USER'S MANUAL FOR LPTØR
A FORTRAN IV LINEAR PROGRAMMING ROUTINE

Interim Report
May 1976

Prepared for:

Office of Water Research and Technology

Under Contract No. B-115-Colorado
Agreement No. 14-31-0001-5060

by

Torkil Jønch-Clausen
Research Assistant

and

Hubert J. Morel-Seytoux
Professor of Civil Engineering

CER75-76TJ-HJM36

TABLE OF CONTENTS

A. PURPOSE OF PROGRAM (ROUTINE)

The program LPTØR solves a Linear Programming (L.P.) problem using an algorithm known as the *General Differential Algorithm.* The general theory for this algorithm is presented in Wilde and Beightler (1967) and in more computational details by Morel-Seytoux (1972).

The reason for developing this Linear Programming algorithm instead of using the standard Simplex algorithm for Linear Programming is two-fold:

a. The General Differential Algorithm (G.D.A.) does not - as does the Simplex algorithm - require an initial basic feasible solution. An initial feasible solution suffices. The original intended use of this routine was for the solution of optimization problems in water resource systems (considering for example water reuse as part of possible alternative solutions). For this type of problems a feasible solution, namely the current design or operation of the existing system, is almost always at hand. Consequently, no search for an initial basic feasible solution (phase 1 in the Simplex procedure, described in Morel-Seytoux, 1972, lecture note No. 16, p.1) is required, with the result that the optimal solution can be attained, at least in principle, faster than by the available Simplex algorithm routines.

b. The program can be relatively easily extended to handle a Quadratic Programming problem (quadratic objective function subject to linear constraints). In fact the algorithm used for the solution of the L.P. problem is precisely what the Quadratic Programming algorithm (Morel-Seytoux, 1972, Lecture Notes 11-15) reduces to when the second order terms in the quadratic objective function are not present.

Minimizing costs in water resources systems by means of Linear Programming techniques assumes constant unit costs. However, unit costs normally decrease when the quantity of water being stored, transferred or treated is increased (economies of scale). A quadratic objective function assumes linear decrease of unit costs with quantity, and accordingly represents a better approximation to the real-world economic system.

B. DEFINITION OF A LINEAR PROGRAMMING PROBLEM

The routine was developed for the following (special) definition of Linear Programming, namely:

Minimize the objective function: $y = \sum_{n=1}^{N} c_n x_n$ (1)

subject to the equality constraints: $\sum_{n=1}^{N} a_{kn} x_n = r_k \quad , \quad k = 1,2,\ldots K_e$ (2)

and the inequality constraints: $\sum_{n=1}^{N} a_{kn} x_n \geqq r_k \quad , \quad k = K_e+1,\ldots K$ (3)

and the non-negativity conditions: $x_n \geqq 0$, for *some* of the x_n (4)

The x_n are the unknown variables of the problem and there are N of them. The c_n are the coefficients of the original variables of the problem, often representing average unit costs associated with storage, transfer or treatment of the water quantity or quality variables x_n. The a_{kn} are the coefficients of the original variables of the problem in the constraints. They often represent *technological* coefficients such as e.g. the *direct* coefficients of the *Processing sectors* in an economic Input-Output Analysis (e.g. Miernyk, 1965, pp. 21-28). The values and signs of the c_n, a_{kn} and r_k parameters are unrestricted.

The present definition of L.P. is *special* because it is limited to the problem of minimization. A maximization problem can be solved with the present routine by minimizing the negative of the objective to be maximized.

Also the special definition used here allows for the presence of *equalities* as well as *inequalities* among the constraints and of *free* variables (i.e. variables not subjected to the nonnegativity conditions but rather with permissible range from $-\infty$ to $+\infty$).

Starting from an initial feasible solution, i.e. a solution which satisfies all the constraints (2), (3) and (4), the algorithm obtains the optimal solution x_n^*, $n = 1,2,\ldots N$, for which the objective function has its minimum value.

The constraints can be any combination of equalities and inequalities. By appropriate rearrangement of terms any constraint can be expressed in the form (2) or (3).

It is assumed that a maximum of K (total number of constraints) out of the N variables x_n are free variables, i.e. variables which can take any value, negative, positive or zero. Consequently, at least N-K of the x_n *must* be non-negative.

C. BRIEF DESCRIPTION OF METHOD OF SOLUTION (ALGORITHM)

Starting with an initial feasible solution (provided by the *User*) the algorithm decreases the value of the objective function in an iterative process. At each step one variable changes value in such a way that the value of the objective function is decreased while keeping the constraints satisfied. The variable to be changed can be one of the original variables, or it can be one of the slack variables, i.e. an artificial non-negative variable:

$x_{K+k} = \sum_{n=1}^{N} a_{kn} x_n - r_k$, which expresses how "loose" the k^{th} constraint is.

The variable to be changed is the one which has the greatest single effect on the objective function at each step.

A set of conditions, known as the Kuhn-Tucker conditions, determines when the optimum has been reached. At the optimum, only insignificant decrease of the objective function will result from changing one of the variables. The *User* decides what he considers insignificant by providing the control parameter ε_y. When the decrease in y, Δy is $< \varepsilon_y$, Δy is considered insignificantly small. Other control parameters which must be supplied by the *User* are listed in the following Input Data description.

The user has a choice of several output options, ranging from printout of input data and final solution only, to complete printout for debugging purposes. The appropriate print-control parameters are listed in the following input description.

D. INPUT DATA DESCRIPTION

The user must provide the information necessary for formulating his Linear Programming problem in the form depicted in the equations (1) - (4) above. Further, he must provide an initial feasible solution, and his tolerance and output requirements.

The maximum size of problems to be handled by the program is pre-decided by the programmer. If the user tries to exceed this limit, a LIMIT EXCEEDED will be printed, together with information on what the present limitation is. Only by changing the COMMON and DIMENSION statements in the deck, can the maximum size problem be increased.

As of date: the maximum size problem is:

Maximum number of original variables, N_{max} = ...

Maximum number of constraints, K_{max} = ...

The input data is provided on punch-cards. The following types of cards are required:

Card A	Constants defining the problem dimensions. By providing only this card the user will be told whether he will exceed the limitation or not. Output will be PROBLEM DIMENSION O.K. or LIMIT EXCEEDED.
Cards B	Coefficients defining the objective function and types of variables (free or non-negative).
Cards C	Coefficients defining the left hand sides of the constraints.
Cards D	Right hand sides of the constraints.
Cards E	Initial feasible solution
Card F	Control parameters.
Card G	Output controls.

A detailed description of the program input follows.

INPUT DESCRIPTION

CARD	FIELD	COLUMNS	MATH. SYMBOL	FTN. SYMBOL	FORMAT	VALUE	DESCRIPTION
	1	1-4	N	N	I4	+	Number of original variables (total)
A	2	5-8	N_f	NF	I4		Number of free, original variables
	3	9-12	K	K	I4	+	Number of constraints (total)
	4	13-16	K_e	KE	I4		Number of equality constraints
	1	1-8	c_1	C(1)	G8.0	+ or -	Coefficient of the first variable, x_1 , in the
	2	9-10	-	ITYPE(1)	I2	0 or 1	objective function (e.g. cost associated with
	3	11-18	c_2	C(2)	G8.0	+ or -	x_1).
B	4	19-20	-	ITYPE(2)	I2	0 or 1	If x_1 is a non-negative variable, ITYPE(1) = 1.
	.	.	.	.	.	.	If x_1 is a free variable, ITYPE(1) = 0.
	.	.	.	.	.	.	Continue punching pairs C(I), ITYPE(I), in fields
	.	.	.	.	.	.	of 10, with a maximum of 8 pairs per card.
	.	.	.	.	.	.	There will be N such pairs.
			c_N	C(N)	G8.0	+ or -	
			-	ITYPE(N)	I2	0 or 1	
	1	1-8	a_{11}	A(1,1)	G8.0	+ or -	Coefficient of the first variable, x_1 , in the
	2	9-16	a_{12}	A(1,2)	G8.0	+ or -	first constraint.
							List constraints, the K_e equality being listed
C-1							first, the $K-K_e$ inequality constraints subse-
							quently.
	.	.	.	.	.	.	Inequality constraints must be written $\sum_{n=1}^{N} a_{kn} \cdot x_n \geq r_k$
	.	.	.	.	.	.	
	.	.	.	.	.	.	Coefficients a_{kn} should be of the same order of
	.	.	.	.	.	.	magnitude.
							First, punch the N coefficients a_{kn} of the
							- cont. -

CARD	FIELD	COLUMNS	MATH. SYMBOL	FTN. SYMBOL	FORMAT	VALUE	DESCRIPTION
	.	.	.	.	.	.	- cont. -
C-1	.	.	.	.	.	.	first constraint, in fields of 8, with a maximum
(cont)			a_{1N}	A(1,N)	G8.0	+ or -	of 10 per card.
	1	1-8	a_{21}	A(2,1)	G8.0	+ or -	Then, beginning with a new card, punch the N
	2	9-16	a_{22}	A(2,2)	G8.0	+ or -	coefficients a_{2n} of the second constraint, etc...,
C-2	.	.	.	.	.	.	
	.	.	.	.	.	.	
(etc.)	.	.	.	.	.	.	
	1	1-8	a_{K1}	A(K,1)	G8.0	+ or -	..... ending with the N coefficients a_{Kn} of
	2	9-16	a_{K2}	A(K,2)	G8.0	+ or -	the last constraint.
C-K	.	.	.	.	.	.	
	.	.	.	.	.	.	
	.	.	.	.	.	.	
	1	1-8	r_1	R(1)	G8.0	+ or -	Right hand sides of constraints.
	2	9-16	r_2	R(2)	G8.0	+ or -	Punch r-values, in fields of 8, with a maximum
	.	.	.	.	.	.	of 8 per card.
D	.	.	.	.	.	.	There will be K such values.
	.	.	.	.	.	.	
			r_K	R(K)	G8.0	+ or -	
	1	1-8	x_1^o	XØ(1)	G8.0	+ or -	Initial feasible solution, i.e. a solution
	2	9-16	x_2^o	XØ(2)	G8.0	+ or -	which satisfies the constraints.
E	.	.	.	.	.	.	x_1^o is the variable corresponding to c_1 of
	.	.	.	.	.	.	the objective function, x_2^o corr. to c_2, etc...,
	.	.	.	.	.	.	a total of N such variables
							- cont. -

CARD	FIELD	COLUMNS	MATH. SYMBOL	FTN. SYMBOL	FORMAT	VALUE	DESCRIPTION
E (cont.)	.	.	. x_N^o	. XØ(N)	. G8.0	. + or -	The type of the variable (non-negative or free) must correspond to the ITYPE specification given in cards B.
F	1	1-8	ε_y	EPSY	G8.0	+	Control parameter: Stop iteration when the change in the objective function $\Delta y < \varepsilon_y$.
	2	9-16	Δ	ACC	G8.0	+	Computational accuracy requirement: A quantity is set equal to zero when $q < \Delta \cdot \bar{q}$, where q is the quantity, $\bar{q}$ the order of magnitude of that quantity, and Δ the selected accuracy (say $\Delta = 0.001$).
	3	17-19	N_{max}	NIMAX	I3	+	Maximum number of iterations.
G	1	1-2		IPRINT	I2	+	Punch 0 if all intermediate results are to be printed (debug) Punch 1 if only input data, tables of correspondence with some frequency specified below (every 0, 1, 5 or 10 iterations), and optimum solution are to be printed. (The table of correspondence shows the values and roles, i.e. states or decisions, of the variables at each step).
	2	3-4		IFREQ	12	+	Punch 0 if no tables of correspondence are to be printed (i.e. only input data and optimum solution wanted). Select frequency of print-out of tables of correspondence by punching: 1 for printout of every step 5 " " " " 5 steps 10 " " " " 10 steps If IPRINT = 0 punch IFREQ = 1

E. ILLUSTRATIVE EXAMPLE

As an example to illustrate the use of the program let us consider the following optimization problem which is already in a form acceptable to LPTØR:

$$\text{Min} \quad \left\{ y = x_1 + 2x_2 + 3x_3 + 4x_4 \right\}$$

$$\begin{aligned} \text{subject to} \quad x_1 - x_3 &\geqq 3 \\ x_2 + x_3 &\geqq 4 \\ x_2 - x_4 &\geqq 1 \\ x_n &\geqq 0 \qquad \text{for all} \quad n = 1,2,3,4 \end{aligned}$$

An initial feasible solution is $(x_1, x_2, x_3, x_4) = (5,3,1,1)$. Given this initial feasible solution the user wants to find the solution which minimizes y, as well as the minimum value of y. He will be satisfied with a solution to the second decimal point, and he is interested in the final result only.

The input data for the problem are shown on the attached FORTRAN coding sheet (Table 1). The computer printout is displayed subsequently.

IBM FORTRAN Coding Form

Printed in U.S.A

PROGRAM: LPTØR / DATA FØR SAMPLE PRØBLEM

PROGRAMMER: T. JØNCH-CLAUSEN DATE: 3-30-76

PAGE 1 OF 1

```
   4   0   3   0
      1. 1      2. 1      3. 1      4. 1
      1.      0.     -1.      0.
      0.      1.      1.      0.
      0.      1.      0.     -1.
      3.      4.      1.
      5.      3.      1.      1.
    0.01   0.001 50
 1 0
```

*A standard card form, IBM electro 888157, is available for punching statements from this form

**Number of forms per pad may vary slightly

Table I. Input data coding sheet for illustrative example

```
********************************************************************************

PROBLEM DIMENSION O.K.

********************************************************************************

LINEAR PROGRAMMING

********************************************************************************

INPUT DATA

********************************************************************************


TOTAL NUMBER OF ORIGINAL VARIABLES N=      4
NUMBER OF FREE VARIABLES           NF=     0
TOTAL NUMBER OF CONSTRAINTS        K=      3
NUMBER OF EQUALITY CONSTRAINTS     KE=     2


INDEX        COEFFICIENTS    TYPE OF VARIABLE    INITIAL SOLUTION

  I            CA(I)            ITYPE(I)             XO(I)

  1            1.0000              1                 4.0000

  2            2.0000              1                 3.0000

  3            3.0000              1                 1.0000

  4            4.0000              1                 1.0000


CONSTRAINT     RIGHT HAND                 COEFFICIENT MATRIX OF CONSTRAINTS
NUMBER         SIDE

  I             P(I)                                      AA(I,J)
  1            3.0000     1.0000     0.         -1.0000    0.
  2            4.0000     0.         1.0000      1.0000    0.
  3            1.0000     0.         1.0000      0.        -1.0000

CONTROL PARAMETERS

EPSY=          .10000E-01
ACC=           .10000E-02
NIMAX=                 50
```

```
IPRINT=                                   1
IFREG=                                    0

****************************************************************************************************


INITIAL VALUE OF OBJECTIVE FUNCTION,Y=        17.000


****************************************************************************************************


OPTIMAL SOLUTION

I    X(I)
1  -3.0000
2   4.0000
3  0.
4  0.

****************************************************************************************************


MINIMUM VALUE OF OBJECTIVE FUNCTION,Y=        11.000

****************************************************************************************************


NUMBER OF ITERATIONS =      3

****************************************************************************************************


 06/07/76  CSU SCOPE 3.3.14  R C012 C013 C140 C141 05/02/76
13.22.08.TC845KV    FROM  AO       11A
13.22.08.TC845.  AEOG****.T30,CM60000,PR40.TORKIL
13.22.08..LINEAR PROGRAMMING.
13.22.08.FTN(L=0)
13.22.58.     11.854 CP SECONDS COMPILATION TIME
13.22.59.LDSET(PRESET=NGINF)
13.23.00.LGO.
13.23.05.FL= 025100 CP 00012.868SEC.  IO 00035.045SEC.
13.23.06.STOP
13.23.06.CP       13.160 SEC.
13.23.06.PP       69.763 SEC.
13.23.06.IO       35.095 SEC.
              4 PAGES PRINTED.
```

F. REFERENCES

Miernyk, W. H. 1965. "The Elements of Input-Output Analysis," Random House, New York, 156 pages.

Morel-Seytoux, H. J. 1972. "Foundations of Engineering Optimization," Lecture Notes, Department of Civil Engineering, Colorado State University, 1972, reprinted 1974, supplemented 1976, 130 pages.

Wilde, D. J. and C. S. Beightler. 1967. "Foundations of Optimization," Prentice-Hall, Inc. 480 pages.

APPENDIX 1

Flowchart

Notation The notation in the following flowchart follows as closely as possible the notation of the notes: "Foundations of Engineering Optimization" (H. J. Morel-Seytoux, 1972, reprinted 1974). However, in cases where the notation used in the program differs from that of the notes, the former is used in order to facilitate the reading of the program listing (appendix 3).

N = number of original variables

N_f = number of free variables

K = number of constraints

K_e = number of equality constraints

y = objective function (minimum value = y^*)

c_n = coefficients of the objective function

x_n = variables

x_n° = initial value of the variable x_n

a_{kn} = element of coefficient matrix of constraints

r_k = right hand side of k^{th} constraint

ITYPE( ) = array indicating the type of a variable

s_i = i^{th} state variable

d_j = j^{th} decision variable

b_{ki} = elements of the Jacobian matrix (B)

d_{kj} = elements of the coefficient matrix of the decision variables (D)

δ_{ij} = coefficients relating state and decision variables (elements of the DELTA = $B^{-1}\cdot D$ matrix)

v_j = constrained derivative of the objective function with with respect to the non-negative decision variable d_j

γ_{dj} = coefficient in the objective function of the n^{th} variable playing the role of the j^{th} decision variable. In other words $\gamma_{dj} = c_n$ where $n = n_d(j)$ and $n_d(j)$ is the array of correspondence between the numbering of the original variables and the role they play as decision variables. For example if $n_d(3) = 7$ it means that the 7^{th} original variable is the 3^{rd} decision variable.

γ_{si} = coefficient in the objective function of the n^{th} variable playing the role of the i^{th} state variable.

jmax = particular j-value, corresponds to p in the notes

imin = particular i-value, corresponds to r in the notes

superscript $^{\circ}$ refers to old (previous) values

superscript $^{\nu}$ refers to new values

In addition to the above listed general symbols a number of auxiliary symbols are defined and used in the flow chart.

Flowchart

START

(1) Problem: Minimize $y = \sum_{n=1}^{N} c_n x_n$

subject to $\sum_{n=1}^{N} a_{kn} x_n = r_k$, $k = 1, K_e$, and

$\sum_{n=1}^{N} a_{kn}\ x_n \geqq r_k$, $k = K_e+1$, K

The problem has N original variables, N_f of which are free ($N-N_f$ non-negative), and K constraints, K_e of which are equality constraints (listed first).

(2) Input data according to the input description, i.e. provide c_n, a_{kn}, r_k, N, N_f, K, K_e, ITYPE(x_n) (0's for free variables, and 1's for non-negative variables), as well as the control and output control parameters.

(3) Input an initial feasible solution, $(x_1^\circ, x_2^\circ, \ldots, x_N^\circ)$

(4) Define zero-slack variables of the K_e equality constraints and calculate the non-negative slack variables of the inequality constraints:

$$x_{N+k} = 0 \qquad k = 1, K_e$$

$$x_{N+K_e+k} = \sum_{n=1}^{N} a_{kn} x_n - r_k \qquad k = 1, K-K_e$$

(5) Define problem in augmented form, i.e. augment c_n-vector, and a_{kn}-matrix to obtain the formulation:

minimize $y = \sum_{n=1}^{N+K} c_n x_n$

subject to $\sum_{n=1}^{N+K} a_{kn} x_n = r_k$, $k = 1, K$

(6)

(5)

(6) Partition the x°-vector into K state variables, s_i, and N decision variables, d_j, by the criteria:

1. Free variables are selected as state variables
2. Non-negative variables of largest numerical value are selected as the remaining state variables (going from low to high index in case of equally large numerical values)
3. Zero-slack variables of equality constraints are selected as decision variables
4. Remaining non-negative variables are selected as decision variables.

Thus arrays $n_s(i)(s_i)$ and $n_d(j)(d_j)$ are created.(The i^{th} state variable s_i is defined as $x[n_s(i)]$, the j^{th} decision variable d_j is defined as $x[n_d(j)]$).

(7) Define the KxK coefficient matrix of the state variables,(the Jacobian) b_{ki}, in which i (column) corresponds to the index of the state variable $(b_{ki} = a_{k,n_s(i)})$. Define the KxN coefficient matrix of the decision variables, d_{kj}, in which j corresponds to the index of the decision variable $d_{kj} = a_{k,n_d(j)}$.

(8) After partition the system $\sum_{n=1}^{N+K} a_{kn} x_n = r_k$ can be written:

$$\sum_{i=1}^{K} b_{ki} s_i = r_k - \sum_{j=1}^{N} d_{kj} d_j \qquad k = 1,K$$

In matrix notation: $B \cdot \underline{s} = \underline{r} - D\,\underline{d}$

When by Gaussian elimination procedures the matrix B is reduced to the identity matrix, the matrix D will be reduced to the matrix DELTA (= $B^{-1} \cdot D$) of elements δ_{ij}. Thus the δ_{ij}-coefficients are found by calling the Gaussian elimination subroutine GAUSS (see appendix 2), with B as the coefficient matrix, and D as the matrix of right hand sides.

(9)

(8)

9 If the Jacobian B is singular, change partition by simplexing a state and a decision variable, starting with the highest indices. (i.e. first simplex s_K and d_N, then if also this partition results in a singular Jacobian, s_{K-1} and d_{N-1}, etc.)
After simplexing, GO TO 7.
If one of the indices is zero, STOP.
If and when the Jacobian B is non-singular, GO TO 10

10 Calculate the constrained derivatives, v_j, with respect to the non-negative decision variables, d_j:

$$v_j = \gamma_{dj} + \sum_{i=1}^{K} \gamma_{si}\,\delta_{ij} \qquad j = K_e+1, N$$

Here, γ_{dj} and γ_{si} refer to the coefficients c_j of the objective function, partitioned according to the partition of the variables (see page 1: Notation)

↓

11 Check Kuhn-Tucker conditions, i.e.

Is $\left\{ v_j = 0 \;\middle|\; d_j > 0 \quad \text{or} \quad v_j > 0 \;\middle|\; d_j = 0 \right\}$ for all $j = K_e+1, N$?

If the K.T. conditions are satisfied, calculate $y^* = \sum_{n=1}^{N} c_n x_n$,
print final results, $(x_1^*, x_2^*, \ldots, x_N^*)$ and y^*, and STOP.
If the K.T. conditions are not satisfied, GO TO 12

12 Find $\max_j \left\{ v_j \;\middle|\; v_j > 0,\ d_j > 0 \right\} = v^+_{j,max} = v_{jmax^+}$

and $\max_j \left\{ -v_j \;\middle|\; v_j < 0 \right\} = -v^-_{j,max} = v_{jmax^-}$

If $v^+_{j,max} > -v^-_{j,max}$: $v_{j,max} = v^+_{j,max}$, $jmax = jmax^+$ GO TO 13

If $-v^-_{j,max} > v^+_{j,max}$: $-v_{j,max} = -v^-_{j,max}$, $jmax = jmax^-$ GO TO 15

(13)

If $\delta_{i,jmax} \leq 0$ for all $i = 1,K$, GO TO 19

If $\delta_{i,jmax} > 0$ for some i, $i = 1,K$, GO TO 14

(14)

For all i's for which $\delta_{i,jmax} > 0$, and for which s_i's are non-negative variables, calculate $\dfrac{s_i}{\delta_{i,jmax}}$

Find $\min\limits_{i} \left\{ \dfrac{s_i}{\delta_{i,jmax}} \right\} = \dfrac{s_{imin}}{\delta_{imin,jmax}} = amin$

If $d_{jmax} > amin$ GO TO 16

If $d_{jmax} \leq amin$ GO TO 19

(15)

If $\delta_{i,jmax} \geq 0$ for all i, $i = 1,K$, the problem is poorly posed. STOP.

For all i's for which $\delta_{i,jmax} < 0$, and for which s_i's are non-negative variables, calculate $\dfrac{s_i}{\delta_{i,jmax}}$

Find $\min\limits_{i} \left\{ \dfrac{s_i}{\delta_{i,jmax}} \right\} = \dfrac{s_{imin}}{\delta_{imin,jmax}} = amin$ GO TO 16

(16)

Change partition: simplex d°_{jmax} and s°_{imin}

Calculate new state variables:

$$s^{\nu}_{i} = s^{\circ}_{i} - \delta^{\circ}_{i,jmax} \cdot amin \quad , \; i \neq imin, \; i = 1,K$$

$$s^{\nu}_{imin} = d^{\circ}_{jmax} - amin \quad , \; i = imin$$

$$d^{\nu}_{jmax} = 0 \quad j = jmax$$

$$d^{\nu}_{j} = d^{\circ}_{j} \text{ (i.e. no change)} \quad , \; j \neq jmax, \; j = 1,N$$

↓

(17)

(16)

(17) Calculate new δ^{ν}_{ij} corresponding to the new partition:

Define: $z_i = \delta^{\circ}_{i,jmax}$, $zz_j = \delta^{\circ}_{imin,j}$, $\delta rp = \delta^{\circ}_{imin,jmax}$

$$\delta^{\nu}_{i,jmax} = \frac{z_i}{\delta rp} \quad \delta^{\nu}_{imin,j} = \frac{zz_j}{\delta rp} \quad \delta^{\nu}_{imin,jmax} = \frac{1}{\delta rp}$$

$$\delta^{\nu}_{ij} = \delta^{\circ}_{ij} - \frac{z_i \cdot zz_j}{\delta rp} \;, \quad i \neq imin, \; j \neq jmax, \quad i = 1,K, \quad j = 1,N$$

(18) Calculate new constrained derivatives, v^{ν}_j , with respect to the non-negative decision variables d^{ν}_j:

$$v^{\nu}_j = v^{\circ}_j - v^{\circ}_{jmax} \frac{zz_j}{\delta rp} \;, \quad j \neq jmax \;, \quad j = K_e + 1,N$$

$$v^{\nu}_{jmax} = \frac{v^{\circ}_{jmax}}{\delta rp} \qquad j = jmax$$

GO TO 11 (Kuhn-Tucker conditions)

(19) Without changing partition, calculate new state variables:

$$s^{\nu}_i = s^{\circ}_i - \delta^{\circ}_{ijmax} \cdot d^{\circ}_{jmax} \;, \quad i = 1,K$$

Define new decision variables:

$$d^{\nu}_{jmax} = 0 \qquad j = jmax$$

$$d^{\nu}_j = d^{\circ}_j \quad \text{(i.e. no change)} \; j \neq jmax, \quad j = 1,N$$

GO TO 11 (Kuhn-Tucker conditions)

END

APPENDIX 2

Special Subroutines

Except for the subroutine in which the Gaussian elimination procedure is performed, none of the subroutines used in Program LPTØR have general applications, and accordingly only subroutine GAUSS shall be documented here. In addition to its application as a subprogram in LPTØR, independent calling programs for using GAUSS for solving systems of linear equations, or for inverting matrices shall be mentioned.

Subroutine GAUSS

Purpose Reduction by the Gaussian elimination procedure of a square, non-singular matrix to the identity matrix, simultaneously performing the same operations on any number of right hand sides. If desired, the determinant can be calculated.

Call statement

CALL GAUSS (N, A, EPSA, RR, DET, INDEXR, X)

Definition of input variables

N = Order of the square matrix A

A = Square N x N matrix

EPSA = Tolerance parameter. By definition, if A(I,J).LT. EPSA, then A(I,J).EQ. O.

RR = N x INDEXR matrix, the columns of which are the INDEXR right hand sides.

DET = Determinant of the matrix A

INDEXR = Number of right hand sides

Definition of output variables

X = N x INDEXR solution matrix. Any column of X is that solution of the system of linear equations which corresponds to the right hand side provided by that column.

DET : DET = 0 is returned of A is a singular matrix

Variables transferred by COMMON statement

COMMON / CONST 9 / IDET, KPRINT

IDET = Control variable. If IDET = 1 the determinant is not computed. If IDET = 1 the determinant is computed.

KPRINT: Output control. If KPRINT = 1 only input data and results, X, (DET), will be printed. If KPRINT ≠ 1 all intermediate results will be printed (debugging)

Mathematical operations

The matrix A of elements a_{ij} is reduced to a diagonal matrix by the Gaussian elimination procedure. First the matrix is reduced to an upper triangular matrix (forward elimination): at the end of the k^{th} step the coefficients of the system will be given by the formulae:

$$m_{ik} = \frac{a_{ik}^{k-1}}{a_{kk}^{k-1}} \qquad i = k+1,\ldots,N \qquad k = 1,2,\ldots,N$$

$$a_{ij}^{k} = a_{ij}^{k-1} - m_{ik} \cdot a_{kj}^{k-1} \qquad i = k+1,\ldots,N \qquad j = k+1,\ldots,N \qquad k = 1,\ldots,N$$

Similarly the right hand sides, r_i, transform by the formula:

$$r_i^{k} = r_i^{k-1} - m_{ik} \cdot r_k^{k-1} \qquad i = k+1,\ldots,N \qquad k = 1,\ldots,N$$

Secondly, the matrix is reduced to a diagonal matrix (backward elimination): at the end of the k^{th} step the coefficients of the system will be given by the formulae:

$$m_{ik} = \frac{a_{i,N-k}^{k-1}}{a_{N-k,N-k}^{k-1}} \qquad i = 1,2,\ldots,N-k-1 \qquad k = 1,\ldots,N$$

$$a_{ij}^{k} = a_{ij}^{k-1} - m_{ik} \cdot a_{N-k,j}^{k-1} \qquad i = 1,2..,N-k-1 \qquad j = 1,2,..,N-k-1 \qquad k = 1,..,N$$

$$r_i^{k} = r_i^{k-1} - m_{ik} \cdot r_{N-k}^{k-1} \qquad i = 1,2,..,N-k-1 \qquad k = 1,..,N$$

If in the process of forward elimination a pivot element a_{kk} is zero the k^{th} row is permuted with the first row i, $i = k+1,..,N$, in which a_{ik} is non-zero. If no such row can be found the matrix is singular and operations are stopped.

The multipliers m_{ik} are stored in the matrix in the positions a_{ik}.

Program GAUSS 1

Purpose: Solution of a system of N linear equations in N unknown for any number of right hand sides.

Operation: Program GAUSS 1 reads data and prints the solution in matrix form (replaces the matrix of right hand sides by the matrix of corresponding solutions). Subroutine GAUSS performs the Gaussian elimination (see description of subroutine GAUSS).

Input description: Input data for GAUSS 1 are provided on punch-cards. Following types of cards are required:

Card A: Constants defining the problem dimensions, and control and output control parameters.

Cards B: Elements of the coefficient matrix

Cards C: Right hand sides

A detailed input description for GAUSS 1 is given below.

Input Description for Program GAUSS 1

CARD	FIELD	COLUMNS	MATH. SYMBOL	FTN. SYMBOL	FORMAT	VALUE	DESCRIPTION
A	1	1-3	N	N	I3	+	Number of rows = number of columns = N
	2	4-6	ir	INDEXR	I3	+	Number of right hand sides
	3	7-9		IDET	I3	+	IDET = 1 : Determinant is not wanted IDET $\neq$ 1 : Compute and print determinant
	4	10-12		KPRINT	I3	+	KPRINT = 1 : Print only input and results KPRINT $\neq$ 1 : Print all intermediate results
	5	13-24	ε_a	EPSA	G12.5	+	Tolerance parameter: $a < \varepsilon_a \Rightarrow a = 0$, where a is an element of matrix A
B-1	1	1-8	a_{11}	A(1,1)	G8.0	+ or -	First element in the coefficient matrix
	2	9-16	a_{12}	A(1,2)	G8.0	+ or -	Second element, first row...
			$\vdots$	$\vdots$			$\vdots$
			a_{1n}	A(1,N)	G8.0	+ or -	N^{th} element, First row.
B-I	1	1-8	a_{i1}	A(I,1)	G8.0	+ or -	Punch elements in fields of 8 with a maximum of 10 per card.
	2	9-16	a_{i2}	A(I,2)	G8.0	+ or -	Start with a new card when starting a new row.
			$\vdots$	$\vdots$			
			a_{in}	A(I,N)	G8.0	+ or -	

CARD	FIELD	COLUMNS	MATH. SYMBOL	FTN. SYMBOL	FORMAT	VALUE	DESCRIPTION
C-1	1	1-8	r_{11}	RR(1,1)	G8.0	+ or -	First element of First right hand side
	2	9-16	r_{12}	RR(1,2)	G8.0	+ or -	First element of Second right hand side
			$r_{1,ir}$	RR(1,INDEXR)	G8.0	+ or -	First element of INDEXRth right hand side
C-I			r_{i1}	RR(I,1)	G8.0	+ or -	Ith element of First right hand side
			r_{i2}	RR(I,2)	G8.0	+ or -	Ith element of Second right hand side
			$r_{i,ir}$	RR(I,INDEXR)	G8.0	+ or -	Ith element of INDEXRth right hand side
							Punch elements in Fields of 8 with a maximum of 10 per card. Organize the right hand sides in a matrix, the columns of which are the right hand sides (INDEXR columns, N rows), starting with a new card when starting a new row.

C-N

Program INVERS

Purpose: Inversion of a non-singular matrix

Operation: Program INVERS reads data and prints the inverse. A special subroutine (INVTØR) is called which defines the system of right hand sides as the identity matrix. This system is subsequently solved by calling the Gaussian elimination subroutine GAUSS (see description of subroutine GAUSS).

Input description: Input data for INVERS are provided on punch-cards. Following types of cards are required:

Card A: Constants defining the problem dimensions, and control and output control parameters.

Cards B: Elements of matrix to be inverted.

A detailed input description for INVERS is given below.

Input Description for Program INVERS

CARD	FIELD	COLUMNS	MATH. SYMBOL	FTN. SYMBOL	FORMAT	VALUE	DESCRIPTION
	1	1-3	N	N	I3	+	Number of rows = number of colunms = N
	2	4-6		IDET	I3	+	IDET = 1 : Determinant is not wanted IDET $\neq$ 1 : Compute and print determinant
A	3	7-9		KPRINT	I3	+	KPRINT = 1 : Print only input and results KPRINT $\neq$ 1 : Print all intermediate results
	4	10-21	ε_a	EPSA	G12.5	+	Tolerance parameter : $a < \varepsilon_a \to a = 0$ where a is an element of matrix A.
	1	1-8	a_{11}	A(1,1)	G8.0	+ or -	First element in the n x n matrix to be inverted
	2	9-16	a_{12}	A(1,2)	G8.0	+ or -	Second element, first row
B-1	. . .		. . .				
			a_{1n}	A(1,N)	G8.0	+ or -	N^{th} element, first row.
	1	1-8	a_{I1}	A(I,1)	G8.0	+ or -	First element, i^{th} row
	2	9-16	a_{I2}	A(I,2)	G8.0	+ or -	
B-I	. . . .		. . . .				Punch elements in Fields of 8, maximum 10 per card. Start with a new card when starting a new row.
			a_{In}	A(I,N)	G8.0	+ or -	

APPENDIX 3

Program Listing

FORTRAN SYMBOLS

Mathematical Symbol	FORTRAN Symbol
N	N
N_f	NF
K	K
K_e	KE
y	Y
c_n	CA(I)
x_n	X(I)
x_n°	XØ(I)
a_{kn}	AA(I,J)
r_k	R(I)
s_i	X(NS(I))
d_j	X(ND(J))
b_{ki}	B(I,J)
d_{kj}	D(I,J)
δ_{ij}	DELTA(I,J)
v_j	V(J)
γ_{dj}	CA(ND(J))
γ_{si}	CA(NS(I))
jmax	JMAX
imin	IMIN

```
      PROGRAM LPTOR(INPUT,OUTPUT)
C*****************************************************************************
C
C     LINEAR PROGRAMMING ALGORITHM
C
C     CODED BY TORKIL JONCH-CLAUSEN, DECEMBER 1975.
C
C     THE LINEAR PROGRAMMING PROBLEM IS SOLVED BY THE GENERAL DIFFERENTIAL ALGO-
C     RITHM, AS OUTLINED BY H. MOREL-SEYTOUX IN CLASS NOTES FOR THE COURSE :
C     OPTIMIZATION IN HYDROLOGY AND WATER RESOURCES 1976, AT COLORADO STATE UNI-
C     VERSITY.
C
C     INPUT REQUIREMENTS :
C     N        : TOTAL NUMBER OF ORIGINAL VARIABLES
C     K        : TOTAL NUMBER OF CONSTRAINTS
C     NF       : NUMBER OF FREE VARIABLES
C     KE       : NUMBER OF EQUALITY CONSTRAINTS
C     CA(I)    : COEFFICIENTS OF OBJECTIVE FUNCTION, I=1,N
C     ITYPE(I) : TYPE OF VARIABLE X(I) : I=0 FOR FREE VARIABLES
C                                        I=1 FOR NON-NEGATIVE VARIABLES
C     AA(I,J)  : COEFFICIENTS OF CONSTRAINTS, I=CONSTRAINT NUMBER, I=1,K
C                                        J=VARIABLE NUMBER, J=1,N
C     R(I)     : RIGHT HAND SIDES OF CONSTRAINTS
C     XO(I)    : VARIABLES IN INITIAL SOLUTION
C     EPSY     : TOLERANCE PARAMETER : IF THE CHANGE IN Y IN THREE SUBSEQUENT ITE
C                RATIONS IS LESS THAN EPSY, THE OPTIMUM IS REACHED
C     ACC      : TOLERANCE PARAMETER DETERMINING THE CALCULATION ACCURACY :
C                A IS DEFINED TO BE ZERO IF A.LT.ACHAR*ACC, WHERE ACHAR IS THE OR
C                DER OF MAGNITUDE OF THE QUANTITY A
C     NIMAX    : MAXIMUM NUMBER OF ITERATIONS
C     IPRINT   : IPRINT=1 : ONLY INPUT, RESULT AND TABLES OF CORRESPONDANCE WITH
C                           FREQUENCY IFREQ WILL BE PRINTED
C                IPRINT=0 : DEBUGGING OUTPUT
C     IFREQ    : IFREQ=0,1,5 OR 10 : FREQUENCY OF TABLES OF CORRESPONDANCE
C
C     FOR THE NORMAL USER, INTERESTED IN RESULTS ONLY, INPUT :
C     EPSY=0, ACC=0.001, NIMAX=50, IPRINT=1, IFREQ=0
C
C     THE PROGRAM IS STILL SUBJECT TO MINOR REVISIONS AND IMPROVEMENTS.
C*****************************************************************************
C
      LOGICAL KT,IVPOS,ICASE1
      COMMON/CONST 1/N,NF,K,KE
      COMMON/CON 1/NSTAR
      COMMON/CONST 7/IPRINT
      COMMON/CONST 8/KOUNT,NIMAX
C
C     READ AND PRINT INPUT DATA
C     FORMULATE AND PRINT STANDARD FORMULATION
C
      CALL READIN
C
C     PARTITION VARIABLES, AND PRINT TABLE OF CORRESPONDENCE
C
      CALL PART
C
C      PARTITION AND PRINT MATRICES OF COEFFICIENTS
C
      CALL PART AA
C
C     CALCULATE DELTA COEFFICIENTS
C
      LS=K+1
      LD=N+1
      CALL IDELTA(LS,LD)
C
C     CALCULATE CONSTRAINED DERIVATIVES
C
      CALL CONDER
C
C     CHECK KUHN-TUCKER CONDITIONS, AND PRINT OPTIMAL SOLUTION IF SATISFIED
C
C     INITIALIZE ITERATION COUNTER
C
      NSTAR=1
      KOUNT=0
      CALL KUNTUC(KT)
      IF(KT) STOP
   10 CONTINUE
```

```
      IVPOS=.FALSE.
      ICASE1=.FALSE.
      JMAX=0
      IMIN=0
      AMIN=0.0
C
C     FIND NUMERICALLY LARGEST CONSTRAINED DERIVATIVE
C     IF POSITIVE, IVPOS=.TRUE., IF NEGATIVE, IVPOS=.FALSE.
C
      CALL MAXV(IVPOS,JMAX)
      IF(.NOT.IVPOS) GO TO 11
C
C     MAXV IS POSITIVE (CASA A1) -- CHECK WHETHER A DECISION OR A STATE GOES TC
C     ZERO FIRST
C
      CALL CASEA1(ICASE1,JMAX,IMIN,AMIN)
      IF(.NOT.ICASE1) GO TO 12
C
C     A DECISION GOES TO ZERO (CASE A1,B1) -- CHECK KUHN-TUCKER CONDITIONS, ANC
C     PRINT OPTIMAL SOLUTION IF SATISFIED
C
      CALL CASEB1(JMAX,KT)
      IF(KT) STOP
      GO TO 10
C
   12 CONTINUE
C
C     A STATE GOES TO ZERO (CASE A1,B3) -- CHANGE PARTITION, CALCULATE NEW CON-
C     STRAINED DERIVATIVES, AND CHECK THE KUHN-TUCKER CONDITIONS.
C     PRINT OPTIMAL SOLUTION IF KUHN-TUCKER CONDITIONS SATISFIED
C
      CALL CASEB3(KT,JMAX,IMIN,AMIN)
      IF(KT) STOP
      GO TO 10
C
C     MAXV IS NEGATIVE(CASE A2)--CHANGE PARTITION, CALCULATE NEW CONSTRAINED DE
C     RIVATIVES, AND CHECK KUHN-TUCKER CONDITIONS.
C     IF KUHN-TUCKER CONDITIONS SATISFIED, PRINT OPTIMAL SOLUTION
C
   11 CONTINUE
      CALL CASEA2(JMAX,KT)
      IF(KT) STOP
      GO TO 10
      END
C
C
C
      SUBROUTINE READIN
C
C*****************************************************************************
C     THIS SUBROUTINE DOES THE FOLLOWING :
C             1.  READS AND PRINTS INPUT DATA
C             2.  FORMULATE LINEAR PROGRAMMING PROBLEM IN STANDARD FORM
C             3.  PRINT STANDARD FORMULATION
C*****************************************************************************
C
      DIMENSION XO(15),Z(15,25)
      COMMON/CONST 1/N,NF,K,KE
      COMMON/CONST 2/N1,N2,N3,N4,N5,N6
      COMMON/CONST 3/EPSY,EPSV,EPSCO,EPSD
      COMMON/KONST 1/IFREQ
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
      COMMON/CONST 8/KOUNT,NIMAX
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 2/X(25),ITYPE(15)
      PRINT 9999
C
C     ....READ PROBLEM DIMENSIONS....
      READ 100, N,NF,K,KE
  100 FORMAT(4I4)
C
C     ....CHECK PROBLEM DIMENSIONS....
      IF(N.GT.25.OR.K.GT.15) PRINT 97
   97 FORMAT(1H1,T5,*LIMIT EXCEEDED*,//T5,*NMAX=25,KMAX=15*)
      IF(N.GT.25.OR.K.GT.15) STOP
      PRINT 9997
 9997 FORMAT(T5,*PROBLEM DIMENSION O.K.*)
```

```
C     ....READ DATA....
      READ 101,(CA(I),ITYPE(I),I=1,N)
  101 FORMAT(8(G8.0,I2))
      DO 10 I=1,K
   10 READ 102, (AA(I,J),J=1,N)
  102 FORMAT(10G8.0)
      READ 103,(R(I),I=1,K)
  103 FORMAT(10G8.0)
      READ 104,(XO(I),I=1,N)
  104 FORMAT (10G8.0)
      READ 105, EPSY,ACC,NIMAX
  105 FORMAT(2G8.0,I3)
      READ 106, IPRINT,IFREQ
  106 FORMAT(2I2)
C
C
C     ....DEFINE TOLERANCE PARAMETERS....
      NN=MIN0(10,N)
      SUMC=0.0
      DO 96 I=1,NN
   96 SUMC=SUMC+CA(I)
      AVERC=SUMC/FLOAT(NN)
      EPSV=ACC*AVERC
C
      SUMX=0.0
      DO 95 I=1,NN
   95 SUMX=SUMX+XO(I)
      AVERX=SUMX/FLOAT(NN)
      EPSD=ACC*AVERX
      N1=N+1
      N2=N+K
      N3=N+KE
      N4=KF+1
      N5=N+KF+1
      N6=(N+K)-(NF+KE)
C
C     PRINT INPUT DATA
C
      PRINT 9999
 9999 FORMAT(///T5,120(1H*),///)
      PRINT 9998
 9998 FORMAT(T5,*LINEAR PROGRAMMING*)
      PRINT 9999
      PRINT 199
  199 FORMAT(1H1,T5,*INPUT DATA*)
      PRINT 9999
      PRINT 200,N,NF,K,KE
  200 FORMAT(////T5,*TOTAL NUMBER OF ORIGINAL VARIABLES*,T40,*N=*,T50,
     1I3//T5,*NUMBER OF FREE VARIABLES*,T40,*NF=*,T50,I3,//T5,*TOTAL NUM
     $BER OF CONSTRAINTS*,T40,*K=*,T50,I3//T5,*NUMBER OF EQUALITY CONSTR
     3AINTS*,T40,*KE=*,T50,I3)
      PRINT 201
  201 FORMAT(////T5,*INDEX*,T28,*COEFFICIENTS*,T45,*TYPE OF VARIABLE*,
     1T65,*INITIAL SOLUTION*)
      PRINT 202
  202 FORMAT(//T7,*I*,T31,*CA(I)*,T49,*ITYPE(I)*,T71,*XO(I)*)
       PRINT 210, (I,CA(I),ITYPE(I),XO(I),I=1,N)
  210 FORMAT(//T5,I3,T28,G12.5,T52,I1,T67,G12.5)
      PRINT 237
  237 FORMAT(1X,////)
      PRINT 203
      PRINT 209
  209 FORMAT(//T8,*I*,T34,*R(I)*,T86,*AA(I,J)*)
      DO 20 I=1,K
   20 PRINT 204, I,R(I),(AA(I,J),J=1,N)
  203 FORMAT(1X,T5,*CONSTRAINT*,T30,*RIGHT HAND*,T70,*COEFFICIENT MATRIX
     $ OF CONSTRAINTS*/T5,*NUMBER*,T30,*SIDE*)
  204 FORMAT(1H0,T6,I3,T29,G12.5,(/T45,7G12.5))
      PRINT 205, EPSY,ACC,NIMAX
  205 FORMAT(///T5,*CONTROL PARAMETERS*,//T5,*EPSY=*,T30,G12.5,/T5,*ACC=
     $*,T30,G12.5,/T5,*NIMAX=*,T40,I3)
      PRINT 206, IPRINT,IFREQ
  206 FORMAT(///T5,*PRINT CONTROLS*,//T5,*IPRINT=*,T40,I3,/T5,*IFREQ=*,
     $T40,I3)
      PRINT 9999
C
C     DEFINE AUGMENTED COEFFICIENT VECTOR OF OBJECTIVE FUNCTION, CA(I).
C
      DO 31 I=N1,N2
   31 CA(I)=0.0
C     DEFINE AUGMENTED COEFFICIENT MATRIX AA(I,J)
      DO 40 I=1,K
      DO 42 J=N1,N2
      IF(J.EQ.N+I) AA(I,J)=-1.0
      IF(J.NE.N+I) AA(I,J)=0.0
   42 CONTINUE
   40 CONTINUE
C
C     CALCULATE SLACK VARIABLES ASSOCIATED WITH INEQUALITY CONSTRAINTS
C     DEFINE ARRAY OF VARIABLES, ORIGINAL AND SLACK
C
C     ORIGINAL VARIABLES
C
      DO 50 I=1,N
   50 X(I)=XO(I)
C
C     SLACK VARIABLES, EQUALITY CONSTRAINTS
C
      DO 60 I=N1,N3
   60 X(I)=0.0
C
C     SLACK VARIABLES, INEQUALITY CONSTRAINTS
C
      IF(KF.EQ.K) GO TO 1000
      DO 70 I=N4,K
      J=1
      Z(I,J)=0.0
      DO 71 J=1,N
      J1=J+1
   71 Z(I,J1)=Z(I,J)+AA(I,J)*X(J)
      IK=N+I
      X(IK)=Z(I,J)-R(I)
   70 CONTINUE
 1000 CONTINUE
C
C     PRINT STANDARD FORMULATION
C
      IF(IPRINT.EQ.1) GO TO 2607
      PRINT 300
  300 FORMAT(1H1,T5,*INPUT DATA IN STANDARD FORMULATION*)
      PRINT 301
  301 FORMAT(1X,///,T5,*INDEX*,T25,*OBJECTIVE FUNCTION*,T45,*TYPE OF VAR
     $IABLE*,T65,*INITIAL SOLUTION*,/T28,*COEFFICIENTS*,//,T7,*I*,T31,*C
     $A(I)*,T49,*ITYPE(I)*,T71,*X(I)*)
      PRINT 302, (I,CA(I),ITYPE(I),X(I),I=1,N)
  302 FORMAT (1X,/,(/T4,I4,T28,G12.5,T52,I1,T68,G12.5))
      PRINT 303, (I,CA(I),X(I),I=N1,N2)
  303 FORMAT(1X,T4,I4,T28,G12.5,T68,G12.5)
      PRINT 304
  304 FORMAT(1H0,//,T5,*CONSTRAINT*,T30,*RIGHT HAND*,T70,*COEFFICIENT MA
     $TRIX OF CONSTRAINTS*/T5,*NUMBER*,T30,*SIDE*,/T7,*I*,T33,*R(I)*,
     $T85,*AA(I,J)*)
      DO 90 I=1,K
   90 PRINT 305,I,R(I),(AA(I,J),J=1,N2)
  305 FORMAT(1H0,T5,I3,T29,G12.5,(/T45,7G12.5))
C
 2607 CONTINUE
      RETURN
      END

C
      SUBROUTINE PART
C
C**************************************************************************
C     THIS SUBROUTINE MAKES THE PARTITION OF X(I) INTO :
C              1.  K STATE VARIABLES, NS(I), FREE + NON.NEG
C              2.  N DECISION VARIABLES, ND(J), ZEROS + NON.NEG
C**************************************************************************
C
      LOGICAL INS
      DIMENSION JMAX1(15)
      COMMON/CONST 1/N,NF,K,KE
      COMMON/CONST 2/N1,N2,N3,N4,N5,N6
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
```

```
C     DEFINE ARRAY OF NON.NEG VARIABLES,NN(J)
C
      J=1
      I=1
      DO 110 L=1,N
      IF(ITYPE(L).EQ.0) NS(I)=L
      IF(ITYPE(L).EQ.0) I=I+1
      IF(ITYPE(L).EQ.1) NN(J)=L
      IF(ITYPE(L).EQ.1) J=J+1
  110 CONTINUE
      DO 111 L=N5,N2
      NN(J)=L
      J=J+1
  111 CONTINUE
C
C     SELECT NON.NEG VARIABLES OF LARGEST NUMERICAL VALUE TO BE STATE
C     VARIABLES
      JJ=1
      JMAX1(1)=0
  113 CONTINUE
      XMAX=0.0
      JMAX=0
      DO 112 L=1,N6
      INS=.FALSE.
      DO 109 IJ=1,JJ
      IF(NN(L).EQ.JMAX1(IJ)) INS=.TRUE.
  109 CONTINUE
      IF(INS) GO TO 114
      IF(X(NN(L)).GE.XMAX) GO TO 115
      GO TO 114
  115 CONTINUE
      XMAX=X(NN(L))
      JMAX=NN(L)
  114 CONTINUE
  112 CONTINUE
      NS(I)=JMAX
      JMAX1(JJ)=JMAX
      JJ=JJ+1
      I=I+1
      IF(I.LE.K)  GO TO 113
      CONTINUE
C
C     SELECT REMAINING VARIABLES TO BE DECISION VARIABLES
C
C     SELECT SLACK VARIABLES OF EQUALITY CONSTRAINTS ( ALL ZEROS) TO
C     BE DECISION VARIABLES
      J=1
      IF(KE.EQ.0) GO TO 129
      DO 120 L=N1,N3
      ND(J)=L
      J=J+1
  120 CONTINUE
  129 CONTINUE
C
C     SELECT REMAINING NON.NEG VARIABLES TO BE DECISION VARIABLES
C
      DO 121 L=1,N6
      DO 122 I=1,K
      IF(NN(L).EQ.NS(I)) GO TO 121
  122 CONTINUE
      ND(J)=NN(L)
      J=J+1
  121 CONTINUE
C
C     PRINT TABLE OF CORRESPONDENCE
C
      IF(IPRINT.EQ.1) GO TO 2607
      PRINT 400
  400 FORMAT(1H1,T5,*TABLE OF CORRESPONDANCE*,///)
      PRINT 401
  401 FORMAT(T5,*STATE VARIABLES*,//,T5,*NS(I)*,T30,*X(NS(I))*)
      PRINT 402,(NS(I),X(NS(I)),I=1,K)
  402 FORMAT(T7,I3,T31,G12.5)
      PRINT 403
  403 FORMAT(1X,//,T5,*DECISION VARIABLES*,//,T5,*ND(J)*,T30,*X(ND(J))*)
      PRINT 404,(ND(J),X(ND(J)),J=1,N)
  404 FORMAT(T7,I3,T31,G12.5)
 2607 CONTINUE
```

```
C
      Y=0.0
      DO 1100 I=1,N
 1100 Y=Y+CA(I)*X(I)
C
C     PRINT INITIAL VALUE OF OBJECTIVE FUNCTION
C
      PRINT 1103,Y
 1103 FORMAT(1X,///,T5,*INITIAL VALUE OF OBJECTIVE FUNCTION,Y=*,T50,
     $G12.5,///)
      PRINT 9999
 9999 FORMAT(///T5,120(1H*),///)
C
C
      RETURN
      END
C
C
C
      SUBROUTINE PART AA
C
C*****************************************************************************
C     THIS SUBROUTINE MAKES THE PARTITION OF AA(I,J) INTO
C        1. THE K*K COEFFICIENT MATRIX B(I,J) OF STATE VARIABLES
C        2. THE K*N COEFFICIENT MATRIX D(I,J) OF DECISION VARIABLES
C*****************************************************************************
C
      COMMON/CONST 1/N,NF,K,KE
      COMMON/CONST 2/N1,N2,N3,N4,N5,N6
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
      COMMON/BLOCK 4/GAMS(15),GAMD(15)
C
      IJ=1
      JJ=1
      JS=1
      JD=1
      I=1
      DO 131 J=1,N2
      DO 132 IS=1,K
      IF(J.EQ.NS(IS)) GO TO 133
  132 CONTINUE
      DO 134 I=1,K
      D(I,JD)=-AA(I,ND(JJ))
  134 CONTINUE
      JJ=JJ+1
      JD=JD+1
      GO TO 131
  133 CONTINUE
      DO 135 I=1,K
      B(I,JS)=AA(I,NS(IJ))
  135 CONTINUE
      IJ=IJ+1
      JS=JS+1
  131 CONTINUE
C
C
C     PRINT PARTITIONED MATRICES B(I,J) AND C(I,J)
C
      IF(IPRINT.EQ.1) GO TO 2607
      PRINT 500
  500 FORMAT(1H1,T5,*COEFFICIENT MATRIX OF STATE VARIABLES B(I,JS)*,//)
      PRINT 503
  503 FORMAT(T7,*I*,T80,*B(I,JS)*)
      DO 501 I=1,K
  501 PRINT 502, I, (B(I,JS),JS=1,K)
  502 FORMAT(1H0,T5,I3,(/T29,8G12.5))
      PRINT 505
  505 FORMAT(1X,///,T5,*COEFFICIENT MATRIX OF DECISION VARIABLES D(I,JD)
     $*,//)
      PRINT 506
  506 FORMAT(T7,*I*,T80,*D(I,JD)*)
      DO 507 I=1,K
  507 PRINT 508, I, (D(I,JD),JD=1,N)
  508 FORMAT(1H0,T5,I3,(/T29,8G12.5))
 2607 CONTINUE
C
C
      RETURN
      END
```

```
      SUBROUTINE IDELTA(LS,LD)
C
C*****************************************************************************
C     THIS SUBROUTINE CALCULATES THE DELTA COEFFICIENTS BY GAUSS ELIMINATION
C     A NEW PARTITION IS TRIED IF THE JACOBIAN IS SINGULAR
C*****************************************************************************
C
      COMMON/BLOCK 5/BETA(15),DELTA(15,15),V(15)
      COMMON/CONST 1/N,NF,K,KE
      COMMON/CONST 9/IDET,KPRINT
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
      KPRINT=1
      IDET=1
      EPSA=0.001
      DET=0.0
      EPSDET=0.001
   31 CONTINUE
      CALL GAUSS(K,B,EPSA,D,DET,N,DELTA)
      IF(ABS(DET).LT.EPSDET) CALL NEWPAR(LS,LD)
      IF(ABS(DET).LT.EPSDET) GO TO 31
C
C     ....PRINT DELTA(I,J)....
      IF(KPRINT.EQ.1) GO TO 2607
      PRINT 32
   32 FORMAT(1X,////,T5,*DELTA COEFFICIENTS*)
      DO 33 I=1,K
   33 PRINT 34, I,(DELTA(I,J),J=1,N)
   34 FORMAT(1H0,T5,I3,(/T10,10G12.5))
 2607 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE GAUSS(N,A,EPSA,RR,DET,INDEXR,X)
C
C*****************************************************************************
C     THIS PROGRAM PERFORMS THE GAUSS ELIMINATION ON A QUADRATIC MATRIX
C
C     MULTIPLIERS ARE STORED IN THE ORIGINAL MATRIX
C     THE DETERMINANT IS COMPUTED IF IDET IS DIFFERENT FROM 1
C     ....INPUT PARAMETERS AND DATA....
C     N=ORDER OF MARTIX
C     A=A(I,J)=QUADRATIC MATRIX
C     EPSA=TOLERANCE PARAMETER
C     RR=MATRIX OF RIGHT HAND SIDES-NUMBER OF RHS=NUMBER OF COLUMNS
C     DET=DETERMINANT OF QUADRATIC MATRIX A(I,J)
C     INDEXR=NUMBER OF RIGHT HAND SIDES
C
C     IDET=CONTROL PARAMETER. IF IDET=1, THE DETERMINANT IS NOT CALCULATED. OTHE
C     RWISE IT IS
C     KPRINT=CONTROL PARAMETER. IF KPRINT=1 ONLY INPUT DATA AND SOLUTION IS PRIN
C     TED. OTHERWISE ALL INTERMEDIATE RESULTS ARE PRINTED.
C*****************************************************************************
      DIMENSION A(15,15),RR(15,15),X(15,15)
      DIMENSION AO(15,15),RO(15),M(15,15),R(15)
      DIMENSION ROL(15,15)
      COMMON/CONST 9/IDET,KPRINT
      REAL M
      K=1
      KOUNT=0
    1 CONTINUE
      K1=K+1
      IF(ABS(A(K,K)).LT.EPSA) GO TO 10
      GO TO 20
   10 CONTINUE
C
C     ....A(K,K)=0, INTERCHANGE ROWS....
      DO 11 I=K1,N
      IF(A(I,K).NE.0.0) GO TO 12
   11 CONTINUE
C
C
C     ALL A(I,K) ARE ZERO, THE MATRIX IS SINGULAR
      PRINT 9
    9 FORMAT(1H0,T5,*THE COEFFICIENT MATRIX IS SINGULAR : RETURN DET=0*)
      DET=0.0
      RETURN
   12 CONTINUE
C
C     NOT ALL A(I,K) ARE ZERO, INTERCHANGE ROWS I AND K
      KOUNT=KOUNT+1
      DO 13 J=K,N
      AO(K,J)=A(K,J)
      A(K,J)=A(I,J)
      A(I,J)=AO(K,J)
   13 CONTINUE
      DO 14 L=1,INDEXR
      ROL(K,L)=RR(K,L)
      RR(K,L)=RR(I,L)
      RR(I,L)=ROL(K,L)
   14 CONTINUE
   20 CONTINUE
C
C
C     ....A(K,K) IS NOT ZERO, PERFORM GAUSS OPERATIONS.....
C
      DO 21 I=K1,N
      M(I,K)=A(I,K)/A(K,K)
      DO 22 J=K,N
      A(I,J)=A(I,J)-M(I,K)*A(K,J)
   22 CONTINUE
C     STORE M(I,K) IN THE MARTIX A
      A(I,K)=M(I,K)
      RR(I,1)=RR(I,1)-M(I,K)*RR(K,1)
   21 CONTINUE
      K=K+1
      N1=N-1
      IF(K.LE.N1) GO TO 1
C
C     MATRIX IS NOW UPPER TRIANGULAR
C     PERFORM GAUSS OPERATIONS TO OBTAIN A DIAGONAL MATRIX
C
C
      IF(KPRINT.EQ.1) GO TO 2607
C     PRINT UPPER TRIANGULAR MATRIX...
      PRINT 50
   50 FORMAT(1H1,T5,*UPPER TRIANGULAR MATRIX AND RIGHT HAND SIDES*)
      DO 51 I=1,N
   51 PRINT 52, (A(I,J),J=1,N)
   52 FORMAT(1H0,T5,(/T5,10G12.5))
      PRINT 53
   53 FORMAT(1X,///,T5,*RIGHT HAND SIDES*)
      DO 54 I=1,N
   54 PRINT 55,(RR(I,J),J=1,INDEXR)
   55 FORMAT(1H0,T5,(/T5,10G12.5))
 2607 CONTINUE
      K=N
      IF(ABS(A(K,K)).LT.EPSA) PRINT 9
      IF(ABS(A(K,K)).LT.EPSA) DET=0.0
      IF(ABS(A(K,K)).LT.EPSA) RETURN
    5 CONTINUE
      K1=K-1
      DO 6 L=1,K1
      I=K-L
      M(I,K)=A(I,K)/A(K,K)
      DO 7 J=K,N
      A(I,J)=A(I,J)-M(I,K)*A(K,J)
    7 CONTINUE
C
C     STORE M(I,J) IN A(I,J)
      A(I,K)=M(I,K)
      RR(I,1)=RR(I,1)-M(I,K)*RR(K,1)
    6 CONTINUE
      K=K-1
      IF(K.GE.2) GO TO 5
C
C     ....CALCULATE DETERMINANT....
      IF(IDET.EQ.1) GO TO 2608
      DET=A(1,1)
      DO 30 L=2,N
      DET=DET*A(L,L)
   30 CONTINUE
      DET=DET*((-1.0)**KOUNT)
 2608 CONTINUE
      IF(IDET.EQ.1) DET=37.
      IF(INDEXR.EQ.1) RETURN
```

```
      DO 42 K=1,N1
      K1=K+1
      DO 43 I=K1,N
      RR(I,L)=RR(I,L)-A(I,K)*RR(K,L)
   43 CONTINUE
   42 CONTINUE
   40 CONTINUE
      DO 47 L=2,INDEXR
      K=N
   49 CONTINUE
      K1=K-1
      DO 48 NN=1,K1
      I=K-NN
      RR(I,L)=RR(I,L)-M(I,K)*RR(K,L)
   48 CONTINUE
      K=K-1
      IF(K.GE.2) GO TO 49
   47 CONTINUE
C
C     ....CALCULATE SOLUTIONS X(I,L)....
      DO 60 L=1,INDEXR
      DO 61 I=1,N
      X(I,L)=RR(I,L)/A(I,I)
   61 CONTINUE
   60 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE NEWPAR(LS,LD)
C**************************************************************************
C     FIRST PARTITION RESULTED IN A SINGULAR JACOBIAN
C     CHANGE PARTITION, SIMPLEXING A STATE AND A DECISUON
C     CALL PART AA. AND CONTINUE ALGORITHM WITH THE NEW PARTITION
C**************************************************************************
C
      COMMON/CONST 1/N,NF,K,KE
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
C
      IF(IPRINT.EQ.1) GO TO 2617
      PRINT 7
    7 FORMAT(1H1,T5,*THE JACOBIAN IS SINGULAR. TRY A NEW PARTITION*)
 2617 CONTINUE
C
C
C     SIMPLEX FROM HIGH TO LOW INDEX
C
      LS=LS-1
      LD=LD-1
      IF(LS.EQ.0) GO TO 10
      IF(LD.EQ.0) GO TO 10
      LSS=NS(LS)
      LDD=ND(LD)
      NS(LS)=LDD
      ND(LD)=LSS
      GO TO 11
C
   10 CONTINUE
      PRINT 80
   80 FORMAT(1X,///,T5,*THE JACOBIAN REMAINS SINGULAR FOR ANY PARTITION
     $. STOP*)
      STOP
   11 CONTINUE
      IF(IPRINT.EQ.1) GO TO 2607
      PRINT 81, LS,LD
   81 FORMAT(1X,///,T5,*LS=*,T10,I3,T15,*LD=*,T20,I3)
C     PRINT TABLE OF CORRESPONDANCE
C
      PRINT 1905
 1905 FORMAT(1H1,T5,*NEW TABLE OF CORRESPONDANCE*,///)
      PRINT 1906
 1906 FORMAT(T5,*STATE VARIABLES*,//,T8,*I*,T10,*NS(I)*,T17,*X(NS(I))*//
     1)
      PRINT 1907,(I,NS(I),X(NS(I)),I=1,K)
 1907 FORMAT(1H0,T6,I3,T11,I3,T15,G12.5)
      PRINT 1908
 1908 FORMAT(1X,///,T5,*DECISION VARIABLES*,//,T8,*J*,T10,*ND(J)*,T17,*X
     1(ND(J))*,//)
      PRINT 1909,(J,ND(J),X(ND(J)),J=1,N)
 1909 FORMAT(1H0,T6,I3,T11,I3,T15,G12.5)
 2607 CONTINUE
C
C
C
      SUBROUTINE CONDER
C
C**************************************************************************
C     THIS SUBROUTINE CALCULATES THE CONSTRAINED DERIVATIVES V(J)  OF THE OBJEC
C     TIVE FUNCTION WITH RESPECT TO THE NON-NEGATIVE DECISION VARIABLES
C     CONSTRAINED DERIVATIVES W.R.T. ZERO-SLACK-VARIABLES (EQUALITY CONSTRAINTS
C     ARE SET AT ZERO
C**************************************************************************
C
      DIMENSION HELP(15,15)
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
      COMMON/CONST 2/N1,N2,N3,N4,N5,N6
      COMMON/CONST 1/N,NF,K,KE
      COMMON/BLOCK 5/BETA(15),DELTA(15,15),V(15)
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/CONST 3/EPSY,EPSV,EPSCO,EPSD
      N5=KE+1
      DO 160 J=N5,N
      HELP(1,J)=CA(NS(1))*DELTA(1,J)
      DO 162 I=2,K
      HELP(I,J)=HELP((I-1),J)+CA(NS(I))*DELTA(I,J)
  162 CONTINUE
      I=K
      V(J)=CA(ND(J))+HELP(I,J)
      IF(ABS(V(J)).LT.EPSV) V(J)=0.0
  160 CONTINUE
C
C     PRINT CONSTRAINED DERIVATIVES V(J)
C
      IF(IPRINT.EQ.1) GO TO 2607
      PRINT 900
  900 FORMAT(1H1,T5,*CONSTRAINT DERIVATIVES V(J)*)
      PRINT 901
  901 FORMAT(1H0,//,T5,*INDEX OF DECISION VARIABLE*,T40,*CONSTRAINT
     $DERIVATIVE*,//,T17,*J*,T48,*V(J)*,///)
      PRINT 902,(J,V(J),J=1,N)
  902 FORMAT(T15,I3,T44,G12.5)
 2607 CONTINUE
C
      RETURN
      END
C
C
C
      SUBROUTINE KUNTUC(KT)
C
C**************************************************************************
C     THIS SUBROUTINE CHECKS THE KUHN-TUCKER-CONDITIONS :
C        1.  KT-CONDITIONS SATISFIED : RETURN KT=.TRUE.
C        2.  KT-CONDITIONS NOT SATISFIED : RETURN KT=.FALSE.
C
C     IF THE OPTIMUM IS REACHED,I.E. IF KT=.TRUE., THE OPTIMAL SOLUTION IS PRIN
C     TED
C**************************************************************************
C
      LOGICAL KT
      DIMENSION YSTAR(50)
      COMMON/CON 1/NSTAR
      COMMON/CONST 1/N,NF,K,KE
      COMMON/CONST 3/EPSY,EPSV,EPSCO,EPSD
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
      COMMON/CONST 7/IPRINT
      COMMON/CONST 8/KOUNT,NIMAX
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/BLOCK 5/BETA(15),DELTA(15,15),V(15)
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
C
C     ....COUNT ITERATIONS....
      KOUNT=KOUNT+1
      IF(KOUNT.GT.NIMAX) PRINT 1109
 1109 FORMAT(///T5,*MAXIMUM NUMBER OF ITERATIONS EXCEEDED,STOP*)
      IF(KOUNT.GT.NIMAX) STOP
      N5=KE+1
      DO 1000 J=N5,N
      IF(X(ND(J)).GT.EPSD.AND.ABS(V(J)).GT.EPSV) GO TO 1001
      IF(X(ND(J)).LT.EPSD.AND.V(J).LT.(-EPSV)) GO TO 1001
 1000 CONTINUE
```

```
      KT=.FALSE.
 1002 CONTINUE
C
      IF(IPRINT.EQ.1) GO TO 2607
C     PRINT KT
C
      PRINT 1003, KT
 1003 FORMAT(1H1,T5,*KUHN-TUCKER CONDITIONS SATISFIED IF KT=T, NOT SATIS
     $FIED IF KT=F*,//,T5,*KT=*,T15,L7)
 2607 CONTINUE
C
C     CALCULATE VALUE OF OBJECTIVE FUNCTION
      Y=0.0
      DO 1100 I=1,N
 1100 Y=Y+CA(I)*X(I)
C
C     ....IF THE CHANGE OF Y IN THREE ITERATIONS IS LESS THAN EPSY :
C         DEFINE KT=.TRUE. ....
C
      YSTAR(NSTAR)=Y
      IF(NSTAR.LE.3) GO TO 10
      DELTAY=ABS(YSTAR(NSTAR)-YSTAR(NSTAR-3))
      IF(DELTAY.LE.EPSY) KT=.TRUE.
   10 CONTINUE
      NSTAR=NSTAR+1
C
      IF(.NOT.KT) RETURN
C
C     PRINT OPTIMAL SOLUTION
C
      PRINT 1101
 1101 FORMAT(1X,////,T5,*OPTIMAL SOLUTION*,///,T5,*I*,T10,*X(I)*)
      PRINT 1102,(I,X(I),I=1,N)
 1102 FORMAT(1H0,T3,I3,T6,G12.5)
      PRINT 9999
 9999 FORMAT(///T5,120(1H*),///)
      PRINT 1103, Y
 1103 FORMAT(1X,///,T5,*MINIMUM VALUE OF OBJECTIVE FUNCTION,Y=*,T50,G12.
     $5)
      PRINT 9999
      PRINT 1104, KOUNT
 1104 FORMAT(///T5,*NUMBER OF ITERATIONS =*,T30,I3)
      PRINT 9999
      RETURN
      END
C
C
C
      SUBROUTINE MAXV(IVPOS,JMAX)
C
C*****************************************************************************
C     THIS SUBROUTINE FINDS THE NUMERICALLY LARGEST CONSTRAINED DERIVATIVE V(J)
C     IF V(J) IS POSITIVE, IVPOS=.TRUE.
C     IF V(J) IS NEGATIVE, IVPOS=.FALSE.
C     IF THE DECISION VARIABLE IS ZERO,AND THE CONSTRAINED DERIVATIVE IS POSITI-
C     VE, THEN THE NEXT LARGEST CONSTRAINED DERIVATIVE IS FOUND
C*****************************************************************************
C
      LOGICAL IVPOS
      COMMON/BLOCK 5/BETA(15),DELTA(15,15),V(15)
      COMMON/CONST 7/IPRINT
      COMMON/CONST 1/N,NF,K,KE
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
C
      JMAX=0
      VMAX=0.0
      NS=KE+1
      DO 1200 J=NS,N
      IF(X(ND(J)).EQ.0.0.AND.V(J).GT.0.0) GO TO 1200
      IF(ABS(V(J)).GT.ABS(VMAX)) GO TO 1201
      GO TO 1200
 1201 CONTINUE
      VMAX=V(J)
      JMAX=J
 1200 CONTINUE
C     END OF LOOP. VMAX=V(JMAX) AND JMAX DETERMINED
      IF(VMAX.GT.0.0) IVPOS=.TRUE.
      IF(VMAX.LT.0.0) IVPOS=.FALSE.
C     ORIGINAL INDEX OF VARIABLE TO BE CHANGED
      IP=ND(JMAX)
```

```
      IF(IPRINT.EQ.1) GO TO 2607
      PRINT 1210
 1210 FORMAT(1H1,T5,*NUMERICALLY LARGEST CONSTRAINT DERIVATIVE, AND VARI
     $ABLE TO BE CHANGED*,///)
      PRINT 1211, VMAX
 1211 FORMAT(T5,*NUMERICALLY LARGEST CONSTRAINT DERIVATIVE, VMAX=*,T60,
     $G12.5,//)
      PRINT 1213, IP
 1213 FORMAT(T5,*VARIABLE TO BE CHANGED:X(IP)=X(ND(JMAX)),IP=*,T66,I3,/)
      PRINT 1215, JMAX
 1215 FORMAT(1X,/,T46,*JMAX=*,T66,I3,/)
      PRINT 1216, X(IP)
 1216 FORMAT(T46,*X(IP)=*,T60,G12.5)
      PRINT 1214, IVPOS
 1214 FORMAT(1X,/T5,*IF V(JMAX) IS POSITIVE, IVPOS=.TRUE.*,/T5,
     $*IF V(JMAX) IS NEGATIVE, IVPOS=.FALSE.*,//,T5,*IVPOS=*,T66,L3)
 2607 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE CASEA1(ICASE1,JMAX,IMIN,AMIN)
C
C*****************************************************************************
C     THIS SUBROUTINE DETERMINES WHETHER :
C         1.  A DECISION VARIABLE GOES TO ZERO : CASE A1,B1 ICASE1=.TRUE.
C         2.  A STATE VARIABLE GOES TO ZERO : CASE A1,B3    ICASE1=.FALSE.
C*****************************************************************************
C
      LOGICAL ICASE1
      DIMENSION A(15)
      COMMON/CONST 1/N,NF,K,KE
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
      COMMON /BLOCK 5/BETA(15),DELTA(15,15),V(15)
C
C     BY DELTA(I,IP) IS MEANT DELTA(I,JMAX) WHERE IP=ND(JMAX)
C     IF DELTA(I,IP)IS NEGATIVE FOR ALL I : ICASE1=.TRUE.
C
      IF(IPRINT.EQ.1) GO TO 2607
      PRINT 1729
 1729 FORMAT(1H1,T5,*WE ARE IN CASE A1:THE VARIABLE X(IP)=X(ND(JMAX)) MU
     1ST DECREASE*, ///,T5,*QUESTION:WILL A STATE VARIABLE GO TO ZERO BE
     2FORE X(IP)*,///,T5,*IF YES,ICASE1=.FALSE.,AND WE ARE IN CASE A1,B3
     3*,//,T5,*IF NO,ICASE1=.TRUE.,AND WE ARE IN CASE A1,B1*)
      PRINT 19, IMIN,JMAX
   19 FORMAT(1X,///,T5,*IMIN=*,T15,I3,T25,*JMAX=*,T35,I3)
 2607 CONTINUE
      DO 1300 I=1,K
      IF(DELTA(I,JMAX).GT.0.0) GO TO 1301
 1300 CONTINUE
      ICASE1=.TRUE.
C
      IF(IPRINT.EQ.1) GO TO 2608
      PRINT 1299, ICASE1
 1299 FORMAT(1H1,T5,*ALL DELTA(I,IP) NEGATIVE OR ZERO*,//,T5,*ICASE1=*,
     $T15,L3)
      PRINT 1733
 1733 FORMAT(1X,/////,T5,*A DECISION VARIABLE GOES TO ZERO : ICASE1=.TRU
     $E*,/T5,*A STATE VARIABLE GOES TO ZERO : ICASE1=.FALSE.*)
 2608 CONTINUE
      RETURN
 1301 CONTINUE
C
C     DETERMINE WHETHER ANY NON-NEG STATE VARIABLE GOES TO ZERO :
C     FIND AMIN=MIN(X(NS(I))/DELTA(NS(I),IP), ALL NS(I)
C
      IMIN=1
      AMIN=10000.
      DO 1302 I=1,K
      IF(ITYPE(NS(I)).EQ.0) GO TO 1305
      IF(DELTA(I,JMAX).LE.0.0) GO TO 1305
      A(I)=X(NS(I))/DELTA(I,JMAX)
      IF(A(I).LT.AMIN) GO TO 1303
      GO TO 1305
 1303 CONTINUE
      AMIN=A(I)
      IMIN=I
 1305 CONTINUE
 1302 CONTINUE
C     END OF LOOP : AMIN=A(IMIN) AND IMIN DEYERMINED
```

```
      IF(X(ND(JMAX)).GT.AMIN) GO TO 1304
      ICASE1=.TRUE.
C
      IF(IPRINT.EQ.1) GO TO 2609
      PRINT 1298, ICASE1
 1298 FORMAT(1H1,T5,*NO NON-NEG STATE VARIABLE DRIVEN TO ZERO*,//,T5,
     $*ICASE1=*,T15,L3)
      PRINT 1700
 1700 FORMAT(1X,/////,T5,*A DECISION VARIABLE GOES TO ZERO : ICASE1=.TRU
     $E*,/T5,*A STATE VARIABLE GOES TO ZERO : ICASE1=.FALSE.*)
 2609 CONTINUE
      RETURN
 1304 CONTINUE
C
C     ORIGINAL INDEX OF STATE VARIABLE DRIVEN TO ZERO
C
      IR=NS(IMIN)
      ICASE1=.FALSE.
C
      IF(IPRINT.EQ.1) GO TO 2610
      PRINT 1297, ICASE1
 1297 FORMAT(1H1,T5,*A NON-NEG STATE VARIABLE IS DRIVEN TO ZERO*,//,
     $T5,*ICASE1=*,T15,L3,///)
      PRINT 1296
 1296 FORMAT(T5,*THE STATE VARIABLE X(IR)=X(NS(IMIN)) GOES TO ZERO*,//)
      PRINT 1295, IR, IMIN
 1295 FORMAT(T5,*IR=*,T15,I3,/,T5,*IMIN=*,T15,I3)
      PRINT 1294, X(IR),AMIN
 1294 FORMAT(T5,*X(IR)=*,T15,G12.5,/,T5,*AMIN=*,T15,G12.5)
      PRINT 1701
 1701 FORMAT(1X,/////,T5,*A DECISION VARIABLE GOES TO ZERO : ICASE1=.TRU
     $E*,/T5,*A STATE VARIABLE GOES TO ZERO : ICASE1=.FALSE.*)
 2610 CONTINUE
C
      RETURN
      END
C
C
C
      SUBROUTINE CASEA2(JMAX,KT)
C
C*****************************************************************************
C     THIS SUBROUTINE HANDLES CASE A2, I.E.
C        1. X(IP) IS INCREASED UNTIL THE STATE VARIABLE X(IR) IS ZERO
C        2. X(IP) AND X(IR) ARE SIMPLEXED AND A NEW TABLE OF CORRESPONDEN
C           CE OBTAINED
C        3. NEW DELTAS AND CONSTRAINED DERIVATIVES ARE COMPUTED
C*****************************************************************************
C
      DIMENSION AB(15)
      LOGICAL KT
      COMMON/CONST 1/N,NF,K,KE
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/KONST 1/IFREQ
      COMMON/CONST 8/KOUNT,NIMAX
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
      COMMON/BLOCK 5/BETA(15),DELTA(15,15),V(15)
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
C
C     FOR NEGATIVE DELTA(I,JMAX), AND NON-NEG STATE VARIABLES, THE MINIMUM OF
C     ABS(X(NS(I))/DELTA(I,JMAX)) IS DETERMINED
C
      IIMIN=1
      AAMIN=10000.
      DO 1900 I=1,K
      IF(ITYPE(NS(I)).EQ.0) GO TO 1901
      IF(DELTA(I,JMAX).GE.0.0) GO TO 1901
      AB(I)=X(NS(I))/DELTA(I,JMAX)
      IF(ABS(AB(I)).LT.AAMIN) GO TO 1902
      GO TO 1901
 1902 CONTINUE
      AAMIN=ABS(AB(I))
      IIMIN=I
 1901 CONTINUE
 1900 CONTINUE
C
C     END OF LOOP, AAMIN=ABS(AA(IIMIN)) AND IIMIN DETERMINED
```

```
      PRINT 1903
 1903 FORMAT(1H1,T5,*WE ARE IN CASE A2  THE VARIABLE X(IP)=X(ND(JMAX)) M
     1UST INCREASE*,///,T5,*A STATE VARIABLE X(IR)=X(NS(IIMIN)) WILL BE
     2DRIVEN TO ZERO*)
      PRINT 19, IIMIN,JMAX
   19 FORMAT(1X,///,T5,*IMIN=*,T15,I3,T25,*JMAX=*,T35,I3)
      IF(AAMIN.LT.10001.0.AND.AAMIN.GT.9999.0) PRINT 1899
 1899 FORMAT(1X,////,T5,*IN THIS CASE THE DECISION X(ND(JMAX)) CAN INCRE
     $ASE INDEFINITELY*,//,T5,*THE PROBLEM IS POORLY POSED -- STOP *)
 2607 CONTINUE
      IF(AAMIN.LT.10001.0.AND.AAMIN.GT.9999.0)STOP
C
C     CHANGE PARTITION   SIMPLEX THE DECISION X(IP) AND THE STATE X(IR)
C
      IP=ND(JMAX)
      IR=NS(IIMIN)
      NS(IIMIN)=IP
      ND(JMAX)=IR
C
C     CALCULATE NEW STATE VARIABLES
C
      XOIP=X(IP)
      DO 1904 I=1,K
      X(NS(I))=X(NS(I))+DELTA(I,JMAX)*AAMIN
 1904 CONTINUE
      X(IP)=XOIP+AAMIN
C
C     DEFINE NEW DECISION VARIABLES
C
      X(IR)=0.0
C
      CALL PRINT(KOUNT,IFREQ)
C
C
C     CALL SUBROUTINE NEWVAL FOR CALCULATION OF NEW DELTAS AND CONSTRAINT DERI
C      TIVES
C
      CALL NEWVAL(JMAX,IIMIN)
      CALL KUNTUC(KT)
      RETURN
      END
C
C
C
      SUBROUTINE CASEB1(JMAX,KT)
C
C*****************************************************************************
C     THIS SUBROUTINE HANDLES CASE A1,B1, I.E.
C        1. SAME PARTITION AS PREVIOUSLY
C        2. ONE DECISION VARIABLE GOES TO ZERO
C*****************************************************************************
C
      LOGICAL KT
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
      COMMON/BLOCK 5/BETA(15),DELTA(15,15),V(15)
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
      COMMON/KONST 1/IFREQ
      COMMON/CONST 8/KOUNT,NIMAX
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/CONST 1/N,NF,K,KE
      IF(IPRINT.EQ.1) GO TO 2617
      PRINT 1923
 1923 FORMAT(1H1,T5,*WE ARE IN CASE A1,B1. THE DECISION X(ND(JMAX)) GOES
     $ ZERO*)
      PRINT 1924, JMAX
 1924 FORMAT(1X,///,T5,*JMAX=*,T10,I3)
 2617 CONTINUE
C
C     CALCULATE NEW STATE VARIABLES
C
      IP=ND(JMAX)
      XOIP=X(IP)
      DO 10 I=1,K
      X(NS(I))=X(NS(I))-DELTA(I,JMAX)*XOIP
   10 CONTINUE
C
C     DEFINE NEW DECISION VARIABLES
C
      X(IP)=0.0
C
      CALL PRINT(KOUNT,IFREQ)
```

```
C     SINCE THE PARTITION IS UNCHANGED, SO ARE THE CONSTRAINED DERIVATIVES
C
      CALL KUNTUC(KT)
      RETURN
      END
C**************************************************************************
C
C
C
      SUBROUTINE CASEB3(KT,JMAX,IMIN,AMIN)
C
C**************************************************************************
C     THIS SUBROUTINE HANDLES CASE A1,B3, I.E.
C       1. X(IP) IS DECREASED UNTIL THE STATE X(IR) IS ZERO
C       2. X(IP) AND X(IR) ARE SIMPLEXED, AND A NEW TABLE OF COORRESPONDENCE IS
C          OBTAINED
C       3. NEW DELTAS AND CONSTRAINED DERIVATIVES ARE COMPUTED
C**************************************************************************
C
      LOGICAL KT
      COMMON/CONST 1/N,NF,K,KE
      COMMON/CONST 8/KOUNT,NIMAX
      COMMON/KONST 1/IFREQ
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
      COMMON/BLOCK 5/BETA(15),DELTA(15,15),V(15)
      COMMON/CONST 7/IPRINT
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
C
      PRINT 1499
 1499 FORMAT(1H1,T5,*WE ARE IN CASE A1,B3, THE DECISION X(IP) WILL DECRE
     1ASE UNTIL THE STATE X(IR) IS DRIVEN TO ZERO*)
      PRINT 19, IMIN,JMAX
   19 FORMAT(1X,///,T5,*IMIN=*,T15,I3,T25,*JMAX=*,T35,I3)
C
C     CHANGE PARTITION,SIMPLEXING THE DECISION X(IP) AND THE STATE X(IR)
C
      IP=ND(JMAX)
      IR=NS(IMIN)
      NS(IMIN)=IP
      ND(JMAX)=IR
C
C     CALCULATE NEW STATE VARIABLES
C
      XOIP=X(IP)
      DO 1500 I=1,K
      X(NS(I))=X(NS(I))-DELTA(I,JMAX)*AMIN
 1500 CONTINUE
      X(IP)=XOIP-AMIN
C
C     DEFINE NEW DECISION VARIABLES
C
      X(IR)=0.0
C
      CALL PRINT(KOUNT,IFREQ)
C
C     CALL NEWVAL FOR CALCULATION OF NEW DELTAS AND CONSTRAINED DERIVATIVES
C
      CALL NEWVAL(JMAX,IMIN)
      CALL KUNTUC(KT)
      RETURN
      END
C
C
C
      SUBROUTINE NEWVAL(JMAX,IMIN)
C
C**************************************************************************
C     THIS SUBROUTINE CALCULATES NEW DELTAS AND CONSTRAINED DERIVATIVES AFTER
C     CHANGE OF PARTITION
C**************************************************************************
C
      DIMENSION Z(15),ZZ(15)
      COMMON/CONST 1/N,NF,K,KE
      COMMON/CONST 7/IPRINT
      COMMON/CONST 2/N1,N2,N3,N4,N5,N6
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
      COMMON/BLOCK 5/BETA(15),DELTA(15,15),V(15)
      N4=KE+1
      DO 2000 I=1,K
      Z(I)=DELTA(I,JMAX)
 2000 CONTINUE
      DO 1999 J=N4,N
      ZZ(J)=DELTA(IMIN,J)
 1999 CONTINUE
      DELTRP=DELTA(IMIN,JMAX)
      DO 2001 I=1,K
      DO 2002 J=N4,N
      DELTA(I,J)=DELTA(I,J)-Z(I)*ZZ(J)/DELTRP
 2002 CONTINUE
 2001 CONTINUE
      DO 2009 J=N4,N
      DELTA(IMIN,J)=-ZZ(J)/DELTRP
 2009 CONTINUE
      DO 2003 I=1,K
      DELTA(I,JMAX)=Z(I)/DELTRP
 2003 CONTINUE
      DELTA(IMIN,JMAX)=1.0/DELTRP
C
C     PRINT NEW DELTA COEFFICIENTS
      IF(IPRINT.EQ.1) GO TO 2607
C
      PRINT 2004
 2004 FORMAT(1H1,T5,*NEW COEFFICIENTS DELTA(I,J)*,////)
      PRINT 2005
 2005 FORMAT(T5,*INDEX*,T75,*COEFFICIENTS*,//,T7,*I*,T76,*DELTA(I,J)*,//
     1)
      DO 2006 I=1,K
      PRINT 2007, I,(DELTA(I,J),J=N4,N)
 2007 FORMAT(1H0,T5,I3,(/T40,7G12.5))
 2006 CONTINUE
 2607 CONTINUE
C
C     NEW CONSTRAINED DERIVATIVES
C
      VP=V(JMAX)/DELTRP
      DO 2010 J=N4,N
      V(J)=V(J)-VP*ZZ(J)
 2010 CONTINUE
      V(JMAX)=VP
C
C     PRINT NEW CONSTRAINED DERIVATIVES
C
      IF(IPRINT.EQ.1) GO TO 2608
      PRINT 2011
 2011 FORMAT(1H1,T5,*NEW CONSTRAINT DERIVATIVES V(J)*)
      PRINT 2012
 2012 FORMAT(1H0,//,T5,*INDEX OF DECISION VARIABLE*,T40,*CONSTRAINT DERI
     1VATIVE*,//,T17,*J*,T48,*V(J)*,///)
      PRINT 2013,(J,V(J),J=1,N)
 2013 FORMAT(T15,I3,T44,G12.5)
 2608 CONTINUE
C
      RETURN
      END
C
C
C
      SUBROUTINE PRINT(KOUNT,IFREQ)
C
C**************************************************************************
C     THIS SUBROUTINE PRINTS TABLES OF CORRESPONDENCE AND VALUES OF THE OBJECTI
C     VE FUNCTION
C     IF IPRINT=0 ALL SORTS OF DEBUGGING PRINTOUTS ARE PROVIDED
C     IF IPRINT=1 ONLY INPUT,TABLES OF CORRESPONDENCE, AND SOLUTION WILL BE PRI
C     TED. FREQUENCY OF PRINTOUTS ARE DETERMINED BY IFREQ.
C           IFREQ=0   ONLY INPUT AND SOLUTION PRINTED
C           IFREQ=1   TABLE OF CORR PRINTED AT EACH LEVEL
C           IFREQ=5   TABLE OF CORR PRINTED AR EACH 5 LEVEL
C           IFREQ=10  TABLE OF CORR PRINTED AT EACH 10 LEVEL
C**************************************************************************
C
      COMMON/CONST 1/N,NF,K,KE
      COMMON/BLOCK 2/X(25),ITYPE(15)
      COMMON/BLOCK 3/NS(15),ND(15),NN(25)
      COMMON/BLOCK 1/CA(15),AA(25,25),R(15),B(15,15),D(15,15)
      IF(IFREQ.EQ.0) GO TO 100
      IF(IFREQ.EQ.1) GO TO 101
      IFIVE=0
      ITEN=0
```

```
      FIVEI=FLOAT(KOUNT/5)
      TENR=FLOAT(KOUNT)/10.0
      TENI=FLOAT(KOUNT/10)
      IF(FIVER.GT.FIVEI*0.999.AND.FIVER.LT.FIVEI*1.001) IFIVE=1
      IF(TENR.GT.TENI*0.999.AND.TENR.LT.TENI*1.001) ITEN=1
      IF(IFREQ.EQ.5.AND.IFIVE.EQ.0) GO TO 100
      IF(IFREQ.EQ.10.AND.ITEN.EQ.0) GO TO 100
  101 CONTINUE
C     PRINT TABLE OF CORRESPONDENCE
C
      PRINT 1905
 1905 FORMAT(1H1,T5,*NEW TABLE OF CORRESPONDENCE*,///)
      PRINT 1906
 1906 FORMAT(T5,*STATE VARIABLES*,//,T8,*I*,T10,*NS(I)*,T17,*X(NS(I))*//
     1)
      PRINT 1907,(I,NS(I),X(NS(I)),I=1,K)
 1907 FORMAT(1H0,T6,I3,T11,I3,T15,G12.5)
      PRINT 1908
 1908 FORMAT(1X,///,T5,*DECISION VARIABLES*,//,T8,*J*,T10,*ND(J)*,T17,*X
     1(ND(J))*,//)
      PRINT 1909,(J,ND(J),X(ND(J)),J=1,N)
 1909 FORMAT(1H0,T6,I3,T11,I3,T15,G12.5)
C
C     CALCULATE AND PRINT NEW VALUE OF OBJECTIVE FUNCTION
C
      Y=0.0
      DO 1910 I=1,N
      Y=Y+CA(I)*X(I)
 1910 CONTINUE
      PRINT 1911, Y
 1911 FORMAT(1X,///,T5,*NEW VALUE OF OBJECTIVE FUNCTION,Y=*,T50,G12.5)
  100 CONTINUE
      RETURN
      END
```