - GMAT(J,K)*SUM
  170       CONTINUE
  180    CONTINUE
  190 CONTINUE
C                          T
C  SOLVE FOR D:  RMAT *D = B.  OVERWRITE D ONTO B.
C        T
C   RMAT (J,I) IS STORED IN GMAT(I,J).
C   RMAT(I,I) IS STORED IN RDIAG(I).
C
      IFLAG = 17
      DO 220 J = 1, L
         JM1 = J - 1
         IF (JM1 .EQ. 0) GOTO 210
         SUM = ZERO
         DO 200 I = 1, JM1
            SUM = SUM + GMAT(I,J)*B(I)
  200       CONTINUE
         B(J) = B(J) - SUM
  210    CONTINUE
         IF (RDIAG(J) .EQ. ZERO) RETURN
         B(J) = B(J)/RDIAG(J)
  220    CONTINUE
      IFLAG = 0
C                            T
C  COMPUTE DMAT = ZMAT2 *ZMAT2
C
C   ZMAT2(I,J) IS STORED IN EMAT(L+I,J)
C
  230 CONTINUE
      IF (.NOT. NCCON) GOTO 330
      LP1 = L + 1
      IF (N .EQ. L) GOTO 270
      DO 260 I = 1, M
         DO 250 J = 1, M
            SUM = ZERO
            DO 240 K = LP1, N
               SUM = SUM + EMAT(K,I)*EMAT(K,J)
  240          CONTINUE
            DMAT(I,J) = SUM
            DMAT(J,I) = SUM
  250       CONTINUE
  260    CONTINUE
  270 CONTINUE
C                          T            T
C  COMPUTE Q = C - ZMAT1 *D + ZMAT2 *Z2. OVERWRITE Q ONTO C.
C
      IF (.NOT. ECCON) GOTO 300
      DO 290 I = 1, M
         SUM = ZERO
         DO 280 J = 1, L
            SUM = SUM + EMAT(J,I)*B(J)
  280       CONTINUE
```

```
         Q(I) = Q(I) - SUM
  290    CONTINUE
  300 CONTINUE
      IF (N .EQ. L) GOTO 330
      DO 320 I = 1, M
         SUM = ZERO
         DO 310 J = LP1, N
            SUM = SUM + EMAT(J,I)*A(J)
  310       CONTINUE
         Q(I) = Q(I) + SUM
  320    CONTINUE
  330 CONTINUE
      RETURN
C
C  LAST CARD OF SUBROUTINE DECOMP
C
      END
```

```
      SUBROUTINE CHOLES(AMAT,NL,N,EPS,IFLAG)
      INTEGER IFLAG, N, NL
      DOUBLE PRECISION EPS, AMAT
      DIMENSION AMAT(NL)
C**********
C
C  CHOLES
C
C   PERFORMS A CHOLESKY DECOMPOSITION, IN PLACE, OF A
C    SYMMETRIC POSITIVE-DEFINITE MATRIX AMAT, INTO TMAT,
C    SUCH THAT
C                    T
C     AMAT = TMAT*TMAT
C
C
C  CALLS TO CHOLES:
C
C   CHOLES (AMAT, NL, N, EPS, IFLAG)
C
C   WHERE
C
C    AMAT     THE LOWER HALF OF THE N-BY-N MATRIX AMAT,
C              STORED AS AN NL-VECTOR.  ON OUTPUT IT CONTAINS
C              THE MATRIX TMAT, SIMILARLY STORED AS AN
C              NL-VECTOR.
C
C    NL       INTEGER VARIABLE GIVING THE SIZE OF AMAT.
C              NL = (N**2 + N)/2
C
C    N        INTEGER VARIABLE GIVING THE ORIGINAL SIZE OF
C              MATRIX AMAT.
C
C    EPS      VARIABLE CONTAINING A SMALL POSITIVE NUMBER
C              USED AS A LOWER BOUND ON THE ABSOLUTE VALUE OF
C              THE DIAGONAL ELEMENTS OF TMAT.
C
C    IFLAG    INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C              INFORMATION REGARDING THE SUCCESS OR FAILURE
C              OF CHOLES.
C
C              IFLAG =  0    SUCCESSFUL RETURN
C                    = 13    FAILURE, INCONSISTENT INTEGER
C                             PARAMTERS NL, N
C                    = 14    FAILURE, DIAGONAL ELEMENT OF
C                             TMAT < EPS
C
C
C  CHOLES CALLS NO SUBROUTINES;
C
C          CALLS FUNCTION SUBROUTINE INDEX;
C
C          CALLS INTRINSIC FUNCTION DSQRT.
C
C
```

```
C   SHIH PING HAN
C   GREGORY D. SHUBERT
C   CORNELL UNIVERSITY
C   1977 NOVEMBER
C
C**********
      INTEGER I, I1, I2, IM1, J, JM1, K
      INTEGER INDEX
      DOUBLE PRECISION EPS2, SUM
      DOUBLE PRECISION DSQRT
      EPS2 = EPS**2
C
C   CHECK INTEGER PARAMETERS
C
      IFLAG = 13
      IF (NL .NE. (N**2 + N)/2) RETURN
      IFLAG = 14
C
C   COMPUTE DECOMPOSITION, ROW BY ROW
C
      DO 80 I = 1, N
         IM1 = I - 1
         IF (IM1 .EQ. 0) GOTO 40
         DO 30 J = 1, IM1
            I1 = INDEX(I,J)
            SUM = AMAT(I1)
            JM1 = J - 1
            IF (JM1 .EQ. 0) GOTO 20
            DO 10 K = 1, JM1
               I1 = INDEX(I,K)
               I2 = INDEX(J,K)
               SUM = SUM - AMAT(I1)*AMAT(I2)
   10          CONTINUE
   20       CONTINUE
            I1 = INDEX(I,J)
            I2 = INDEX(J,J)
            AMAT(I1) = SUM/AMAT(I2)
   30       CONTINUE
C
C   COMPUTE DIAGONAL ELEMENTS
C
   40    CONTINUE
         I1 = INDEX(I,I)
         SUM = AMAT(I1)
         IF (IM1 .EQ. 0) GOTO 60
         DO 50 K = 1, IM1
            I2 = INDEX(I,K)
            SUM = SUM - AMAT(I2)**2
   50       CONTINUE
   60    CONTINUE
         IF (SUM .GT. EPS2) GOTO 70
C
C   MATRIX NO LONGER POSITIVE DEFINITE; SIGNAL THE ERROR.
C
```

```
          RETURN
   70     CONTINUE
          AMAT(I1) = DSQRT(SUM)
   80     CONTINUE
        IFLAG = 0
        RETURN
C
C  LAST CARD OF SUBROUTINE CHOLES
C
        END
```

```
      SUBROUTINE FORWAR(TMAT,XMAT,NL,N,M,IFLAG)
      INTEGER IFLAG, M, N, NL
      DOUBLE PRECISION TMAT, XMAT
      DIMENSION TMAT(NL), XMAT(N,M)
C**********
C
C  FORWAR
C
C   SOLVES FOR Y:  TMAT*Y = XMAT, BY FORWARD SUBSTITUTION
C    AND OVERWRITES THE SOLUTION Y ONTO XMAT.
C
C
C  CALLS TO FORWAR:
C
C   FORWAR (TMAT, XMAT, NL, N, M, IFLAG)
C
C   WHERE
C
C    TMAT    THE NON-ZERO HALF OF AN N-BY-N LOWER TRIANGULAR
C             MATRIX, STORED AS AN NL-VECTOR.
C
C    XMAT    AN N-BY-M MATRIX.  ON OUTPUT IT CONTAINS THE
C             SOLUTION Y DESCRIBED ABOVE.
C
C    NL,     INTEGER VARIABLES GIVING THE SIZES OF TMAT AND
C    M,N      XMAT.  NL = (N**2 + N)/2     M >= 1
C
C    IFLAG   INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C             INFORMATION REGARDING THE SUCCESS OR FAILURE
C             OF FORWAR.
C
C             IFLAG =  0    SUCCESSFUL RETURN
C                   = 15,16 FAILURE, INCONSISTENT INTEGER
C                            PARAMETERS NL, M, N
C                   = 17    FAILURE, A DIAGONAL ELEMENT OF
C                            TMAT = 0
C
C
C  FORWAR CALLS NO SUBROUTINES;
C
C        CALLS FUNCTION SUBROUTINE INDEX:
C
C        CALLS NO INTRINSIC FUNCTIONS.
C
C
C  SHIH PING HAN
C  GREGORY D. SHUBERT
C  CORNELL UNIVERSITY
C  1977 NOVEMBER
C
C**********
      INTEGER I, I1, IM1, J, K
      INTEGER INDEX
      DOUBLE PRECISION DIAG, SUM, ZERO
```

```
      DATA ZERO /0D0/
C
C  CHECK INTEGER INPUT PARAMETERS
C
      IFLAG = 15
      IF (NL .NE. (N**2 + N)/2) RETURN
      IFLAG = 16
      IF (M .LT. 1) RETURN
      IFLAG = 17
C
C  SOLVE BY FORWARD SUBSTITUTION
C
      DO 50 I = 1, N
         IM1 = I - 1
         IF (IM1 .EQ. 0) GOTO 30
         DO 20 J = 1, M
            SUM = ZERO
            DO 10 K = 1, IM1
               I1 = INDEX(I,K)
               SUM = SUM + TMAT(I1)*XMAT(K,J)
   10          CONTINUE
            XMAT(I,J) = XMAT(I,J) - SUM
   20       CONTINUE
   30    CONTINUE
         I1 = INDEX(I,I)
         DIAG = TMAT(I1)
         IF (DIAG .EQ. ZERO) RETURN
         DO 40 J = 1, M
            XMAT(I,J) = XMAT(I,J)/DIAG
   40       CONTINUE
   50    CONTINUE
      IFLAG = 0
      RETURN
C
C  LAST CARD OF SUBROUTINE FORWAR
C
      END
```

```
      SUBROUTINE QR(GMAT,PI,ROIAG,N,L,EPS,IFLAG)
      INTEGER IFLAG, L, N
      DOUBLE PRECISION EPS, GMAT, PI, ROIAG
      DIMENSION GMAT(N,L), PI(L), ROIAG(L)
C**********
C
C  QR
C
C   COMPUTES THE QR-DECOMPOSITION OF MATRIX GMAT SUCH THAT
C    R = Q*GMAT, WHERE R IS UPPER TRIANGULAR AND Q IS
C    ORTHOGONAL.
C
C
C  CALLS TO QR:
C
C   QR (GMAT, PI, ROIAG, N, L, EPS, IFLAG)
C
C   WHERE
C
C    GMAT    AN N-BY-L MATRIX TO BE DECOMPOSED.  ON OUTPUT,
C             THE SUPER-DIAGONAL ELEMENTS OF R ARE STORED IN
C             THE UPPER TRIANGLE OF GMAT, AND Q IS STORED IN
C             THE LOWER TRAPEZOID OF GMAT AS ELEMENTARY
C             REFLECTORS
C               Q = Q(M)*Q(M-1)* ... *Q(1), WHERE
C                   M = MIN(N-L)
C
C    PI      AN L-VECTOR CONTAINING THE SCALARS WHICH OCCUR
C             IN THE REFLECTORS.     T
C               Q(K) = I - V(K)*V(K) /PI(K)
C
C    ROIAG   AN L-VECTOR CONTAINING THE DIAGONAL ELEMENTS OF
C             OF THE R MATRIX.
C
C    N,L     INTEGER VARIABLES GIVING THE SIZES OF GMAT, PI,
C             AND ROIAG.
C
C    EPS     VARIABLE CONTAINING A SMALL POSITIVE NUMBER
C             WHICH IS THE LOWER BOUND ON THE LARGEST
C             ELEMENT IN ABSOLUTE VALUE IN EACH COLUMN OF
C             GMAT.
C
C    IFLAG   INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C             INFORMATION REGARDING SUCCESS OR FAILURE
C             OF QR.
C
C             IFLAG =  0    SUCCESSFUL RETURN
C                   = 18    FAILURE, ORIGINAL MATRIX G IS
C                            NEARLY SINGULAR
C
C
C  QR CALLS NO SUBROUTINES OR FUNCTION SUBROUTINES;
C
C     CALLS INTRINSIC FUNCTIONS DABS, DMAX1, DSIGN, DSQRT,
```

```
C       AND MINO.
C
C
C   SHIH PING HAN
C   GREGORY D. SHUBERT
C   CORNELL UNIVERSITY
C   1977 NOVEMBER
C
C**********
      INTEGER I, J, K, KP1, M
      INTEGER MINO
      DOUBLE PRECISION S, SUM, T, TAU, ZERO
      DOUBLE PRECISION DABS, DMAX1, DSIGN, DSQRT
      DATA ZERO /0D0/
      M = MINO(N-1,L)
      IF (M .EQ. 0) GOTO 190
      DO 180 K = 1, M
C
C   COMPUTE T = MAX |GMAT(I,K)|, I = K, K+1, ... N
C
         T = ZERO
         DO 100 I = K, N
            T = DMAX1(T, DABS(GMAT(I,K)))
  100       CONTINUE
C
C   IF ALL THE ELEMENTS IN THE COLUMN ARE LESS THAN EPS IN
C    ABSOLUTE VALUE, THEN MATRIX GMAT IS NEARLY SINGULAR, AND
C    A REFLECTOR CANNOT BE FORMED.  SET PI(K) = 0, AND SIGNAL
C    THE ERROR.
C
         IF (T .GT. EPS) GOTO 110
         IFLAG = 18
         PI(K) = ZERO
         RETURN
  110    CONTINUE
C
C   THE REFLECTORS Q(K) ARE OF THE FORM
C                              T
C    Q(K) = I - V(K)*V(K) /PI(K), WHERE
C    THE NON-ZERO COMPONENTS (K, K+1, ... N) OF V(K) ARE
C    STORED IN GMAT(K,K) THROUGH GMAT(N,K).
C
         DO 120 I = K, N
            GMAT(I,K) = GMAT(I,K)/T
  120       CONTINUE
C
C   COMPUTE V(K,K), STORE IT IN GMAT(K,K), AND COMPUTE PI(K)
C    AND RDIAG(K)
C
         SUM = ZERO
         DO 130 I = K, N
            SUM = SUM + GMAT(I,K)**2
  130       CONTINUE
         S = DSIGN(DSQRT(SUM), GMAT(K,K))
```

```
          GMAT(K,K) = GMAT(K,K) + S
          PI(K) = S*GMAT(K,K)
          ROIAG(K) = -T*S
C
C   CARRY THE REFLECTION THROUGH THE REST OF THE MATRIX
C
          KP1 = K + 1
          IF (K .EQ. L) GCTO 170
          DO 160 J = KP1, L
             SUM = ZERC
             DO 140 I = K, N
                SUM = SUM + GMAT(I,K)*GMAT(I,J)
  140           CCNTINUE
             TAU = SUM/PI(K)
             DC 150 I = K, N
                GMAT(I,J) = GMAT(I,J) - TAU*GMAT(I,K)
  150           CCNTINUE
  160        CCNTINUE
  170     CONTINUE
  180     CONTINUE
  190 CCNTINUE
      IF (N .LE. L) ROIAG(N) = GMAT(N,N)
      IFLAG = 0
      RETURN
C
C   LAST CARD OF SUBROUTINE QR
C
      END
```

```
      FUNCTION INDEX(I,J)
      INTEGER I, INDEX, J, N
      COMMON /SIZE/ N
C**********
C
C  INDEX
C
C   COMPUTES THE INDEX IN A VECTOR CORRESPONDING TO ROW I
C     AND COLUMN J OF A LOWER TRIANGULAR N-BY-N MATRIX.
C     INDEX STARTS NUMERING AT THE UPPER LEFT-HAND CORNER AND
C     NUMBERS COLUMNS 1, 2, ..., IN ORDER, FROM TOP TO
C     BOTTOM.  INDEX(1,1) = 1.  INDEX(I,1) = I.  INDEX
C     RETURNS -1 IF EITHER I OR J IS NOT POSITIVE OR IS
C     GREATER THAN N.  ROUTINES WHICH CALL INDEX MAY WISH
C     TO TEST FOR THIS.
C
C
C  CALLS TO INDEX:
C
C   INDEX(I,J)
C
C   WHERE
C
C     I,J      INTEGER VARIABLES GIVING THE ROW AND COLUMN IN
C               THE LOWER TRIANGULAR MATRIX.
C
C     N        INTEGER VARIABLE PASSED IN COMMON BLOCK /SIZE/,
C               GIVING THE NUMBER OF ROWS AND COLUMNS IN THE
C               MATRIX.
C
C
C  INDEX CALLS NO SUBROUTINES, FUNCTION SUBROUTINES, OR
C         INTRINSIC FUNCTIONS.
C
C
C  GREGORY C. SHUBERT
C  CORNELL UNIVERSITY
C  1977 NOVEMBER
C
C**********
      INDEX = 0
      IF (I .LE. 0 .OR. J .LE. 0 .OR.
     1    I .GT. N .OR. J .GT. N) INDEX = -1
      IF (INDEX .EQ. -1) RETURN
      INDEX = (J - 1)*(N - 1) + I - (J - 1)*(J - 2)/2
      RETURN
C
C  LAST CARD OF SUBROUTINE INDEX
C
      END
```

```
      SUBROUTINE LEMKE(OMAT,Q,W,U,N,NP1,TOL,LIMIT,IBASIC,
     1                 NONBAS,ITER,IFLAG)
      INTEGER IBASIC, IFLAG, ITER, LIMIT, N, NP1, NONBAS
      DOUBLE PRECISION OMAT, Q, TOL, U, W
      DIMENSION OMAT(N,NP1), IBASIC(N), NONBAS(NP1), Q(N),
     1          TOL(3), U(N), W(N)
C**********
C
C  LEMKE
C
C   SOLVES THE LINEAR COMPLEMENTARITY PROBLEM USING LEMKE'S
C    METHOD
C
C     W = Q + OMAT*U
C                                        T
C     SUBJECT TO   W >= 0,    U >= 0,   W *U = 0
C
C
C  CALLS TO LEMKE:
C
C   LEMKE (OMAT, Q, W, U, N, NP1, TOL, LIMIT, IBASIC,
C          NONBAS, ITER, IFLAG)
C
C   WHERE
C
C    OMAT     N-BY-NP1 MATRIX, THE LAST COLUMN OF WHICH IS AN
C              ARTIFICIAL COLUMN CONTAINING NON-NEGATIVE
C              NUMBERS.
C
C    Q        N-VECTOR, WHICH WITH OMAT SPECIFIES THE PROBLEM
C              TO BE SOLVED.
C
C    U,       N-VECTORS.  ON OUTPUT (NORMAL RETURN,) THEY
C    W         CONTAIN THE SOLUTION.
C
C    N,       INTEGER VARIABLES INDICATING DIMENSIONS.
C    NP1       NP1 = N + 1.
C
C    TOL      3-VECTOR OF POSITIVE TOLERANCES FOR STOPPING
C              TESTS
C
C    LIMIT    INTEGER VARIABLE; THE MAXIMUM NUMBER OF LEMKE
C              ITERATIONS ALLOWED.
C
C    IBASIC   INTEGER N-VECTOR WHICH INDICATES THE CURRENT
C              BASIC VARIABLES.  IF IBASIC(I) = +K OR -K,
C              THEN THE BASIC VARIABLE IS U(K) OR W(K),
C              RESPECTIVELY.
C
C    NONBAS   INTEGER NP1-VECTOR WHICH INDICATES THE CURRENT
C              NON-BASIC VARIABLES.  IF NONBAS(I) = +K OR -K,
C              THEN THE NON-BASIC VARIABLE IS U(K) OR W(K),
C              RESPECTIVELY.
C
```

```
C     ITER      INTEGER VARIABLE.  ON OUTPUT IT GIVES THE
C                NUMBER OF LEMKE ITERATIONS PERFORMED.
C
C     IFLAG     INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C                INFORMATION REGARDING THE SUCCESS OR FAILURE
C                OF LEMKE.
C
C                IFLAG =  0     SUCCESSFUL RETURN
C                      = 21     FAILURE, INCONSISTENT INPUT
C                                 PARAMETERS N, NP1
C                      = 22     FAILURE, NUMBER OF LEMKE
C                                 ITERATIONS > LIMIT
C                      = 23-25 FAILURE, ERRORS IN ROUTINE
C                                 CHOOSE
C
C
C   OTHER VARIABLES USED IN LEMKE:
C
C     IPIVOT    INTEGER, INDEX OF THE PIVOTING ROW.
C
C     JPIVOT    INTEGER, INDEX OF THE PIVOTING COLUMN
C
C     IDUMMY    INTEGER, INDEX OF THE ROW CONTAINING THE DUMMY
C                VARIABLE
C
C     IT        INTEGER, NUMBER OF COMPONENTS OF Q < -TOL(1).
C
C
C   LEMKE CALLS SUBROUTINES CHOOSE AND PIVOT;
C
C         CALLS NO FUNCTION SUBROUTINES OR INTRINSIC
C          FUNCTIONS.
C
C
C   SHIH PING HAN
C   GREGORY D. SHUBERT
C   CORNELL UNIVERSITY
C   1977 OCTOBER
C
C**********
      INTEGER I, IDUMMY, IPIVOT, IT, J, JPIVOT, K
      DOUBLE PRECISION T, TMIN, ZERO
      DATA ZERO /0D0/
C
C   CHECK INTEGER INPUT PARAMETERS
C
      IFLAG = 21
      IF (NP1 .NE. N + 1) RETURN
      IFLAG = 0
C
C   ASSIGN INITIAL VALUES TO IBASIC AND NONBAS
C
      DO 100 I = 1, N
         IBASIC(I) = -I
```

```
          NONBAS(I) = I
  100     CONTINUE
       NONBAS(NP1) = NP1
C
C  SET ALL NEGATIVE COMPONENTS OF THE NP1-ST COLUMN OF DMAT
C   TO ZERO
C
       DO 110 I = 1, N
          IF (DMAT(I,NP1) .LT. ZERO) DMAT(I,NP1) = ZERO
  110     CONTINUE
C
C  IF Q(I) >= -TOL(1) FOR ALL I (IT = 0), THEN SKIP LEMKE
C   ITERATION
C
C  DETERMINE THE FIRST PIVOTING ROW AND COLUMN.  FIND
C   TMIN = MIN Q(I)/DMAT(I,NP1) : Q(I) < -TOL(1).
C   LET IPIVOT BE THE I CORRESPONDING TO TMIN.
C   LET JPIVOT BE NP1.
C
       IT = 0
       DO 130 I = 1, N
          IF (Q(I) .GE. -TOL(1)) GOTO 130
          IT = IT + 1
          IF (DMAT(I,NP1) .LT. TOL(2)) DMAT(I,NP1) = TOL(2)
          T = Q(I)/DMAT(I,NP1)
          IF (IT .EQ. 1) GOTO 120
          IF (T .LT. TMIN) GOTO 120
          GOTO 130
  120     CONTINUE
          TMIN = T
          IPIVOT = I
  130     CONTINUE
       IF (IT .EQ. 0) GOTO 170
       JPIVOT = NP1
       IDUMMY = IPIVOT
C
C  LEMKE ITERATION
C
  140 CONTINUE
       IFLAG = 22
       IF (ITER .GT. LIMIT) RETURN
       IFLAG = 0
       CALL PIVOT(DMAT,Q,N,NP1,IPIVOT,JPIVOT)
       ITER = ITER + 1
C
C  INTERCHANGE BASIC AND NON-BASIC VARIABLES
C
       K = IBASIC(IPIVOT)
       IBASIC(IPIVOT) = NONBAS(JPIVOT)
       NONBAS(JPIVOT) = K
C
C  CHECK CONVERGENCE
C
       IF (Q(IDUMMY) .LE. TOL(1)) GOTO 170
```

```
      IF (ITER .GT. 1 .AND. IDUMMY .EQ. IPIVOT) GOTO 170
C
C  CHOOSE THE NEXT PIVOTING COLUMN AND ROW.  IF W(K) OR X(K)
C   BECOMES NONBASIC THEN WE CHOOSE THE COLUMN CORRESPONDING
C   TO U(K) OR W(K) AS THE PIVOTING COLUMN, RESPECTIVELY.
C
      IT = -NONBAS(JPIVOT)
      DO 150 J = 1, NP1
         JPIVOT = J
         IF (NONBAS(J) .EQ. IT) GOTO 160
  150    CONTINUE
  160 CONTINUE
      CALL CHOOSE(Q,DMAT(1,JPIVOT),N,IPIVOT,IDUMMY,TOL,
     1            IFLAG)
      IF (IFLAG .NE. 0) RETURN
C
C  NEXT LEMKE ITERATION
C
      GOTO 140
C
C  EVALUATE U AND W
C   ALL NONBASIC VARIABLES ARE SET EQUAL TO ZERO.
C   IF IBASIC(I) = +K OR -K, THEN U(K) OR W(K),
C   RESPECTIVELY, GETS THE VALUE OF Q(I).
C
  170 CONTINUE
      DO 180 I = 1, N
         U(I) = ZERO
         W(I) = ZERO
  180    CONTINUE
      DO 190 I = 1, N
         K = IBASIC(I)
         IF (K .GT. 0 .AND. K .NE. NP1) U(K) = Q(I)
         IF (K .LT. 0) W(-K) = Q(I)
  190    CONTINUE
      RETURN
C
C  LAST CARD OF SUBROUTINE LEMKE
C
      END
```

```
      SUBROUTINE CHCOSE(Q,Y,N,IPIVOT,IOUMMY,TCL,IFLAG)
      INTEGER IOUMMY, IFLAG, IPIVOT, N
      DOUBLE PRECISION Q, TCL, Y
      DIMENSION Q(N), TOL(3), Y(N)
C**********
C
C  CHCOSE
C
C   SELECTS THE PIVCTING RCW CNCE THE PIVOTING COLUMN IS
C    DETERMINED
C
C
C  CALLS TO CHCCSE:
C
C   CHOOSE (Q, Y, N, IPIVOT, IOUMMY, TOL, IFLAG)
C
C   WHERE
C
C    Q        N-VECTCR; THE CURRENT VECTOR IN THE PRCBLEM
C                                                      T
C              W = Q + DMAT*U,  W >= 0,  U >= 0,  W *U = 0
C
C    Y        N-VECTCR; THE PIVOTING COLUMN FROM DMAT.
C
C    N        INTEGER VARIABLE INDICATING THE SIZE OF Q AND Y
C
C    IPIVOT   INTEGER VARIABLE.  ON OUTPUT IT INDICATES THE
C              PIVCTING RCW.
C
C    IOUMMY   INTEGER VARIABLE: THE INDEX OF THE RCW
C              CCNTAINING THE DUMMY VARIABLE.
C
C    TOL      3-VECTCR OF POSITIVE TOLERANCES FOR STOPPING
C              TESTS
C
C    IFLAG    INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C              INFORMATICN REGARDING THE SUCCESS OR FAILURE
C              OF CHCOSE.
C
C              IFLAG =  0    SUCCESSFUL RETURN
C                    = 23    FAILURE, INFEASIBLE
C                              Q(I) <  -TOL(1), ANY I
C                    = 24    FAILURE, NO SOLUTION
C                              Y(I) >= -TOL(2), ALL I
C                    = 25    FAILURE, PIVCT TOO SMALL
C
C
C  CTHER VARIABLES USED IN CHOOSE:
C
C    IT       THE NUMBER CF COMPONENTS IN Y < -TOL(2)
C
C
C
C  CHCOSE CALLS NC SUBROUTINES OR FUNCTION SUBROUTINES:
```

```
C
C          CALLS INTRINSIC FUNCTION DABS.
C
C
C   SHIH PING HAN
C   GREGORY D. SHUBERT
C   CORNELL UNIVERSITY
C   1977 OCTOBER
C
C**********
      INTEGER I, IT
      DOUBLE PRECISION T, TMIN, YMIN
      DOUBLE PRECISION DABS
C
C   CHECK FOR FEASIBILITY AND EXISTENCE OF SOLUTION,
C
C   FIND TMIN = MIN -(Q(I) + TOL(1))/ Y(I) : Y(I) < -TOL(2)
C
      IT = 0
      IFLAG = 23
      DO 10 I = 1, N
         IF (Q(I) .LT. -TOL(1)) RETURN
         IF (Y(I) .GE. -TOL(2)) GOTO 10
         IT = IT + 1
         T = -(Q(I) + TOL(1))/Y(I)
         IF (IT .EQ. 1) TMIN = T
         IF (T .LT. TMIN) TMIN = T
   10    CONTINUE
      IFLAG = 24
      IF (IT .EQ. 0) RETURN
      IFLAG = 0
C
C   FIND I CORRESPONDING TO
C
C      MAX |Y(I)| : -Q(I)/Y(I) <= TMIN  AND  Y(I) < -TOL(2)
C
C    LET IDUMMY BE PREFERRED IN A TIE AMONG I.
C    LET IPIVOT BE THIS I.
C
      YMIN = -TOL(2)
      T = -Q(IDUMMY)/Y(IDUMMY)
      IF (Y(IDUMMY) .GT. -TOL(2) .OR. T .GT. TMIN) GOTO 20
      IPIVOT = IDUMMY
      YMIN = Y(IPIVOT)
   20 CONTINUE
      DO 30 I = 1, N
         IF (Y(I) .GT. -TOL(2)) GOTO 30
         T = -Q(I)/Y(I)
         IF (T .GT. TMIN .OR. Y(I) .GE. YMIN) GOTO 30
         IPIVOT = I
         YMIN = Y(I)
   30    CONTINUE
C
C   IF | Y(IPIVOT) | < TOL(3)*MAX | Y(I) |
```

```
C    THEN THE PIVOT IS TOO SMALL.
C
      T = DABS(Y(IPIVOT))
      DO 40 I = 1, N
         IFLAG = 25
         IF (T .LT. TOL(3)*DABS(Y(I))) RETURN
   40    CONTINUE
      IFLAG = 0
      RETURN
C
C  LAST CARD OF SUBROUTINE CHOOSE
C
      END
```

```
      SUBROUTINE PIVOT(D,Q,M,N,IPIVOT,JPIVOT)
      INTEGER IPIVOT, JPIVOT, M, N
      DOUBLE PRECISION D, Q
      DIMENSION D(M,N), Q(M)
C**********
C
C  PIVOT
C
C   PIVOTS ON ROW = IPIVOT  AND  COLUMN = JPIVOT
C
C
C  CALLS TO PIVOT:
C
C   PIVOT (D, Q, M, N, IPIVOT, JPIVOT)
C
C   WHERE
C
C    D       M-BY-N MATRIX.  ON OUTPUT IT CONTAINS THE
C             MATRIX REFLECTING THE PIVOT.
C
C    Q       M-VECTOR.  ON OUTPUT IT CONTAINS THE VECTOR
C             REFLECTING THE PIVOT.
C
C    M,      INTEGER VARIABLES INDICATING THE SIZE OF D AND
C    N        Q.
C
C    IPIVOT, INTEGER VARIABLES; INDICATE THE PIVOT ROW AND
C    JPIVOT   PIVOT COLUMN, RESPECTIVELY.
C
C
C  PIVOT CALLS NO SUBROUTINES OR FUNCTION SUBROUTINES;
C
C        CALLS NO INTRINSIC FUNCTIONS.
C
C
C  SHIH PING HAN
C  GREGORY D. SHUBERT
C  CORNELL UNIVERSITY
C  1977 OCTOBER
C
C**********
      INTEGER I, J
      DOUBLE PRECISION APIVOT, ONE
      DATA ONE /1.0D0/
C
C  COMPUTE THE PIVOTING ROW
C
      D(IPIVOT,JPIVOT) = ONE/D(IPIVOT,JPIVOT)
      APIVOT = D(IPIVOT,JPIVOT)
      DO 10 J = 1, N
         IF (J .EQ. JPIVOT) GOTO 10
         D(IPIVOT,J) = -D(IPIVOT,J)*APIVOT
   10    CONTINUE
      Q(IPIVOT) = -Q(IPIVOT)*APIVOT
```

```
C
C   COMPUTE EVERY ROW EXCEPT PIVOTING ROW
C
      DO 30 I = 1, M
         IF (I .EQ. IPIVOT) GOTO 30
C
C     ALL COLUMNS BUT PIVOTING COLUMN
C
         DO 20 J = 1, N
            IF (J .EQ. JPIVOT) GOTO 20
            D(I,J) = D(I,J) + D(I,JPIVOT)*D(IPIVOT,J)
   20       CONTINUE
C
C     PIVOTING COLUMN
C
         Q(I) = Q(I) + D(I,JPIVOT)*Q(IPIVOT)
         D(I,JPIVOT) = D(I,JPIVOT)*APIVOT
   30    CONTINUE
      RETURN
C
C   LAST CARD OF SUBROUTINE PIVOT
C
      END
```

```
      SUBROUTINE EVALU(TMAT,GMAT,PI,RDIAG,ZMAT,D,Z,U,V,X,L,
     1                 M, N,NL,EQCON,NQCON,IFLAG)
      INTEGER IFLAG, L, M, N, NL
      LOGICAL EQCON, NQCON
      DOUBLE PRECISION D, GMAT, PI, RDIAG, TMAT, U, V, X, Z,
     1                 ZMAT
      DIMENSION D(L), GMAT(N,L), PI(L), RDIAG(L), TMAT(NL),
     1          U(M), V(L), X(N), Z(N), ZMAT(N,M)
C**********
C
C  EVALU
C
C   COMPUTES VECTORS V AND X FROM VECTOR U AND OTHER
C    MATRICES AND VECTORS FOUND PREVIOUSLY.
C
C
C  CALLS TO EVALU:
C
C   EVALU (TMAT, GMAT, PI, RDIAG, ZMAT, D, Z, U, V, X, L, M,
C          N, NL, EQCON, NQCON, IFLAG)
C
C   WHERE
C
C    TMAT     THE NON-ZERO HALF OF AN N-BY-N LOWER TRIANGULAR
C              MATRIX STORED AS AN NL-VECTOR.
C
C    GMAT     AN N-BY-L MATRIX CONTAINING R IN ITS UPPER
C              TRIANGLE AND THE REFLECTORS Q(K) OR Q BELOW.
C              Q*GMAT = R.  Q = Q(IM)*Q(IM-1)* ... *Q(1),
C              WHERE IM = MIN(N-1,L).
C
C    PI       AN L-VECTOR CONTAINING SCALARS WHICH OCCUR IN
C              THE REFLECTORS Q(K).
C
C    RDIAG    AN L-VECTOR CONTAINING THE DIAGONAL ELEMENTS OF
C              THE R MATRIX (THE REST OF WHICH IS IN GMAT).
C
C    ZMAT     AN N-BY-M MATRIX CONTAINING ZMAT1 IN ITS UPPER
C              L ROWS, AND ZMAT2 BELOW.
C
C    Z        AN N-VECTOR CONTAINING Z1 IN ITS UPPER L ROWS,
C              AND Z2 BELOW.
C
C    D        AN L-VECTOR FROM THE LINEAR COMPLEMENTARITY
C              PROBLEM.
C
C    U,V      AN M-VECTOR AND AN L-VECTOR, RESPECTIVELY.  ON
C              OUTPUT, THEY CONTAIN TWO OF THE DESIRED
C              SOLUTION VECTORS.
C
C    X        AN N-VECTOR.  ON OUTPUT, IT CONTAINS THE OTHER
C              SOLUTION VECTOR.
C
C    L,M,     INTEGER VARIABLES GIVING THE SIZES OF THE
```

```
C     N,NL      VARIOUS MATRICES AND VECTORS.
C               NL = (N**2 + N)/2
C
C     EQCON,  LOGICAL VARIABLES, WHICH ARE TRUE OF THERE ARE
C     NQCON     EQUALITY OR INEQUALITY CONSTRAINTS,
C               RESPECTIVELY, AND FALSE OTHERWISE.
C
C     IFLAG   INTEGER VARIABLE.  ON OUTPUT IT CONTAINS
C               INFORMATION REGARDING SUCCESS OR FAILURE
C               OF EVALU.
C
C               IFLAG =  0    SUCCESSFUL RETURN
C                     = 31    FAILURE, INCONSISTENT INTEGER
C                              INPUT PARAMETERS NL, N
C
C
C   EVALU CALLS NO SUBROUTINES;
C
C         CALLS FUNCTION SUBROUTINE INDEX;
C
C         CALLS INTRINSIC FUNCTION MINO.
C
C
C   SHIH PING HAN
C   GREGORY C. SHUBERT
C   CORNELL UNIVERSITY
C   1977 NOVEMBER
C
C**********
      INTEGER I, I1, IM, IP1, J, K, LP1
      INTEGER INDEX
      INTEGER MINO
      DOUBLE PRECISION SUM, ZERO
      DATA ZERO /0D0/
C
C   CHECK FOR ABSENCE OF EQUALITY OR INEQUALITY CONSTRAINTS
C
      IF (.NOT. EQCON) L = 0
      IF (.NOT. NQCON) M = 0
C
C   CHECK INTEGER INPUT PARAMETERS
C
      IFLAG = 31
      IF (NL .NE. (N**2 + N)/2) RETURN
      IFLAG = 0
C
C                     T       T |      -D      |
C   SOLVE FOR X:  TMAT *X = -Q *|--------------|
C                                | ZMAT2*U + Z2 |
C    RIGHT-HAND SIDES
C    FOR SPECIAL CASES:    = -ZMAT2*U - Z2        (.NOT. EQCON)
C
C                            T | -D |
C                        = -Q *|----|             (.NOT. NQCON)
```

```
C                                  | Z2 |
C
C                              = -Z2                    (.NOT. EQCON
C                                               .AND. .NOT. NQCON)
C
C
C  COMPUTE THE RIGHT-HAND SIZE AND STORE IT IN X
C
      IF (.NOT. EQCON) GOTO 200
C
C                |       -D       |          | -D |
C  STORE X = -|----------------|   OR   -|----|
C                | ZMAT2*U + Z2 |          | Z2 |
C
C   IF NQCON IS TRUE OR FALSE, RESPECTIVELY.
C
      DO 100 I = 1, L
         X(I) = D(I)
  100    CONTINUE
      LP1 = L + 1
      IF (L .EQ. N) GOTO 140
      DO 110 I = LP1, N
         X(I) = -Z(I)
  110    CONTINUE
      IF (.NOT. NQCON) GOTO 140
      DO 130 I = LP1, N
         SUM = ZERO
         DO 120 J = 1, M
            SUM = SUM + ZMAT(I,J)*U(J)
  120       CONTINUE
         X(I) = X(I) - SUM
  130    CONTINUE
  140 CONTINUE
C                                   T
C  MULTIPLY X ON THE LEFT BY Q  , THAT IS, BY Q(IM), Q(IM-1),
C  ... Q(1), IN TURN, WHERE IM = MIN(N-1,L).  STORE THE
C  RESULT IN X.
C                             T
C   Q(K) = I - V(K)*V(K) /PI(K)
C                                 T
C   Q(K)*X = X - V(K)*(V(K) *X)/PI(K)
C
C   AND THE NON-ZERO ELEMENTS (K, K+1, ... N) OF V(K) ARE
C   STORED IN GMAT(K,K) THROUGH GMAT(N,K).
C
      IM = MIN0(N-1,L)
      IF (IM .EQ. 0) GOTO 180
C
C  DO K = IM, IM-1, . . . 2, 1
C
      DO 170 I = 1, IM
         K = IM - I + 1
C
C   IF PI(K) = 0, THERE IS NO REFLECTION
```

```
C
         IF (PI(K) .EQ. ZERO) GOTO 170
C                       T
C  COMPUTE SUM = V(K) *X/PI(K)
C
         SUM = ZERO
         DO 150 J = K, N
            SUM = SUM + GMAT(J,K)*X(J)
  150       CONTINUE
         SUM = SUM/PI(K)
C
C  COMPLETE THE UPDATE OF X
C
         DO 160 J = K, N
            X(J) = X(J) - GMAT(J,K)*SUM
  160       CONTINUE
  170    CONTINUE
  180 CONTINUE
      GOTO 240
C
C  HERE, EQCON = .FALSE.
C
  200 CONTINUE
C
C  STORE X = -ZMAT2*U - Z2  OR  -Z2
C   IF NQCON IS TRUE OR FALSE, RESPECTIVELY
C
      DO 210 I = 1, N
         X(I) = -Z(I)
  210    CONTINUE
      IF (.NOT. NQCON) GOTO 240
      DO 230 I = 1, N
         SUM = ZERO
         DO 220 J = 1, M
            SUM = SUM + ZMAT(I,J)*U(J)
  220       CONTINUE
         X(I) = X(I) - SUM
  230    CONTINUE
  240 CONTINUE
C
C  LET Y BE THE RIGHT-HAND SIDE VECTOR PRESENTLY STORED IN
C   X.  SOLVE BY BACKWARD SUBSTITUTION FOR X:
C        T
C     TMAT *X = Y
C
C   NOTE TMAT IS STORED AS A NL-VECTOR
C
C
C  DO I = N, N-1, ... , 2, 1
C
      DO 320 J = 1, N
         I = N - J + 1
         SUM = ZERO
         IP1 = I + 1
```

```
         IF (I .EC. N) GOTO 310
         SUM = ZERO
         DO 300 K = IP1, N
            I1 = INDEX(K,I)
            SUM = SUM + TMAT(I1)*X(K)
  300       CCNTINUE
  310    CONTINUE
         I1 = INDEX(I,I)
         X(I) = (X(I) - SUM)/TMAT(I1)
  320    CONTINUE
C
C  SOLVE FOR V:  RMAT*V = -(ZMAT1*U + Z1 + D)
C
C   RIGHT-HAND SIDE FOR (.NCT. NQCON):   -(Z1 + D)
C
C   SKIP THIS SECTICN IF EQCON IS FALSE
C
C   FIRST, STORE THE RIGHT-HAND SIDE IN V
C
      IF (.NCT. EQCCN) GCTO 470
      DO 400 I = 1, L
         V(I) = -(Z(I) + D(I))
  400    CONTINUE
      IF (.NCT. NCCCN) GOTO 430
      DO 420 I = 1, L
         SUM = ZERO
         DO 410 J = 1, M
            SUM = SUM + ZMAT(I,J)*U(J)
  410       CCNTINUE
         V(I) = V(I) - SUM
  420    CONTINUE
  430 CCNTINUE
C
C  LET Y BE THE RIGHT-HAND SIDE VECTOR PRESENTLY STORED IN
C   V.  SOLVE BY BACKWARD SUBSTITUTION FOR V:
C
C    R*V = Y
C
C   NCTE R IS STOREC IN THE UPPER TRIANGLE OF GMAT, AND ITS
C   DIAGONAL ELEMENT ARE STCREC IN RDIAG
C
C
C  DO I = L, L-1, ... 2, 1
C
      DC 460 J = 1, L
         I = L - J + 1
         SUM = ZERO
         IP1 = I + 1
         IF (I .EC. L) GCTO 450
         DO 440 K = IP1, L
            SUM = SUM + GMAT(I,K)*V(K)
  440       CCNTINUE
  450    CONTINUE
         V(I) = (V(I) - SUM)/RDIAG(I)
```

```
  460    CONTINUE
  470 CCNTINUE
      RETURN
C
C  LAST CARD CF SUBRCUTINE EVALU
C
      END
```