SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

153 003

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina
27709 USA

SPLA CONTROL NUMBER: 203

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the "SHARE Reference Manual".

(1) Program Number (to be filled in by SPLA)..... 360D-15.3.003

(2) System Type (machine).....

(3) Search Key..... Quadratic/Program/; Convex/Program/; Lemke's/Algorithm/; Nonlinear/Program/; Linear/Program/; Complementary/Algorithm/

(4) Programming Systems/Languages..... FORTRAN 4

(5) Author's Name and Address..... Professor A. Ravindran
School of Industrial Engineering
Purdue University, W. Lafayette, IN 47907

(6) Direct Technical Inquiries to Name & Address (if different than Author) (Same as above)

(7) Title of Program..... A Complementary Pivot Method for Solving Quadratic Programming Problems

(8) Submitter's Installation Membership Code.....

(9) Submitter's Own Program Identification and Suffix(Optional).. QPLP

(10) Primary Subject Code..... 15.3

(11) Minimum System Requirements See Abstract

(12) New or Revision Code (if revision, show prior Program Number in Item 1) N

(13) Year Completed..... 1968

(14) Date of Submittal..... August, 1976

(15) Documentation (number of original pages submitted)..... 13

(16) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

Revised 4/74

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. **Purpose**
b. **Programming Language used**
c. **Version and modification level or release number**
d. **Field of application**
e. **Type of routine (main program, subroutine, function)**
f. **Specific description of machine requirements**

DISCLAIMER

Triangle Universities Computation Center (TUCC) serves solely as the distribution agent for contributed programs and does not test or maintain them. They are distributed essentially in the original form submitted by the author. Neither TUCC nor SHARE, INC., makes any warranty, expressed or implied, as to the documentation, function, or performance of the contributed programs.

ABSTRACT

This program can solve any convex quadratic programming or linear programming problem. The entire program is written in FORTRAN 4 so that it can be implemented easily in any computing system. The program is based on the complementary pivot method for solving complementary problems. Its main field of application is in management science/operations research for solving nonlinear programming or constrained optimization problems. The program consists of a main program and a number of subroutines written in FORTRAN language. In its present form it requires 70k words in CDC 6500 machine for loading and executing and can solve quadratic or linear programming problems whose number of rows plus columns do not exceed 75. The problem size can be reduced to accommodate core availability of smaller machines. Larger problems can be solved by increasing the size of the DIMENSION statements.

(Please attach additional pages if necessary).....Total pages attached

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program."

(17) **Signature of Submitter and Date** A. Ravindran 8/4/76

(18) **Signature of Installation Addressee**

DOCUMENTATION

Table of Contents

I. Purpose

(i) Quadratic Program

This program can solve any convex quadratic programming problem expressed in the following form:

$$\begin{aligned} &\text{Minimize} && z = cx + x'Dx \\ &\text{Subject to} && Ax \geq b \\ & && x \geq 0 \end{aligned} \qquad (1)$$

where A is an (m x n) constraint matrix, b is an (m x 1) requirement vector, c is an (1 x n) cost vector, x is an (n x 1) vector of decision variables, and D is the matrix of quadratic form which should be positive definite or positive semi-definite for the algorithm to guarantee an optimal solution. (The user may request the program to verify whether D is positive definite or positive semi-definite by setting the optional parameter NCHCK to a value 1.)

Equality constraints in the problem can be easily changed to inequality form as described below:

Suppose there are 'k' equality constraints of the form

$$\sum_j a_{ij}x_j = b_i \qquad \text{for } i=1,\ldots,k$$

This can be converted to equivalent set of (k+1) inequalities as follows:

$$\sum_j a_{ij}x_j \geq b_i \qquad \text{for } i=1,\ldots,k$$

$$-\sum_{i=1}^{k} \sum_j a_{ij}x_j \geq -\sum_{i=1}^{k} b_i$$

(ii) Linear Program

The same computer program can also be used to solve any linear programming problem expressed in the form:

$$\begin{aligned} &\text{Minimize} \quad z = cx \\ &\text{Subject to} \quad Ax \geq b \\ &\qquad\qquad x \geq 0 \end{aligned} \tag{2}$$

where A, b, c, and x are as defined before. Note that a linear program is a special case of quadratic program with the matrix of quadratic form D as zero.

II. Functional and Mathematical Methods

The computer code is based on Lemke's complementary pivot algorithm [1] and the author's computer routine [3] to solve a complementary problem of the form:

Find vectors w and z such that

$$\begin{aligned} &w = Mz + q \\ &w, z \geq 0 \\ &w'z = 0 \end{aligned} \tag{3}$$

where M is an $(n \times n)$ matrix; w, z, and q are n-dimensional column vectors.

The two most important applications of the complementary problem (3) are to solve linear and convex quadratic programs by converting them to equivalent complementary problems.

Quadratic Programming: Consider the quadratic program defined in (1). An optimal solution to that problem may be obtained by solving a complementary problem of the form:

$$\begin{pmatrix} v \\ u \end{pmatrix} = \begin{pmatrix} D+D' & -A' \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c' \\ -b \end{pmatrix} \qquad (4)$$

$$u,v,x,y \geq 0$$

$$v'x + u'y = 0$$

where u denotes the slack variables of the given quadratic program, and (y,v) denotes the variables of the dual problem. Comparing the above system with the original complementary problem given by (3), we note that

$$w = \begin{pmatrix} v \\ u \end{pmatrix}, \quad z = \begin{pmatrix} x \\ y \end{pmatrix}, \quad M = \begin{pmatrix} D+D' & -A' \\ A & 0 \end{pmatrix}, \text{ and } \quad q = \begin{pmatrix} c' \\ -b \end{pmatrix}. \qquad (5)$$

The computer routine first transforms the given quadratic program to an equivalent complementary problem using transformation (5), solves the complementary problem by Lemke's algorithm, and retransforms the solution.

It should be remarked here that Lemke's algorithm is guaranteed to work whenever the matrix M is positive semi-definite. In system (5), M is positive semi-definite if and only if the matrix of quadratic program D is positive semi-definite or positive definite.

Linear Programming: The only difference between a linear program and a quadratic program is in the objective function. Hence, by setting $D = 0$ in system (5), we get the equivalent complementary problem for a linear program.

For a detailed discussion of the complementary pivot method and its applications to linear and quadratic programming, refer to Phillips, Ravindran, and Solberg [2]. Recent studies by Ravindran [4] and Ravindran and Lee [5] have shown the superiority of the complementary pivot method for solving linear and convex quadratic programming problems.

III. Limitations

1. In its present form, the program can handle problems of size $(m+n) \leq 75$. For larger problems, a change in the DIMENSION statement is required.

2. The program is guaranteed to solve any linear program and convex quadratic program. Convexity of the quadratic function is dependent on the matrix D being positive definite or positive semi-definite. If D does not satisfy this property, the program could either converge to a local minimum or terminate erroneously indicating that there is no optimal solution. The user may request the program to verify whether D is positive definite or positive semi-definite by setting the optional parameter NCHECK in the input data card to a value one.

3. In this program, the property of positive semi-definiteness of D is checked by utilizing a driver routine "RSEIG" developed by the Purdue University Computing Center and thus, is a system dependent subroutine. Hence, this should be substituted by a similar routine which will compute the eigenvalues and eigenvectors (if available) at other computing centers. If this subroutine is not needed, a dummy subroutine "RSEIG" should be included. This program could then be used as such by making sure that the integer variable "NCHCK" is set equal to zero or "-1" in the appropriate data card to avoid checking the property of the definiteness of D.

4. The timing routine is also a system dependent function which could be taken care of as follows:

(i) Delete the variable "START" from all the COMMON cards;

(ii) Delete the statement "CALL SECOND (START)" from the main program;

(iii) Delete the three statements:

CALL SECOND (FTN)

FIN = FIN - START

PRINT 109, TIME

from the subroutine "SORT".

(iv) Delete the three statements:

CALL SECOND (FIN)

TIME = FIN - START

PRINT 113, TIME

from the subroutine "PRINT".

IV. Environment Requirements

1. In its present form, the program's core requirements are close to 70000 octal words. This can be reduced by reducing the size of the DIMENSION statements.

2. In subroutine "SORT", the statement

TOL = AMAX*2.0**(-37)

should be changed according to the computer system available. In general, TOL = AMAX*2.0**(-NB) and NB should be replaced by (B-11) where B is the number of bits in the floating point mantissa of the computer. This tolerance statement is needed to avoid complications caused by degeneracy in the solution.

V. Input/Output

Input Format:

1. The program will solve any number of problems in succession. The first data card must contain the title for the problem in (20A4) format.

2. The second data card contains the following information in (6I2) format:

NVAR = Number of decision variables

NCON = Number of constraints

IPACK = 0 or blank, if matrices D and A are to be input in their entirety

= 1 if only the nonzero elements of D and A are to be input in a packed form

NCHCK = -1 if D is already known to be positive definite or positive semi-definite

= 0 if D is not known to be positive definite or semi-definite and no such determination is required

= 1 if a determination of definiteness of D is to be performed

LP = 0 or blank, if the problem is a quadratic program

= 1 if it is a linear program

NPRNT = 1 if printout of input is desired

= 0 if it is not desired

3. Use this only if IPACK = 0.

(i) The next set of data cards contains the quadratic form matrix D, columnwise in (7F10.5) format. (Omit this card if LP = 1.)

(ii) The next set of data cards contains the constraint matrix A, columnwise in (7F10.5) format.

4. Use this only if IPACK = 1.

Use of packed form input: For each column, in [7(I2,F8.4)] format, enter the row in which the nonzero element occurs, followed by the element.

Note that each column must end with a blank or zero, e.g., if there are seven nonzero elements in a column, a blank card should follow the card containing the elements. Each column (including zero columns) must have at least one card.

(i) The next set of data cards contains the nonzero elements of the quadratic form matrix D, columnwise in packed form. (Omit this card if LP = 1.)

(ii) The next set of data cards contains the nonzero elements of the constraint matrix A, columnwise in packed form.

5. The next set of cards contains the cost vector c in (7F10.5) format.

6. The next set of cards should contain the right hand side vector b in (7F10.5) format.

7. If there is another problem, repeat steps 1 through 6; otherwise, terminate with a blank card as the last card in the data deck.

Output:

In general the output will contain the following information:

1. The output of each problem starts on a new page.

2. The first item printed is the title of the problem.

3. The next output is a statement pertaining to the definiteness of the quadratic form matrix D, unless NCHCK was set to a negative integer, in which case no statement is printed.

4. If NPRNT was set equal to "one", the input is then printed in the following order:

(a) the cost vector c,

(b) the right hand side vector b,

(c) the constraint matrix A, (rowwise),

(d) the quadratic form matrix D.

(Note: D is printed only if the variable LP = 0. In addition, D is printed in the symmetric form, i.e., $\frac{1}{2}(D+D^T)$ is printed.)

5. The optimal solution and the number of iterations. (Note: If the problem does not have an optimal solution, the message "PROBLEM HAS NO SOLUTION" and the "ITERATION NO." will be printed.)

6. The Lagrange multipliers which may be interpreted as shadow prices for the constraints.

7. The value of the objective function.

8. The CPU time for the problem.

VI. How to Use the Program

Since the entire program and its subroutines are given in FORTRAN IV, only the usual control cards for compiling and executing a FORTRAN program are needed. The data structure and a sample output for an illustrative problem is given below:

Illustration

Consider the Quadratic Program:

$$\text{Minimize} \quad f = -6x_1 + 2x_1^2 - 2x_1x_2 + 2x_2^2$$

$$\text{Subject to} \quad -x_1 - x_2 \geq -2$$

$$x_1, x_2 \geq 0$$

Comparing with notations used before,

$$C = (-6, 0); \quad D = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}; \quad A = (-1, -1); \quad b = (-2).$$

(Note: D is positive definite and hence Lemke's algorithm is guaranteed to find an optimal solution if one exists.)

INPUT DATA

The input cards would be punched as shown below:

Card No.	Punched Data
1	AN EXAMPLE PROBLEM
2	020100010001
3	2.0 -1.0
4	-1.0 2.0
5	-1.0
6	-1.0
7	-6.0 0.0
8	-2.0
9	(Blank card)

SAMPLE OUTPUT

```
                              AN EXAMPLE PROBLEM

***** D IS A POSITIVE DEFINITE MATRIX. LEMKE S ALGORITHM IS *****
***** GUARANTEED TO FIND AN OPTIMAL SOLUTION IF ONE EXISTS *****

THE COST VECTOR

   -6.00000       .00000

THE RIGHT HAND SIDE VECTOR

   -2.00000

THE CONSTRAINT MATRIX (ROWWISE)

   -1.00000    -1.00000

THE QUADRATIC FORM MATRIX

    2.00000    -1.00000

   -1.00000     2.00000

SOLUTION -- ITERATION NO.    4

   X(  1)=          1.50000

   X(  2)=           .50000

THE LAGRANGE MULTIPLIERS ARE
```

```
Y(  1)=         1.00000

THE VALUE OF THE OBJECTIVE FUNCTION AT THE OPTIMAL POINT IS  -5.50000

FOR THIS PROBLEM THE CPU TIME WAS      .056 SECONDS
```

VII. Prior Testing

The program has been tested extensively for solving both linear and quadratic programming problems. In an experimental study conducted by the author [4], this program was compared against the simplex method for solving linear programming problems. More than 170 randomly generated linear programs were run varying in size from (5 x 5) to (15 x 15) to compare the iteration counts and execution times. The computer routine consistently did better than the simplex method. Encouraged by this success, another experimental study to solve quadratic programming problems has just been completed by Ravindran and Lee [5]. In this study other important quadratic programming algorithms are compared against this computer routine. The results of the study clearly demonstrate the superiority of the complementary pivot method to solve quadratic programs as well.

VIII. Magnetic Tape Key

This volume contains 5 files and 5 tape marks arranged as follows:

File 1 FORTRAN SOURCE deck
EBCDIC
Sequence set up;
PROGRAM LEMKE 100 through 1290 in cols. 77-80; LMK in cols. 73-75; 120 cards.
SUBROUTINE MATRIX 100 through 1750 in cols. 77-80; MTX in cols. 73-75; 166 cards.
SUBROUTINE INITIA 100 through 700 in cols. 78-80; INT in cols. 73-75; 61 cards.
SUBROUTINE NEWBAS 100 through 490 in cols. 78-80; NEW in cols. 73-75; 40 cards.
SUBROUTINE SORT 100 through 680 in cols. 78-80; SRT in cols. 73-75; 59 cards.

SUBROUTINE PIVOT 100 through 390 in cols. 78-80; PVT in cols. 73-75; 30 cards.
SUBROUTINE PPRINT 100 through 710 in cols. 78-80; PPT in cols. 73-75; 62 cards.
(Total of 538 cards)
538 card images blocked 20 per block
27 blocks of 1600 characters each.
T/M

File 2 :Sample Problem 1
EBCDIC
Sequence 01 through 09 in cols. 79-80;
PR01 in cols. 73-76; 9 cards.
9 card images blocked 20 per block
1 block of 1600 characters
T/M

File 3 :Output from sample problem 1
EBCDIC
Sequence 01 through 40 in cols. 131-132;
SOL1 in cols. 125-128; 40 cards
40 card images blocked 10 per block
4 blocks of 1320 characters each
T/M

File 4 :Sample Problem 2
EBCDIC
Sequence 01 through 17 in cols. 79-80;
PRO2 in cols. 73-76; 17 cards.
17 card images blocked 20 per block
1 block of 1600 characters
T/M

File 5 :Output from sample problem 2
EBCDIC
Sequence 01 through 72 in cols. 131-132;
SOL2 in cols. 125-128; 72 cards
72 card images blocked 10 per block
8 blocks of 1320 characters each
T/M

IX. References

1. Lemke, C. E., "Bimatrix Equilibrium Points and Mathematical Programming", Management Science, Vol. 11, pp. 681-689 (1965).

2. Phillips, D. T., A. Ravindran, and J. J. Solberg, Operations Research: Principles and Practice, John Wiley & Sons, Inc., New York, 1976.

3. Ravindran, A., "A Computer Routine for Quadratic and Linear Programming Problems", Communications of the ACM, Vol. 15, No. 9, pp. 818-820 (1972).

4. Ravindran, A., "A Comparison of Primal Simplex and Complementary Pivot Method for Linear Programming", Naval Research Logistics Quarterly, Vol. 20, No. 1, pp. 95-100 (1973).

5. Ravindran, A., and Harvey Lee, "A Comparison of Five Algorithms for Solving Quadratic Programming Problems", Purdue Research Memorandum, School of Industrial Engineering, August 1976.