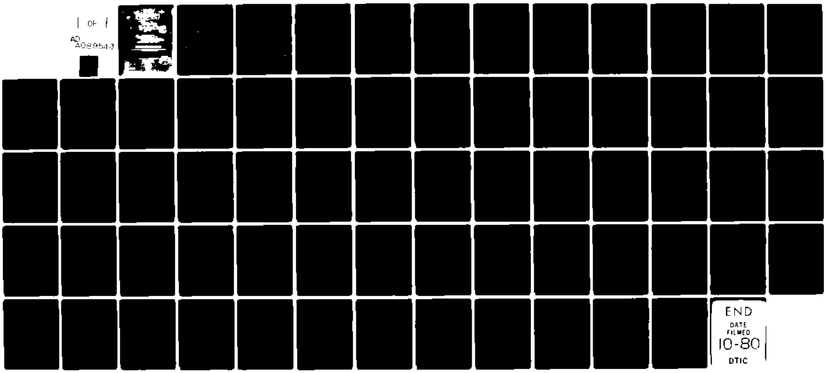
AD-A089 543 STANFORD UNIV CA DEPT OF OPERATIONS RESEARCH F/6 12/2
A COMPUTER PROGRAM FOR THE STAIRCASE INTEGER PROGRAMMING PROBLE--ETC(U)
JUL 80 L J POLLENZ N00014-76-C-0418
UNCLASSIFIED TR-95 NL



A COMPUTER PROGRAM FOR THE STAIRCASE INTEGER PROGRAMMING PROBLEM

BY

LYNNE J. POLLENZ

TECHNICAL REPORT NO. 95

JULY 1980

PREPARED UNDER CONTRACT

N00014-76-C-0418 (NR-047-061)

FOR THE OFFICE OF NAVAL RESEARCH

Frederick S. Hillier, Project Director

DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

Also issued as Technical Report No. 80-16, Dept. of Operations Research
Stanford University, under National Science Foundation Grant MCS76-81259

A Computer Program for the Staircase Integer Programming Problem

1. Introduction

The computer code SDA to be described in this report solves the staircase integer linear programming problem (SP) given by:

$$
\text{(SP)}\qquad
\begin{aligned}
&\text{maximize} && \sum_{t=1}^{T} c_t x_t \\
&\text{subject to:} && A_1 x_1 \le b_1 \\
& && A_t x_t + B_{t-1} x_{t-1} \le b_t, \quad t=2,3,\ldots,T \\
& && L_t \le x_t \le U_t, \quad t=1,2,\ldots,T \\
& && x_t \text{ integer}, \quad t=1,2,\ldots,T.
\end{aligned}
$$

This formulation arises in multiplant production allocation problems, multisector economic planning models, and multitime period production and inventory problems.

The solution method used by the program SDA relies upon decomposition of the problem (SP) into smaller subproblems (S_t), each of which can then be efficiently solved by an LP-based branch-and-bound routine. Each subproblem (S_t) is specified by a vector of costs c_t, the "diagonal" submatrix A_t, the "offdiagonal" submatrix B_{t-1}, the right-hand side b_t, and upper and lower bounds U_t and L_t on the variables x_t. The actual formulation is as follows:

$$
(S_t)\qquad
\begin{aligned}
&\text{maximize} && c_t x_t \\
&\text{subject to:} && A_t x_t \le b_t - B_{t-1} x_{t-1} \\
& && L_t \le x_t \le U_t \\
& && x_t \text{ integer.}
\end{aligned}
$$

The algorithm proceeds by finding a solution to a subproblem (say S_t) and moving forward to the next subproblem (S_{t+1}) with a new right-hand side determined by the solution to S_t. When a set of solutions to all the subproblems is found, this solution will be feasible for the original problem. Prices are calculated to help guide the search for subproblem solutions toward an optimal value. Bounds on the maximum objective value obtainable given the current solutions to a subset of the subproblems are utilized to speed fathoming. See Pollenz [1980] for a detailed discussion of the staircase decomposition algorithm.

The branch-and-bound search procedure, which is used to solve the subproblems obtained via decomposition of (SP), is a slightly modified version of the computer program BB written by Gary Kochman as part of his dissertation at Stanford University. This computer code solves pure integer programming problems with general upper and lower bounds on the variables using a branch-and-bound technique similar to that presented by Dakin [1965]. Tomlin's [1971] improved penalties are employed to guide the choice of branching variable at unfathomed nodes, with the branch being taken in the direction opposite to the maximum penalty. Nodes are removed from the branch-and-bound list according to a last-in-first-out (LIFO) strategy, and reoptimization after a branch is accomplished by the dual simplex method. See Kochman [1976] for further details about this program.

The linear programming portions of the code BB were developed by John Tomlin of the Systems Optimization Laboratory at Stanford University and adapted by Kochman to efficiently deal with simple upper and lower bounds on the variables. The most important features of LPM-1 are: storage of the basis inverse in product form (see Orchard-Hays [1968]); LU decomposition of the basis inverse (see

Benichou et al. [1977]); and storage of all data, including the inverse, in compressed form (i.e., zeroes are not saved).

In order to maintain compatibility with BB, the computer code SDA was written in FORTRAN. Testing and evaluation of this program was carried out on the SCORE DEC System 20 computer at Stanford University. Due to the restrictive nature of the FORTRAN-10 compiler at SCORE, few of the features of SDA would be unacceptable to another system. The sole exception is the usage of the system clock routine IHPTIM, which is employed to record total CPU time and the percentage of execution time devoted to various portions of the algorithm. Such information is valuable, but if no system clock is available, the appropriate sections of the program can be eliminated without affecting the whole.

The main program and input requirements for SDA are discussed in the next section. The current restrictions on problem size are given in section 3. The various subroutines which comprise the bulk of the program SDA are outlined in section 4. The output generated by SDA is described in section 5, and sample input and output are given at the end of the appendix, following the listing of the program.

2. Main Program and Input Requirements

The main program calls the input subroutine, a subroutine which generates a few initial bounds, and then iteratively calls the branch-and-bound program to solve the appropriate subproblems. After the optimal solution has been discovered (and optimality verified), control passes to the next section of the main program, which computes some timing information and calls the output subroutine.

The first input card must contain values for the parameters

IFPROB, NP, IOBJ, INVFRQ, ITRFRQ, and INITBD, in (4I4, I5, I10) format. These variables have the following significance:

IFPROB = problem identification number (must be nonzero).

NP = number of time periods (equivalently, number of subproblems).

IOBJ = row number of objective row. Default is 1. (Currently, IOBJ must be 1 for subroutine BOUNDR to operate correctly; this can be corrected.)

INVFRQ = frequency with which basis inversion is carried out in the linear programming portions of the code. If INVFRQ = k, basis reinversion will occur after every k simplex iterations. INVFRQ should not be set greater than the number of rows in the smallest subproblem.

ITRFRQ = upper limit on total number of simplex iterations. Computations are terminated if the total number of simplex iterations exceeds ITRFRQ. If ITRFRQ = 0, the default value of 999999 is used.

INITBD = initial lower bound estimate for maximum objective value. It is used in fathoming tests until the first feasible solution with objective value greater than INITBD is found.

Following the initial input card, data for each subproblem is read in sequentially. The data must be input in the following order:

1. A leading card with 'SUB nnnn' in columns 5-12, where nnnn

is replaced by the appropriate subproblem number. Subproblem numbers must be sequential, beginning with 1.

2. Nonzero entries of the offdiagonal submatrix B_{t-1} (except for subproblem 1, for which no offdiagonal matrix is needed).

3. Row name and type for each constraint.

4. Nonzero entries of the diagonal matrix A_t.

5. The right-hand side b_t.

6. Lower bounds L_t, followed by upper bounds U_t, in format 15F5.0.

The type of constraint is denoted by a single letter preceding the row name. 'E' stands for an equality, 'L' and 'G' identify less than or equal to and greater than or equal to inequalities respectively, and 'N' marks the objective row. The format for this row identification input is (1X, A1, 2X, A8).

For the reading of all coefficients, the following information must be specified: column name, row name, and entry value. The format used for this is (4X, A8, 2X, A8, 2X, F12.4, 3X, A8, 2X, F12.4). The pattern for row name and value is repeated so that two entries for the same column may be input on one line. The row names must match a name read in earlier (input step #3). Zero coefficients may be omitted completely. This format is the same as used for LPM-1, and is consistent with standard MPS format.

The only restriction on input data is that the cost coefficients are assumed to be integral. Furthermore, these coefficients must be input with the opposite sign (i.e., if you wish to maximize cx, you must input -c for the objective row). This is an unfortunate result of the fact that LPM-1 is a minimization routine.

After all data has been read in, an END card signifies the end of the data file.

The input for a sample run of SDA is included in the appendix, following the program listing.

3. Restrictions on the Use of SDA

For current dimensioning, the following restrictions apply to use of the SDA:

1. The number of subproblems (NP) must be $\leq$ 10.
2. The total number of rows, excluding the objective row, must not exceed 60.
3. The total number of columns (including slacks) must be $\leq$ 120.
4. The total number of nonzero elements in the constraint matrix (excluding the elements of A_1) must not exceed 1000. (This restriction is necessary only for subroutine BOUNDR.)
5. The total number of nonzero elements in the first diagonal submatrix, A_1, must not exceed 1000.
6. No right-hand side $b_t(i)$ may exceed 1000 (due to default upper bounds on the slack variables).
7. The number of nodes in the branch-and-bound tree must never exceed 500. (For [0,1] problems, the number of nodes will never exceed the total number of columns, so this restriction will not be a factor.)

4. Subroutines of SDA

BLOCK DATA (from LPM-1): sets initial values for many global program constants, including maximum problem dimensions and minimum tolerances.

SUBROUTINE INPUT (IFPROB,INITBD) (from LPM-1): reads in all problem data, checks that problem dimensions do not exceed current specifications.

Parameters:

IFPROB - nonzero problem identification number (output)

INITBD - initial lower bound estimate of maximal objective value (output parameter)

SUBROUTINE INPSTO: stores all data relevant to subproblem whose number is stored in variable NS. This subroutine is called after reading the data initially (from INPUT) and before each forward step of the algorithm.

Entry points:

STORE - entry point from subroutine TESTX. After an integral solution is found to subproblem NS, the current state of the corresponding search is saved by a call to STORE.

SUBROUTINE RESTOR (MNFLAG): restores all data from subproblem NS in preparation for either a forward or a backtracking step. For a forward step, the LP relaxation of this subproblem, with new right-hand side determined by a call to FIXRHS, is solved.

Parameters:

MNFLAG - input parameter set at 0 if backtracking, 1 if taking a forward step.

SUBROUTINE FIXRHS: computes new right-hand side for subproblem NS, given the (newly established) setting of variables of the previous subproblem.

SUBROUTINE BOUNDR: calculates two LP-based bounds on the maximum objective values for some aggregations of the subproblems. These bounds are used for fathoming in subroutines TESTX, BKTRAK, and PENLTS. Also, a vector of prices for use in guiding the branch-and-bound search (see subroutine PENLTS) is computed.

SUBROUTINE UPDATX: updates right-hand side by taking into consideration variables which are nonbasic at their upper or lower (nonzero) bounds.

SUBROUTINE FTRAN (IPAR) (from LPM-1): performs forward transformation of matrix column (stored in vector Y) by basis inverse.

Parameters:

IPAR - input parameter indicating which eta-vectors are used to update the matrix column.

SUBROUTINE BTRAN (from LPM-1): performs backward transformation on column stored in vector Y.

SUBROUTINE FORMC (from LPM-1): forms objective function vector for Phase I of primal simplex method.

SUBROUTINE PRICE (from LPM-1): prices out nonbasic columns for primal simplex method and chooses pivot column (stored in variable JCOLP). Also checks for dual feasibility.

SUBROUTINE CHUZR (from LPM-1): chooses pivot row for primal simplex method using min ratio test; stores pivot row in variable IROWP.

SUBROUTINE WRETA (from LPM-1): forms a new eta-vector for the product form of the inverse.

SUBROUTINE SHIFTR (IOLD, INEW) (from LPM-1): rearranges data storage.

Parameters:

IOLD, INEW - input parameters indexing storage locations. Move from location designated by IOLD to that designated by INEW.

SUBROUTINE INVERT (from LPM-1): Creates basis inverse by LU decomposition.

SUBROUTINE UNPACK (IV) (from LPM-1): expands compressed matrix columns by inserting zeroes appropriately.

Parameters:

IV - input parameter indexing the matrix column to be expanded.

SUBROUTINE SHFTE (from LPM-1): Subroutine for INVERT.

SUBROUTINE UPBETA (from LPM-1): updates right-hand sides following a primal or dual simplex pivot.

SUBROUTINE NORMAL (ITSINV) (from LPM-1): directs execution of primal simplex method.

Parameters:

ITSINV - counts number of simplex iterations since last basis inversion (for comparison with INVFRQ).

SUBROUTINE BANDB (INITBD) (from BB): master program for branch-and-bound integer programming routine. It also serves as master program for reoptimization via the revised dual simplex method after a forward branch.

Parameters:

INITBD - initial lower bound estimate on maximum objective value.

Entry points:

BBENTR - reentry point from main program to apply branch-and-bound search to any subproblem after the first.

SUBROUTINE DCHUZR (from BB): Selects pivot row for dual simplex method and stores it in variable IROWP. Also checks for dual optimality.

SUBROUTINE DCHUZC (from BB): selects pivot column for dual simplex method and stores it in variable JCOLP. Also checks for dual feasibility.

SUBROUTINE TESTX (from BB): tests LP-optimal solution at current node for integrality and for fathoming (fathoming tests and branching strategy modified substantially from those of BB). Any new complete solution found is immediately printed out and saved in the array INCUMB; any integral solution to a subproblem (other than the last) causes a call to entry STORE in preparation for a forward step. Variable MSTAT flags the result of testing and is checked in subroutine BANDB.

SUBROUTINE PENLTS (from BB): computes Tomlin's improved up- and down-penalties and the Gomory penalty at each node. Checks for forced branches on both basic and non-basic variables. If fathoming does not occur, a branch variable is chosen in accordance with the combination of Tomlin's penalties and the prices computed in BOUNDR, and subroutine BRANCH is called. (Substantial modifications have been made from the version of this subroutine given in BB.)

SUBROUTINE BRANCH (from BB): Performs necessary bookkeeping for branching on variable indexed by ICOL. Increments list of stored nodes, revises bounds on branching variable, and saves opposite branch direction and a bound on the maximum objective value on that opposite branch.

SUBROUTINE BKTRAK (from BB): backtracks to a promising (unfathomed) node from the list of stored nodes. Employs last-in-first-out selection rule. If backtracking brings us back to the previous subproblem, the appropriate data and status of the search of that subproblem are restored by a call to subroutine RESTOR.

SUBROUTINE WRAPUP (from BB): Outputs final solution information. (See output from sample run at the end of the appendix.)

For further information on subroutines derived from LPM-1 see Tomlin [1975].

5. Description of Output

The output produced by this program falls onto 3 sections. The initial output contains the problem identification number and dimensions, followed by the prices and bounds computed by subroutine BOUNDR. At this point the iterative portion of the algorithm is begun. Each time an improved feasible solution is discovered, the time of discovery, total number of branches taken, and new maximum objective value (called INCVAL) are printed out by subroutine TESTX. After termination, some information regarding the total time taken and time spent at certain tasks is output. If termination occurs normally, the final section of output contains the optimum solution and its objective value. The solution is printed in 20I5 format in the following order: objective value, slack variables, and integer variables for subproblem 1, then subproblem 2, ..., and finally for subproblem NP. If computation is halted because the limit on iterations (ITRFRQ) has been exceeded, then the best solution found so far and the total number of simplex iterations used are printed.

REFERENCES

Benichou, M., J. M. Gauthier, G. Hentges, and G. Ribiere, "The Efficient Solution of Large-Scale Linear Programming Problems - Some Algorithmic Techniques and Computational Results." Mathematical Programming 13 (1977), 280-322.

Dakin, R. J., "A Tree Search Algorithm for Mixed Integer Programming Problems." Computer Journal 8, (1965), 250-255.

Gary A. Kochman, "Computer Programs for Decomposition in Integer Programming." Technical Report 71, Department of Operations Research, Stanford University, (1976).

Orchard-Hays, William, Advanced Linear Programming Computing Techniques, McGraw-Hill, New York (1968).

Pollenz, Lynne J., "The Staircase and Related Structures in Integer Programming." Technical Report 94, Department of Operations Research, Stanford University, (1980).

Tomlin, John A., "An Improved Branch-and-Bound Method for Integer Programming." Operations Research 19 (1971), 1070-1075.

Tomlin, John A., "Users Guide for LPM-1." Systems Optimization Laboratory, Department of Operations Research, Stanford University (1975).

Appendix:

Program Listing, Sample Input and Output.

****** Sample Input ******

```
   1    4    1    5 9999          0
     SUB      1
ROWS
 N   OBJ
 L   ROW   101
 L   ROW   102
 L   ROW   103
 L   ROW   104
 L   ROW   105
COLUMNS
     COL   101   OBJ                   -10.00    ROW   101            -10.00
     COL   101   ROW   102              -6.00    ROW   103             18.00
     COL   101   ROW   104               7.00    ROW   105              9.00
     COL   102   OBJ                   -19.00    ROW   101             -2.00
     COL   102   ROW   102              -9.00    ROW   103             15.00
     COL   102   ROW   104              15.00    ROW   105              0.00
     COL   103   OBJ                    -7.00    ROW   101              4.00
     COL   103   ROW   102               9.00    ROW   103              0.00
     COL   103   ROW   104              -9.00    ROW   105             -8.00
     COL   104   OBJ                   -12.00    ROW   101              9.00
     COL   104   ROW   102              15.00    ROW   103            -10.00
     COL   104   ROW   104               4.00    ROW   105             -8.00
     COL   105   OBJ                    -6.00    ROW   101              5.00
     COL   105   ROW   102              15.00    ROW   103             12.00
     COL   105   ROW   104               0.00    ROW   105              0.00
RHS
     RHS1        ROW   101              39.00    ROW   102             36.00
     RHS1        ROW   103              25.00    ROW   104             20.00
     RHS1        ROW   105              35.00
BOUNDS
   0.    0.    0.    0.    0.
   1.    1.    1.    1.    1.
     SUB      2
ROWS
 N   OBJ
 L   ROW   201
 L   ROW   202
 L   ROW   203
 L   ROW   204
 L   ROW   205
OFFDIAGONAL COLUMNS
     COL   101   ROW   201               1.00    ROW   202             12.00
     COL   101   ROW   203              17.00    ROW   204              0.00
     COL   101   ROW   205               3.00
     COL   102   ROW   201              -2.00    ROW   202              0.00
     COL   102   ROW   203              16.00    ROW   204             17.00
     COL   102   ROW   205              13.00
     COL   103   ROW   201              -8.00    ROW   202             12.00
     COL   103   ROW   203               0.00    ROW   204             -1.00
     COL   103   ROW   205               5.00
     COL   104   ROW   201              17.00    ROW   202              7.00
     COL   104   ROW   203              -2.00    ROW   204             -1.00
     COL   104   ROW   205               3.00
     COL   105   ROW   201              -4.00    ROW   202              7.00
     COL   105   ROW   203               0.00    ROW   204              0.00
     COL   105   ROW   205              -8.00
COLUMNS
```

```
        COL   201  OBJ                  -4.00    ROW   201              7.00
        COL   201  ROW   202             3.00    ROW   203             -1.00
        COL   201  ROW   204             0.00    ROW   205              8.00
        COL   202  OBJ                 -12.00    ROW   201              6.00
        COL   202  ROW   202             0.00    ROW   203             -9.00
        COL   202  ROW   204            16.00    ROW   205             17.00
        COL   203  OBJ                  -4.00    ROW   201             19.00
        COL   203  ROW   202            -6.00    ROW   203            -10.00
        COL   203  ROW   204            14.00    ROW   205             20.00
        COL   204  OBJ                 -15.00    ROW   201             12.00
        COL   204  ROW   202            15.00    ROW   203             15.00
        COL   204  ROW   204             0.00    ROW   205              0.00
        COL   205  OBJ                 -12.00    ROW   201             -2.00
        COL   205  ROW   202             0.00    ROW   203              2.00
        COL   205  ROW   204             8.00    ROW   205             10.00
RHS
        RHS1       ROW   201            31.00    ROW   202             29.00
        RHS1       ROW   203            26.00    ROW   204             23.00
        RHS1       ROW   205            23.00
BOUNDS
      0.    0.    0.    0.    0.
      1.    1.    1.    1.    1.
        SUB     3
ROWS
 N  OBJ
 L  ROW   301
 L  ROW   302
 L  ROW   303
 L  ROW   304
 L  ROW   305
OFFDIAGONAL COLUMNS
        COL   201  ROW   301             9.00    ROW   302             17.00
        COL   201  ROW   303            -9.00    ROW   304              8.00
        COL   201  ROW   305             6.00
        COL   202  ROW   301             0.00    ROW   302             15.00
        COL   202  ROW   303             8.00    ROW   304             20.00
        COL   202  ROW   305            11.00
        COL   203  ROW   301            18.00    ROW   302             -5.00
        COL   203  ROW   303            -3.00    ROW   304             14.00
        COL   203  ROW   305            -6.00
        COL   204  ROW   301            -7.00    ROW   302             -1.00
        COL   204  ROW   303           -10.00    ROW   304              4.00
        COL   204  ROW   305           -10.00
        COL   205  ROW   301            14.00    ROW   302              0.00
        COL   205  ROW   303            -2.00    ROW   304             -4.00
        COL   205  ROW   305             0.00
COLUMNS
        COL   301  OBJ                  -2.00    ROW   301              9.00
        COL   301  ROW   302             0.00    ROW   303              8.00
        COL   301  ROW   304            -6.00    ROW   305              0.00
        COL   302  OBJ                 -12.00    ROW   301              3.00
        COL   302  ROW   302            -6.00    ROW   303              8.00
        COL   302  ROW   304            -2.00    ROW   305              6.00
        COL   303  OBJ                 -18.00    ROW   301              5.00
        COL   303  ROW   302             0.00    ROW   303             -6.00
        COL   303  ROW   304            -4.00    ROW   305              0.00
        COL   304  OBJ                  -2.00    ROW   301             10.00
        COL   304  ROW   302            -6.00    ROW   303             11.00
        COL   304  ROW   304            10.00    ROW   305             16.00
        COL   305  OBJ                 -12.00    ROW   301             -2.00
```

```
    COL   305  ROW   302            5.00   ROW   303             2.00
    COL   305  ROW   304           -4.00   ROW   305           -10.00
RHS
    RHS1       ROW   301           22.00   ROW   302            38.00
    RHS1       ROW   303           21.00   ROW   304            23.00
    RHS1       ROW   305           21.00
BOUNDS
   0.    0.    0.    0.    0.
   1.    1.    1.    1.    1.
    SUB     4
ROWS
 N  OBJ
 L  ROW   401
 L  ROW   402
 L  ROW   403
 L  ROW   404
 L  ROW   405
OFFDIAGONAL COLUMNS
    COL   301  ROW   401            1.00   ROW   402            -2.00
    COL   301  ROW   403           -6.00   ROW   404            14.00
    COL   301  ROW   405            4.00
    COL   302  ROW   401            1.00   ROW   402             4.00
    COL   302  ROW   403           15.00   RCW   404            19.00
    COL   302  ROW   405           10.00
    COL   303  ROW   401           -4.00   ROW   402            11.00
    COL   303  ROW   403           18.00   ROW   404            -3.00
    COL   303  ROW   405           17.00
    COL   304  ROW   401            5.00   ROW   402             6.00
    COL   304  ROW   403            9.00   ROW   404            16.00
    COL   304  ROW   405           14.00
    COL   305  ROW   401            4.00   ROW   402            20.00
    COL   305  ROW   403           10.00   ROW   404             3.00
    COL   305  ROW   405           16.00
COLUMNS
    COL   401  OBJ                 -1.00   ROW   401             7.00
    COL   401  ROW   402           17.00   ROW   403             3.00
    COL   401  ROW   404           12.00   ROW   405            16.00
    COL   402  OBJ                -16.00   ROW   401            12.00
    COL   402  ROW   402           14.00   RCW   403            11.00
    COL   402  ROW   404           12.00   ROW   405           -10.00
    COL   403  OBJ                -12.00   ROW   401            17.00
    COL   403  ROW   402            6.00   ROW   403             4.00
    COL   403  ROW   404           -8.00   ROW   405            12.00
    COL   404  OBJ                -14.00   ROW   401             5.00
    COL   404  ROW   402           10.00   ROW   403            11.00
    COL   404  ROW   404           -4.00   ROW   405            18.00
    COL   405  OBJ                -10.00   ROW   401            16.00
    COL   405  ROW   402           17.00   ROW   403             6.00
    COL   405  ROW   404           -6.00   ROW   405             3.00
RHS
    RHS1       ROW   401           38.00   ROW   402            29.00
    RHS1       ROW   403           29.00   ROW   404            36.00
    RHS1       ROW   405           27.00
BOUNDS
   0.    0.    0.    0.    0.
   1.    1.    1.    1.    1.
EOF
```

```
**** Sample Output ****

                    PROBLEM    1
      20 ROWS    20 COLUMNS     4 PERIODS

PRICE OF OFFDIAGONAL COLUMNS IN SUB.  4
 -2.29    4.57  12.57   6.86  22.86
PRICE OF OFFDIAGONAL COLUMNS IN SUB.  3
  0.00    0.00   0.00   0.00   0.00
PRICE OF OFFDIAGONAL COLUMNS IN SUB.  2
  2.12    9.18   3.53   2.12  -5.65

      MAXIMUM CUMULATIVE OBJECTIVE VALUES
FOR LAMBDA = .5:    196    140    89
FOR LAMBDA =  0:     90     49    42

      INTERMEDIATE SOLUTIONS FOUND
TIME =    0.90 SECONDS;  BRANCHES =        21;  INCVAL =         80
TIME =    0.97 SECONDS;  BRANCHES =        28;  INCVAL =         84
TIME =    0.99 SECONDS;  BRANCHES =        31;  INCVAL =         90
TIME =    1.32 SECONDS;  BRANCHES =        76;  INCVAL =         92
TIME =    1.84 SECONDS;  BRANCHES =       145;  INCVAL =        105

TOTAL SOLUTION TIME =     2.12 SECONDS
TIME FOR INPUT =    0.44 SECONDS
TIME FOR LP SOLUTIONS =   0.31 SECONDS
TIME FOR BOUNDING & PRICING =   0.32 SECONDS
TIME FOR BOOKKEEPING OPERATIONS =   0.33 SECONDS
NUMBER OF FORWARD STEPS TAKEN =     80
TOTAL NUMBER OF BRANCHES TAKEN =       174

      OPTIMAL INTEGER SOLUTION
SUBPROBLEM    1
  18   25    6   23   16   43    0    0    0    1    1
SUBPROBLEM    2
  39    2    0   20    0    1    0    1    0    1    1
SUBPROBLEM    3
  20    1   24   23   13   20    1    0    1    0    0
SUBPROBLEM    4
  28   12    0    2   21    4    0    1    1    0    0
MAX OBJECTIVE VALUE =   105
```

```
C
C        STAIRCASE-STRUCTURED MATRIX BRANCH-AND-BOUND CODE
C        PURE-INTEGER LINEAR PROGRAMMING IN GENERAL INTEGER VARIABLES
C        SDA: TWO BOUNDING PROCEDURES, EXTRA TIMERS IF NEEDED
C        WRITTEN BY LYNNE POLLENZ, LAST UPDATED MAY 1979
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
      COMMON/TIMERS/ITOT,TSTORE,TIMELP,TIMEDR,TIMEDC,TIMINV
C
C   MAIN PROGRAM
C
C   START TIMER
100     ITOT = IHPTIM(1)
C   INPUT PROBLEM DATA
      CALL INPUT(IFPROB,INITBD)
      ITIMIN = IHPTIM(1)
      IF (IFPROB .EQ. 0) GO TO 1000
      CALL BOUNDR
      ITIMLP = IHPTIM(1)
      TSTORE = 0.
      TIMELP = 0.
C UNDER THE SCORE COMPILER, 'D' IN THE FIRST COLUMN IS READ AS 'C'
C  UNLESS A SPECIAL OPTION IS USED, IN WHICH CASE IT IS TREATED AS A ' '.
D     TIMINV = 0.
D     TIMEDC = 0.
D     TIMEDR = 0.
      NBRANC = 0
      NFORWD = 1
C
C   APPLY BRANCH-AND-BOUND SEARCH ROUTINE, STARTING IN PERIOD 1
C
      NS = 1
      CALL RESTOR(1)
      CALL BANDR(INITBD)
30    IF (LISTL.EQ.0) GO TO 500
C
C   A PARTIAL SOLUTION HAS BEEN FOUND. GO ON TO THE NEXT PERIOD.
C
      NS = NS + 1
      NFORWD = NFORWD + 1
      CALL RESTOR(1)
      CALL BBFNTR
      GO TO 30
C
C   STOP TIMER, ALL DONE. REPORT TIMING INFORMATION.
C
500   ITIM2 = IHPTIM(1)
      TOT = (ITIM2-ITOT)/100000.
      WRITE (21,1) TOT
1     FORMAT (/' TOTAL SOLUTION TIME =',F8.2,' SECONDS')
      TIMINP = (ITIMIN - ITOT)/100000.
      WRITE (21,2) TIMINP
2     FORMAT(' TIME FOR INPUT =',F7.2,' SECONDS')
      TIMPRC = (ITIMLP - ITIMIN)/100000.
      WRITE (21,3) TIMELP
3     FORMAT(' TIME FOR LP SOLUTIONS =',F7.2,' SECONDS')
      WRITE (21,4) TIMPRC
```

```
4     FORMAT(' TIME FOR BOUNDING & PRICING =',F7.2,' SECONDS')
      TSTORE = TSTORE - TIMELP
      WRITE (21,5) TSTORE
5     FORMAT(' TIME FOR BOOKKEEPING OPERATIONS =',F7.2,' SECONDS')
      WRITE (21,7) NFORWD
7     FORMAT(' NUMBER OF FORWARD STEPS TAKEN =',I7)
      WRITE (21,8) NBRANC
8     FORMAT(' TOTAL NUMBER OF BRANCHES TAKEN =',I10)
D     WRITE (21,9) TIMINV
D9    FORMAT(' TIME SPENT INVERTING BASIS =',F7.2,' SECONDS')
D     WRITE(21,10) TIMEDC
D10   FORMAT(' TIME SPENT IN SUBROUTINE DCHUZC =',F7.2,' SECONDS')
D     WRITE(21,11) TIMEDR
D11   FORMAT(' TIME SPENT IN SUBROUTINE DCHUZR =',F7.2,' SECONDS')
C   OUTPUT OPTIMAL SOLUTION
      CALL WRAPUP
      GO TO 100
1000  STOP
      END
C-----------------------------------------------------------------------
      BLOCK DATA
C
C         INITIALIZE GLOBAL PROGRAM CONSTANTS
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C         BY J.A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      COMMON/TOTSIZ/ MAXTRW,MAXTCL,MAXNP,MAXELE,MAXAEL
      COMMON/CONSTS/ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      DATA MAXTRW/70/,MAXTCL/130/,MAXNP/10/,MAXELE/2000/,MAXAEL/1000/
      DATA ZTOLZE/1.E-5/,ZTOLPV/1.E-4/,ZTCOST/1.E-3/,ZTOLSM/1.E-10/
      DATA NRMAX/60/,NTMAX/500/,NEGINF/-100000/
      DATA QBL/'    '/,QA/'A   '/,QI/'I   '/,QF/'F   '/,QN/'N   '/,
     1  QSUB/'SUB '/,QB/'B   '/,QC/'C   '/,QE/'E   '/,QH/'H   '/,
     2  QL/'L   '/,QO/'O   '/,QR/'R   '/,QM/'M   '/,QG/'G   '/
      END
C-----------------------------------------------------------------------
      SUBROUTINE INPUT(IFPROB,INITBD)
C
C         READ ALL PROBLEM DATA
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C         ***DESCRIPTION OF PARAMETERS***
C         IFPROB = NONZERO PROBLEM ID NUMBER (OUTPUT)
C         INITBD = INITIAL LOWER BOUND ESTIMATE FOR MAXIMAL OBJECTIVE
C                  VALUE (OUTPUT)
C         DATA MUST BE INPUT IN THE FOLLOWING ORDER: PROBLEM CONSTANTS,
C         FOR EACH PERIOD: ROWS,OFFDIAG. COLS,DIAGONAL COLS,RHS,BOUNDS
C         THE OBJECTIVE ROW MUST BE THE FIRST ROW IN EACH SUBPROBLEM
C         VARIABLES LOCAL TO ONE PERIOD MUST APPEAR AFTER VARIABLES
C          OF THAT PERIOD WHICH ALSO APPEAR IN THE FOLLOWING PERIOD.
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER NAME(6),NAMCOL(130,2)
      DOUBLE PRECISION E(2000),ATEMP1,ATEMP2
      REAL A(1000)
```

```
C
      COMMON/TOTSI2/ MAXTRW,MAXTCL,MAXNP,MAXELE,MAXAEL
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
      COMMON/OFFDAG/OFFD(2000),LCOLOD(130),IROWOD(2000),COST(130)
C
C         ***DESCRIPTIONS OF LOCAL VARIABLES***
C         JCPTR POINTS TO CURRENT OFFDIAGONAL INPUT COLUMN
C         NAMCOL(I,*) = NAME OF COLUMN I
C         ICS1,ICS2 = NAME OF CURRENT COLUMN
C         NTCOL = TOTAL NUMBER OF COLUMNS SO FAR
C         NCOL1 = FIRST COLUMN OF CURRENT SUBPROBLEM
C         NROWL = LAST SLACK COLUMN (AND LAST ROW) OF CURRENT SUBPROBLEM
C         MAXTRW = MAX TOTAL NUMBER OF ROWS (INCLUDING NS OBJECTIVE ROWS)
C         MAXTCL = MAX TOTAL NUMBER OF COLUMNS INCLUDING SLACK COLUMNS
C         MAXNP = MAXIMUM NUMBER OF SUBPROBLEMS
C         MAXAEL = MAX. NUMBER OF NONZERO ELEMENTS OF ANY DIAGONAL BLOCK
C         MAXELE = MAXIMUM SUM OF NONZERO ENTRIES IN ALL DIAGONAL BLOCKS
C
C         ***DESCRIPTIONS OF SOME IMPORTANT VARIABLES IN BLANK COMMON***
C         B(I) = RIGHT HAND SIDE OF ROW I [IN CURRENT SUBPROBLEM]
C         X(I) = LP VALUE FOR JH(I), WHICH IS THE VARIABLE BASIC IN ROW I
C         A CONTAINS THE NONZERO ELEMENTS OF THE CONSTRAINT MATRIX, INCL.
C           THE OBJECTIVE ROW IOBJ. LA(J) = LOCATION IN A OF THE FIRST
C           ELEMENT OF COL J. IA(I) = ROW IN WHICH ELEMENT I OF A BELONGS.
C         E CONTAINS THE NONZERO ELEMENTS OF THE CURRENT LP BASIS INVERSE
C           IN ETA VECTOR FORM. LE, IE ARE TO E AS LA, IA ARE TO A.
C         MSTAT FLAGS FEASIBILITY OF CURRENT LP
C         ITCNT = NO. OF SIMPLEX ITERATIONS SO FAR; IF > ITRFRQ, STOP
C         INVFRQ = NUMBER OF SIMPLEX ITERATIONS BEFORE E IS REINVERTED
C         NROW = NO. OF ROWS; NCOL = NO. OF COLUMNS IN CURRENT SUBPROBLEM
C         KINBAS(J) = ( I IF J IS BASIC IN ROW I, I.E. J = JH(I)
C                       0 IF J IS NONBASIC AT ITS LOWER BOUND XLB(J)
C                      -1 IF J IS NONBASIC AT ITS UPPER BOUND XUB(J) )
C         ***DESCRIPTIONS OF SOME VARIABLES IN GESTALT COMMON***
C         JFCOL(I),JFROW(I),JFELEM(I) ARE FIRST COL, ROW, ELEMENT OF A,
C          RESPECTIVELY, FOR PERIOD I
C         NP = TOTAL NUMBER OF PERIODS (OR EQUIVALENTLY, SUBPROBLEMS)
C         NS = CURRENT SUBPROBLEM NUMBER
C
C    INITIALIZATIONS
C
      ITCNT = 0
      NS = 0
      NOELEM = 0
      NTCOL = 0
      JFCOL(1) = 1
      JFROW(1) = 1
      JFELEM(1) = 1
      DO 1 I=1,MAXTCL
1        COST(I) = 0.
      READ (20,7000,END=9999) IFPROB,NP,IOBJ,INVFRQ,ITRFRQ,INITBD
7000  FORMAT (4I4,I5,I10)
      IF (IFPROB .EQ. 0) RETURN
```

```
      IF (NP.GT.MAXNP) GO TO 9996
      NEMAX = 4000/NP
      NLES = 1000/NP
      IF (IOBJ .EQ. 0)IOBJ = 1
      IF (IOBJ.NE.1) GO TO 9995
      IF (INVFRQ .EQ. 0)INVFRQ = 99999
      IF (ITRFRQ .EQ. 0)ITRFRQ = 999999
      WRITE(21,8010) IFPROB
8010  FORMAT(/19X,'PROBLEM ',I4)
C
C   INITIALIZE FOR READING EACH SUBPROBLEM
C
5     IF (NS.EQ.NP) GO TO 25
      NROW = 0
      ICS1 = 0
      ICS2 = 0
      DO 10 I=1,NRMAX
10       B(I) = 0.
      NS = NS + 1
      NCOL1 = JFCOL(NS)
      READ(20,99) (NAME(I),I=1,3)
99    FORMAT(2A4,I4)
      IF (NAME(2).NE.QSUB .OR. NAME(3).NE.NS ) GO TO 9998
C
C   READ IN DATA FOR SUBPROBLFM NS
C
25    READ(20,101) K1,K2,K3,K4,(NAME(I),I=1,4),ATEMP1,NAME(5),NAME(6),
     1  ATEMP2
101   FORMAT(4A1,2A4,2X,2A4,2X,F12.4,3X,2A4,2X,F12.4)
      IF(K1 .EQ. QBL) GO TO 50
      IF(K1 .EQ. QR .AND. K2 .EQ. QO) L=1
      IF(K1 .EQ. QR .AND. K2 .EQ. QO) GO TO 25
      IF(K1 .EQ. QC) GO TO 150
      IF(K1 .EQ. QO) GO TO 160
      IF(K1 .EQ. QR .AND. K2 .EQ. QH) L=4
      IF(K1 .EQ. QR .AND. K2 .EQ. QH) GO TO 25
      IF(K1 .EQ. QB) GO TO 600
      IF(K1 .EQ. QE) GO TO 700
      WRITE(21,8020) K1,K2,K3,K4,NS
8020  FORMAT(' IMPROPER INPUT ',4A1,' IN SUBPROBLEM',I4)
      GO TO 9999
50    GO TO(210,320,400,500),L
C
150   L = 2
      NROWL = NCOL1 + NROW - 1
      GO TO 25
C
160   L = 3
      ICS1 = -9999
      GO TO 25
C
C   READ ROW NAMES ( = SLACK COLUMN NAMES)
C
210   NROW=NROW+1
      NCOL=NROW
      NTCOL = NTCOL + 1
      NAMCOL(NTCOL,1) = NAME(1)
      NAMCOL(NTCOL,2) = NAME(2)
C
C   TEST ROW TYPE: <=,=,>=, OR OBJ. ROW; SET SLACK BOUNDS
```

```
C
      IF(K2.EQ.QL .OR. K3.EQ.QL)  GO TO 220
      IF(K2.EQ.QE .OR. K3.EQ.QE)  GO TO 230
      IF(K2.EQ.QG .OR. K3.EQ.QG)  GO TO 240
      IF(K2.EQ.QN .OR. K3.EQ.QN)  GO TO 250
      GO TO 230
220   XLB(NROW) = 0.
      XUB(NROW) = 1.E4
      GO TO 250
230   XLB(NROW) = 0.
      XUB(NROW) = 0.
      GO TO 250
240   XLB(NROW) = 0.
      XUB(NROW) = 1.E4
      A(NROW) = -1.
      GO TO 260
250   A(NROW) = 1.
260   IA(NROW) = NROW
      LA(NROW) = NROW
      NELEM=NROW
      GO TO 25
C
C   MATRIX ELEMENTS
C
320   J = 3
      K = 4
      IF (DABS(ATEMP1) .GT. ZTOLZE)  GO TO 324
      J=5
      K=6
      IF(DABS(ATEMP2) .LE. ZTOLZE)  GO TO 25
      ATEMP1=ATEMP2
C        TEST FOR COLUMN MATCH
324   IF (NAME(1) .EQ. ICS1 .AND. NAME(2) .EQ. ICS2)  GO TO 330
      NCOL = NCOL + 1
      NTCOL = NTCOL + 1
      ICS1 = NAME(1)
      ICS2 = NAME(2)
      NAMCOL(NTCOL,1) = ICS1
      NAMCOL(NTCOL,2) = ICS2
      LA(NCOL) = NELEM + 1
C        RECORD OBJECTIVE VALUE
330   IF (NAME(J) .EQ. NAMCOL(NCOL1+IOBJ-1,1)) COST(NTCOL) = ATEMP1
C        TEST FOR ROW MATCH
      DO 340 I = NCOL1,NROWL
            IF (NAME(J).NE.NAMCOL(I,1).OR.NAME(K).NE.NAMCOL(I,2))GO TO 340
            NELEM = NELEM + 1
            IA(NELEM) = I - NCOL1 + 1
            A(NELEM) = ATEMP1
            LA(NCOL+1)=NELEM+1
            IF(K .GT. 5)  GO TO 25
            IF(DABS(ATEMP2) .LE. ZTOLZE)  GO TO 25
            J = 5
            K = 6
            ATEMP1 = ATEMP2
            GO TO 330
340      CONTINUE
      WRITE(21,8300) NAME(J),NAME(K),NAME(1),NAME(2)
8300  FORMAT(17HNO MATCH FOR ROW ,2A4,11H AT COLUMN ,2A4)
      GO TO 9999
C
```

```
C    OFFDIAGONAL MATRIX ELEMENTS
C
400   J = 3
      K = 4
      IF (DABS(ATEMP1) .GT. ZTOLZE) GO TO 420
      J = 5
      K = 6
      IF(DABS(ATEMP2) .LE. ZTOLZE)  GO TO 25
      ATEMP1 = ATEMP2
C         TEST FOR COLUMN MATCH
420   IF (NAME(1).EQ.ICS1 .AND. NAME(2).EQ.ICS2) GO TO 450
      ICS1 = NAME(1)
      ICS2 = NAME(2)
      IBEG = JCPTR + 1
      IEND = JFCOL(NS) - 1
      IF (IBEG.GT.IEND) GO TO 435
      DO 430 I=IBEG,IEND
              LCOLOD(I) = NOELEM + 1
              IF (ICS1.EQ.NAMCOL(I,1) .AND. ICS2.EQ.NAMCOL(I,2)) GO TO 440
430       CONTINUE
435   WRITE(21,8250) ICS1,ICS2,NS
8250  FORMAT(' NO MATCH FOR COLUMN ',2A4,' IN SUBPROBLEM',I4)
      GO TO 9999
440   JCPTR = I
C         TEST FOR ROW MATCH
450   DO 460 I=NCOL1,NTCOL
              IF(NAME(J).NE.NAMCOL(I,1).OR.NAME(K).NE.NAMCOL(I,2)) GO TO 460
              NOELEM = NOELEM + 1
              IROWOD(NOELEM) = I - NCOL1 + 1
              OFFD(NOELEM) = ATEMP1
              LCOLOD(JCPTR + 1) = NOELEM + 1
              IF (K.GT.5) GO TO 25
              IF(DABS(ATEMP2) .LE. ZTOLZE) GO TO 25
              J = 5
              K = 6
              ATEMP1 = ATEMP2
              GO TO 450
460       CONTINUE
      WRITE(21,8300) NAME(J),NAME(K),NAME(1),NAME(2)
      GO TO 9999
C
C    RIGHT HAND SIDE
C
500   J = 3
      K = 4
      IF (DABS(ATEMP1) .GT. ZTOLZE)  GO TO 530
      J=5
      K=6
      IF(DABS(ATEMP2) .LE. ZTOLZE)  GO TO 25
      ATEMP1=ATEMP2
C         TEST FOR ROW MATCH
530   DO 540 I = NCOL1,NROWL
              IF(NAME(J).NE.NAMCOL(I,1).OR.NAME(K).NE.NAMCOL(I,2))GO TO 540
              B(I-NCOL1+1) = ATEMP1
              IF(K .GT. 5)  GO TO 25
              IF(DABS(ATEMP2) .LE. ZTOLZE)  GO TO 25
              J = 5
              K = 6
              ATEMP1 = ATEMP2
              GO TO 530
```

```
540       CONTINUE
      WRITE(21,8300) NAME(J),NAME(K)
      GO TO 9999
C
C   BOUNDS ON INTEGER VARIABLES
C
600   K = NROW + 1
C        INPUT LOWER AND UPPER BOUNDS ON DECISION VARIABLES
      READ (20,8500) (XLB(J), J=K,NCOL)
      READ (20,8500) (XUB(J), J=K,NCOL)
8500  FORMAT (15F5.0)
C   CHECK PROBLEM SIZE
      JFCOL(NS+1) = JFCOL(NS) + NCOL
      JFROW(NS+1) = JFROW(NS) + NROW
      JFELEM(NS+1) = JFELEM(NS) + NELEM
      IF (NROW.GT.NRMAX) GO TO 9996
      IF (NELEM.GT.MAXAEL) GO TO 9996
      IF (JFROW(NS+1) .GT. (MAXTRW + 1)) GO TO 9996
      IF (JFCOL(NS+1) .GT. (MAXTCL + 1)) GO TO 9996
      IF (JFELEM(NS+1) .GT. (MAXELE + 1)) GO TO 9996
      JCPTR = JFCOL(NS) + NROW - 1
      CALL INPSTO
      GO TO 5
C
C   END OF INPUT
C
700   NTROW = JFROW(NP+1) - 1 - NP
      NSCOLS = JFCOL(NP+1)  - JFROW(NP+1)
      WRITE(21,8800) NTROW,NSCOLS,NP
8800  FORMAT(5X,I4,' ROWS',2X,I4,' COLUMNS',2X,I4,' PERIODS',/)
      RETURN
C
C   ERROR MESSAGES FOR INPUT ERRORS
C
9995  WRITE(21,8960)
8960  FORMAT(' OBJECTIVE ROW MUST BE THE FIRST ROW')
      GO TO 9999
9996  WRITE(21,8970)
8970  FORMAT(' PROBLEM IS TOO LARGE FOR CURRENT DIMENSIONING')
      WRITE(21,8975) MAXNP,MAXTRW,MAXTCL,MAXELE
8975  FORMAT('MAX SUBS=',I3,'MAX ROWS=',I4,'MAX COLS=',I4,'MAX ELE=',I6)
      GO TO 9999
9998  WRITE(21,8990) NS
8990  FORMAT(' SUBPROBLEM',I4,' NOT IN PROPER POSITION')
9999  IERROR = 0
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE INPSTO
C
C        STORE ALL DATA RELEVANT TO SUBPROBLEM NS
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE,LCOLA,IROWA,IESTOR,LESTOR
      DOUBLE PRECISION X(2000),ESTORE(4000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  QLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QF,QH,QL,QO,QR,QM,QG
```

```
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1 E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2 NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3 KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1 NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
      COMMON/TIMERS/ITOT,TSTORE,TIMELP,TIMEDR,TIMEDC,TIMINV
      COMMON/SUBSAV/BMOD(70),BORIG(70),XSTORE(70),XUBSTO(130),
     1 XLBSTO(130),XUBORG(130),XLBORG(130),ASTORE(2000),ESTORE,
     2 IROWA(2000),LCOLA(140),IESTOR(4000),LESTOR(1002),JHSTOR(70),
     3 INBSTO(130),IPARTC(10),INVSTO(10),NELSTO(10),NETSTO(10)
C
C   COMMON BLOCK SUBSAV IS A STORAGE AREA FOR THE SUBPROBLEM VARIABLES
C   XLBORG,XUBORG,BORIG STORE THE ORIGINAL VALUES OF THE CORRESP. VARS.
C   BMOD(I) = RHS FOR ROW I AFTER OFFDIAGONAL VALUES HAVE BEEN ADDED
C
C   AFTER READING INPUT, STORE THE DIAGONAL MATRIX ELEMENTS AND RHS
C
      IPTR = JFELEM(NS)
      LENGTH = JFELEM(NS + 1) - IPTR
      DO 10 I=1,LENGTH
            ASTORE(IPTR) = A(I)
            IROWA(IPTR) = IA(I)
10       IPTR = IPTR + 1
      IPTR = JFCOL(NS) - 2 + NS
      IEND = NCOL + 1
      DO 15 I=1,IEND
15       LCOLA(IPTR + I) = LA(I)
C   STORE ORIGINAL RHS B; BOUNDS XUB,XLB
      IPTR = JFROW(NS)
      DO 20 I=1,NROW
            BORIG(IPTR) = B(I)
20       IPTR = IPTR + 1
      IPTR = JFCOL(NS)
      DO 30 I=1,NCOL
            XUBORG(IPTR) = XUB(I)
            XLBORG(IPTR) = XLB(I)
30       IPTR = IPTR + 1
      RETURN
C
C   STORE THE CURRENT STATE OF SUBPROBLEM NS
C
      ENTRY STORE
C   STORE THE PARTIAL OBJ. VALUE ISUMC; RHS B; SOLUTION X
      ITIME = IHPTIM(1)
      IPARTC(NS) = ISUMC
      IPTR = JFROW(NS)
      DO 300 I=1,NROW
            BMOD(IPTR) = B(I)
            XSTORE(IPTR) = X(I)
            JHSTOR(IPTR) = JH(I)
300      IPTR = IPTR + 1
C   STORE BASIC VAR INDICATOR KINBAS; BOUNDS XUB,XLB
      IPTR = JFCOL(NS)
      DO 350 I=1,NCOL
            INBSTO(IPTR) = KINBAS(I)
            XUBSTO(IPTR) = XUB(I)
            XLBSTO(IPTR) = XLB(I)
350      IPTR = IPTR + 1
C   STORE BASIS INVERSE. IF TOO LARGE, REINVERT.
```

```
      IF (NETA.LT.NLES) GO TO 390
      CALL INVERT
      ITSINV = 0
390   INVSTO(NS) = ITSINV
      NELSTO(NS) = NELEM
      NETSTO(NS) = NETA
      LEPTR = (NS-1) * NLES
      JEPTR = (NS-1) * NEMAX
      DO 400 I=1,NELEM
            JEPTR = JEPTR + 1
            ESTORE(JEPTR) = E(I)
400      IESTOR(JEPTR) = IE(I)
      IEND = NETA + 1
      DO 450 I=1,IEND
            LEPTR = LEPTR + 1
450      LESTOR(LEPTR) = LE(I)
      ITIME2 = IHPTIM(1)
      TSTORE = TSTORE + (ITIME2-ITIME)/100000.
      RETURN
      END
C----------------------------------------------------------------------
      SUBROUTINE RESTOR(MNFLAG)
C
C        RESTORE ALL DATA RELEVANT TO SUBPROBLEM NS
C        ***DESCRIPTION OF PARAMETER***
C        MNFLAG = 0 IF BACKTRACKING, 1 IF TAKING A FORWARD STEP (INPUT)
C        IF TAKING A FORWARD STEP, RESTORE ORIGINAL LP-OPT. IF TAKING
C         THIS STEP FOR THE FIRST TIME, COMPUTE ORIGINAL LP SOLUTION.
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE,LCOLA,IROWA,IESTOR,LESTOR
      DOUBLE PRECISION E(2000),ESTORE(4000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),TSUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
      COMMON/TIMERS/ITOT,TSTORE,TIMELP,TIMEDR,TIMEDC,TIMINV
      COMMON/SUBSAV/BMOD(70),BORIG(70),XSTORE(70),XUBSTO(130),
     1  XLBSTO(130),XUBORG(130),XLBORG(130),ASTORE(2000),ESTORE,
     2  IROWA(2000),LCOLA(140),IESTOR(4000),LESTOR(1002),JHSTOR(70),
     3  INBSTO(130),IPARTC(10),INVSTO(10),NELSTO(10),NETSTO(10)
C
C    RESTORE SUBPROBLEM DIMENSIONS; A MATRIX
C
      ITIME = IHPTIM(1)
      NROW = JFROW(NS+1) - JFROW(NS)
      NCOL = JFCOL(NS+1) - JFCOL(NS)
      IPTR = JFELEM(NS)
      LENGTH = JFELEM(NS+1) - IPTR
C    RESTORE ELEMENTS OF A MATRIX FROM DIAGONAL BLOCK NS
      DO 10 I=1,LENGTH
            A(I) = ASTORE(IPTR)
            IA(I) = IROWA(IPTR)
```

```
10       IPTR = IPTR + 1
      IPTR = JFCOL(NS) + NS - 2
      IEND = NCOL + 1
      DO 15 I=1,IEND
15       LA(I) = LCOLA(IPTR + I)
      JEPTR = (NS-1) * NEMAX
      LEPTR = (NS-1) * NLES
      MSTAT = QBL
      IF (MNFLAG.EQ.1) GO TO 200
C
C   BACKTRACK STEP: RESTORE RHS B; LP SOLUTION X,JH,KINBAS; BOUNDS
C
      IPTR = JFROW(NS)
      DO 20 I=1,NROW
              B(I) = BMOD(IPTR)
              X(I) = XSTORE(IPTR)
              JH(I) = JHSTOR(IPTR)
20       IPTR = IPTR + 1
      IPTR = JFCOL(NS)
      DO 40 I=1,NCOL
              XUB(I) = XUBSTO(IPTR)
              XLB(I) = XLBSTO(IPTR)
              KINBAS(I) = INBSTO(IPTR)
40       IPTR = IPTR + 1
C   RESTORE PARTIAL OBJ. VALUE IF BACKTRACKING (NOT FOR FORWARD STEPS)
      ISUMC = IPARTC(NS)
C
C   RESTORE LP BASIS INVERSE
C
      ITSINV = INVSTO(NS)
      NELEM = NELSTO(NS)
      NETA = NETSTO(NS)
      DO 110 I=1,NELEM
              JEPTR = JEPTR + 1
              E(I) = ESTORE(JEPTR)
110      IE(I) = IESTOR(JEPTR)
      IEND = NETA + 1
      DO 120 I=1,IEND
              LEPTR = LEPTR + 1
120      LE(I) = LESTOR(LEPTR)
      GO TO 500
C
C   FORWARD STEP
C
C   RESTORE ORIGINAL BOUNDS FOR A FORWARD STEP
200   IPTR = JFCOL(NS)
      DO 210 I=1,NCOL
              XUB(I) = XUBORG(IPTR)
              XLB(I) = XLBORG(IPTR)
210      IPTR = IPTR + 1
      IPTR = JFROW(NS) - 1
      DO 220 I=1,NROW
220      B(I) = BORIG(IPTR + I)
C
C   SOLVE LP RELAXATION OF SUBPROBLEM NS
C
      IF (NS.GT.1) CALL FIXRHS
C   LP BASIS STARTS OFF AS ALL SLACK BASIS
      ITIMLP = IHPTIM(1)
      DO 310 I=1,NROW
```

```
310       JH(I) = I
      DO 320 I=1,NROW
320       KINBAS(I) = I
      NROWP1 = NROW + 1
      DO 330 I=NROWP1,NCOL
330       KINBAS(I) = 0
C
C   SOLVE LP
C
      ITSINV = 99999
      CALL NORMAL(ITSINV)
      IF (MSTAT.EQ.QN) GO TO 2000
      ITIML2 = IHPTIM(1)
      TIMELP = TIMELP + (ITIML2-ITIMLP)/100000.
C
500   ITIME2 = IHPTIM(1)
      TSTORE = TSTORE + (ITIME2-ITIME)/100000.
      RETURN
C
C   LP IS INFEASIBLE
C
2000  IF (NS.EQ.1) WRITE(21,2010)
2010  FORMAT(' SUBPROBLEM 1 IS INFEASIBLE')
      GO TO 500
      END
C-----------------------------------------------------------------------
      SUBROUTINE FIXRHS
C
C         GIVEN SETTING OF VARIABLES FOR PERIOD (NS-1), COMPUTE NEW
C          RHS FOR PERIOD NS IN PREPARATION FOR A FORWARD STEP.
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1   NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
      COMMON/OFFDAG/OFFD(2000),LCOLOD(130),IROWOD(2000),COST(130)
C
C   START AT FIRST NONSLACK COLUMN OF LAST PERIOD
      JBEG = JFCOL(NS-1) + (JFROW(NS) - JFROW(NS-1))
      JEND = JFCOL(NS) - 1
C
C   COMPUTE CONTRIBUTION TO RHS FROM LAST PERIOD
C
      DO 100 J=JBEG,JEND
C   UNPACK OFFDIAGONAL COLUMN J IF NONZERO
              IBEG = LCOLOD(J)
              IEND = LCOLOD(J+1) - 1
              IF (IBEG.GT.IEND) GO TO 100
              DO 20 I=1,NROW
20            YTEMP(I) = 0.
              DO 30 I=IBEG,IEND
                 IR = IROWOD(I)
```

```
30          YTEMP(IR) = OFFD(I)
C    MULTIPLY COLUMN J BY ICURX(J), THE VALUE OF X(J) AS LAST SET
            YTEMP(IOBJ) = 0.
            XVALUE = ICURX(J)
            DO 40 I=1,NROW
40          B(I) = B(I) - (YTEMP(I) * XVALUE)
100      CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BOUNDR
C
C         CALCULATE 2 LP BASED BOUNDS ON MAX OBJECTIVE VALUES FOR PERIODS
C          NP,NP-1,...,K FOR K = NP,...,2 INCLUDING OFFDIAGONAL COLUMNS.
C          ALSO PRICE OUT THE OFFDIAGONAL COLUMNS.
C         THIS SUBROUTINE ASSUMES OBJECTIVE ROW = 1 IN EACH SUBPROBLEM,
C          AND THAT VARIABLES LOCAL TO PERIOD NS ARE NUMBERED AFTER
C          THOSE VARS. WHICH HAVE NONZERO ENTRIES IN PERIODS NS & NS+1.
C         ALSO TOTAL NUMBER OF NONZERO ELEMENTS IN THE CONSTRAINT
C          MATRIX INCLUDING OFFDIAGONAL COLUMNS MUST NOT EXCEED 1000
C          (FOR CURRENTLY DIMENSIONING).
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE,LCOLA,IROWA,IESTOR,LESTOR
      DOUBLE PRECISION E(2000),ESTORE(4000)
      REAL A(1000)
C
      COMMON/CONSTS/ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
      COMMON/SUBSAV/BMOD(70),BORIG(70),XSTORE(70),XUBSTO(130),
     1  XLBSTO(130),XUBORG(130),XLBORG(130),ASTORE(2000),ESTORE,
     2  IROWA(2000),LCOLA(140),IESTOR(4000),LESTOR(1002),JHSTOR(70),
     3  INBSTO(130),IPARTC(10),INVSTO(10),NELSTO(10),NETSTO(10)
      COMMON/OFFDAG/OFFD(2000),LCOLOD(130),IROWOD(2000),COST(130)
C
C    MAXC(K) = WEIGHTED LP BOUND ON MAX OBJ VALUE FOR SUBS K+1,...,NP
C     OFFDIAGONAL VARIABLES OF SUBPROBLEM K+1 HAVE A WEIGHT OF 1,
C     ALL OTHER VARIABLES HAVE A WEIGHT OF 2. THUS AN LP BOUND ON
C     THE OBJ. VALUE FOR DIAGONAL BLOCK K + MAXC(K) IS A VALID
C     BOUND ON DOUBLE THE OBJ. VALUE FOR PERIODS K,K+1,...,NP.
C     (IN TECH. REPORT BY POLLENZ [1980], THIS CORRESPONDS TO
C     MAXC(1/2,K).)
C    MAXC2(K) = LP BOUND ON MAX OBJ VALUE FOR SUBS K+1,...,NP. PERIOD K
C     VARIABLES HAVE COST 0, THOSE FROM PERIODS > K HAVE ORIGINAL COSTS.
C     ( THIS CORRESPONDS TO MAXC(0,K) IN THE TECH. REPORT.)
C
      MAXC(NP) = 0
      MAXC2(NP) = 0
      LASTC = JFCOL(NP+1) - 1
      DO 20 J=1,LASTC
20       PRICE(J) = 0.
      NROW = 0
C    READ IN SLACK COLUMNS; SKIP OBJ. SLACKS FOR PERIODS 2,...,NP
```

```
      DO 40 NS=1,NP
              IBEG = JFELEM(NS)
              IEND = IBEG + JFROW(NS+1) - JFROW(NS) - 1
              IF (NS.NE.1) IBEG = IBEG + 1
              DO 30 I=IBEG,IEND
                 NROW = NROW + 1
                 A(NROW) = ASTORE(I)
                 LA(NROW) = NROW
                 IA(NROW) = NROW
                 JH(NROW) = NROW
                 KINBAS(NROW) = NROW
                 B(NROW) = 0.
                 JCOL = JFCOL(NS) + 1 + I - IBEG
                 IF (NS.EQ.1) JCOL = JCOL - 1
                 XUB(NROW) = XUBORG(JCOL)
30            XLB(NROW) = XLBORG(JCOL)
40         CONTINUE
      NS = NP
      NCOL = NROW
      NAPTR = NROW + 1
C
C    READ RIGHT HAND SIDE FOR PERIOD NS
C
100   IEXTRA = NS - 1
      IBEG = JFROW(NS) - NS + 2
      IEND = JFROW(NS+1) - NS
      DO 150 IROW=IBEG,IEND
150      B(IROW) = BORIG(IROW + IEXTRA)
C
C    ADD ON FULL COLUMN FOR NONSLACK PERIOD NS VARIABLES
C
      JBEG = JFCOL(NS) + (JFROW(NS+1) - JFROW(NS))
      JEND = JFCOL(NS+1) - 1
      DO 300 J=JBEG,JEND
C        FIRST ADD ON OFFDIAGONAL ELEMENTS OF COLUMN J IF ANY
              NCOL = NCOL + 1
              LA(NCOL) = NAPTR
              IF (NS.EQ.NP) GO TO 220
              IBEG = LCOLOD(J)
              IEND = LCOLOD(J+1) - 1
              IF (IBEG.GT.IEND) GO TO 220
              DO 210 I=IBEG,IEND
                 IF (IROWOD(I).EQ.IOBJ) GO TO 210
                 A(NAPTR) = OFFD(I)
                 IA(NAPTR) = IROWOD(I) + JFROW(NS+1) - (NS+1)
                 NAPTR = NAPTR + 1
210           CONTINUE
C        NEXT ADD ON DIAGONAL ELEMENTS OF COLUMN J
220         IBEG = LCOLA(J+NS-1) + JFELEM(NS) - 1
              IEND = LCOLA(J+NS) + JFELEM(NS) - 2
              DO 250 I=IBEG,IEND
                 A(NAPTR) = ASTORE(I)
                 IF (IROWA(I).NE.IOBJ) GO TO 230
                 A(NAPTR) = A(NAPTR) * 2.0
                 IA(NAPTR) = IOBJ
                 GO TO 240
230           IA(NAPTR) = IROWA(I) + JFROW(NS) - NS
240           NAPTR = NAPTR + 1
250           CONTINUE
               KINBAS(NCOL) = 0
```

```
              XUB(NCOL) = XUBORG(J)
              XLB(NCOL) = XLBORG(J)
300       CONTINUE
      LA(NCOL+1) = NAPTR
C    RESOLVE LP WITH PERIOD NS VARIABLES ADDED
      ITSINV = 99999
      CALL NORMAL(ITSINV)
C
C    ADD OFFDIAGONAL COLUMNS FROM PERIOD NS-1 TO LP
C
      NAPTRO = NAPTR
      NTEMPC = NCOL
C    START AT FIRST NONSLACK COLUMN OF PERIOD NS-1
      JBEG = JFCOL(NS-1) + (JFROW(NS) - JFROW(NS-1))
      JEND = JFCOL(NS) - 1
      DO 430 J=JBEG,JEND
              IBEG = LCOLOD(J)
              IEND = LCOLOD(J+1) - 1
              NCOL = NCOL + 1
              LA(NCOL) = NAPTRO
              KINBAS(NCOL) = 0
C     ADD IN COST OF 0 AND BOUNDS OF COLUMN J
              IA(NAPTRO) = 1
              A(NAPTRO) = 0.
              NAPTRO = NAPTRO + 1
              XLB(NCOL) = XLBORG(J)
              XUB(NCOL) = XUBORG(J)
              IF (IBEG.GT.IEND) GO TO 430
              DO 420 I=IBEG,IEND
                 IF (IROWOD(I).EQ.IOBJ) GO TO 420
                 IA(NAPTRO)=IROWOD(I)+JFROW(NS)-NS
                 A(NAPTRO) = OFFD(I)
                 NAPTRO = NAPTRO + 1
420         CONTINUE
430       CONTINUE
      LA(NCOL+1) = NAPTRO
C
C  PRICE OUT EACH OFFDIAGONAL COLUMN OF SUB. NS AND STORE IN PRICE.
C
      CALL UPDATX
      CALL FORMC
      CALL BTRAN
      DO 500 J=JBEG,JEND
              DPRICE = 0.
              IBEG = LCOLOD(J)
              IEND = LCOLOD(J+1) - 1
              IF (IBEG .GT. IEND) GO TO 470
              DO 450 I=IBEG,IEND
                 IR = IROWOD(I)
                 IF (IR .EQ. IOBJ) GO TO 450
                 IR = IR + JFROW(NS) - NS
                 DE = OFFD(I)
                 DPRICE = DPRICE + (DE * Y(IR))
450         CONTINUE
C    RECORD REDUCED COSTS, RECALLING THAT OBJECTIVE VALUES WERE DOUBLED.
470       PRICE(J) = DPRICE / 2.0
500       CONTINUE
      WRITE(21,1010) NS
      WRITE(21,1020) (PRICE(J),J=JBEG,JEND)
1010  FORMAT(' PRICE OF OFFDIAGONAL COLUMNS IN SUB.',I3)
```

```
1020  FORMAT(10F7.2)
C
C  RESOLVE LP WITH OFFDIAGONAL COLUMNS ADDED ON. CALCULATE MAXC2.
C
      ITSINV = 99999
      CALL NORMAL(ITSINV)
      DOBJ = X(IOBJ) + ZTOLZE
      IF (DOBJ.GE.0.) GO TO 510
      DOBJ = X(IOBJ) - ZTOLZE
510   MAXC2(NS-1) = IDINT(DOBJ/2.0)
C
C  ADD IN OFFDIAGONAL COSTS FOR CALCULATION OF MAXC
C
      NCOL = NTEMPC
      DO 520 J=JBEG,JEND
            NCOL = NCOL + 1
520      A(LA(NCOL)) = COST(J)
C
C  RESOLVE LP WITH OFFDIAGONAL COLUMNS ADDED ON AND COSTS SET.
C
      ITSINV = 99999
      CALL NORMAL(ITSINV)
      IF (MSTAT.EQ.ON) GO TO 2000
      NS = NS - 1
      NCOL = NTEMPC
      DOBJ = X(IOBJ) + ZTOLZE
      MAXC(NS) = IDINT(DOBJ)
      IF (DOBJ. LT. 0.) MAXC(NS) = MAXC(NS) - 1
      IF (NS.NE.1) GO TO 100
C
C   MAXC AND PRICE HAVE BEEN CALCULATED. RETURN.
C
      K = NP - 1
      WRITE(21,1505)
1505  FORMAT(/'        MAXIMUM CUMULATIVE OBJECTIVE VALUES')
      WRITE(21,1510) (MAXC(I),I=1,K)
1510  FORMAT(' FOR LAMBDA = .5:',9I6)
      WRITE(21,1520) (MAXC2(I),I=1,K)
1520  FORMAT(' FOR LAMBDA =  0:',9I6)
      WRITE(21,1525)
1525  FORMAT('  ')
      RETURN
C
C   SUBPROBLEM NS IS INFEASIBLE. QUIT
C
2000  WRITE(21,2010) NS
2010  FORMAT(' SUBPROBLEM',I3,' IS INFEASIBLE')
      STOP
      END
C-----------------------------------------------------------------------
      SUBROUTINE UPDATX
C
C        UPDATE RHS USING VARIABLE BOUNDS ACCORDING TO KINBAS, THEN
C         USE BASIS INVERSE TO TRANSFORM INTO CORRECT X VECTOR.
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
```

```
C
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      CALL SHIFTR(1,3)
      DO 100 J=1,NCOL
             IF (KINBAS(J))20,40,100
20       DE = XUB(J)
             GO TO 60
40       DE = XLB(J)
60       LL = LA(J)
             KK = LA(J+1) - 1
             DO 80 I=LL,KK
                IR = IA(I)
80          Y(IR) = Y(IR) - A(I)*DE
100      CONTINUE
      CALL FTRAN(1)
      CALL SHIFTR(3,2)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE FTRAN(IPAR)
C
C        PERFORM FORWARD TRANSFORMATION ON COLUMN STORED IN VECTOR Y
C        SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C         BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C        ***DESCRIPTION OF PARAMETERS***
C        IPAR = PARAMETER INDICATING WHICH ETA-VECTORS MATRIX (ALL E OR
C               JUST U OF LU DECOMP) IS USED TO UPDATE COLUMN Y (INPUT)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      NLE = NETA
      NFE = 1
      IF (IPAR.EQ.2) NFE = NLETA + 1
      IF (NFE .GT. NLE) RETURN
      DO 1000 IK = NFE,NLE
             LL = LE(IK)
             KK = LE(IK+1) - 1
             IPIV = IE(LL)
             DY = Y(IPIV)
             DY = DY/E(LL)
             Y(IPIV) = DY
             IF (KK .LE. LL) GO TO 1000
             LL = LL + 1
             DO 500 J = LL,KK
                IR = IE(J)
                Y(IR) = Y(IR) - E(J) * DY
500         CONTINUE
```

```
1000      CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BTRAN
C
C         PERFORM BACKWARD TRANSFORMATION ON COLUMN STORED IN VECTOR Y
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IKOMP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      IF (NETA .LE. 0) RETURN
      DO 1000 I = 1,NETA
              IK = NETA - I + 1
              LL = LE(IK)
              KK = LE(IK+1) - 1
              IPIV = IE(LL)
              DP = E(LL)
              DY = Y(IPIV)
              DSUM = 0.
              IF (KK .LE. LL) GO TO 600
              LL = LL + 1
              DO 500 J = LL,KK
                 IR = IE(J)
                 DE = E(J)
                 DPROD = DE * Y(IR)
                 DSUM = DSUM + DPROD
500         CONTINUE
600       Y(IPIV) = (DY - DSUM) / DP
1000      CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE FORMC
C
C         FORM OBJECTIVE FUNCTION VECTOR; IF BASIS IS INFEASIBLE, SET
C         OBJECTIVE FUNCTION TO BE INFEASIBILITY FORM FOR PHASE I.
C         CALLED FROM SUBROUTINES NORMAL AND BOUNDR.
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLFS,NTMAY,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
```

```
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      MSTAT = QF
      Y(IOBJ) = 0.
      DO 30 I=1,NROW
             IF (I .EQ. IOBJ) GO TO 30
             ICOL = JH(I)
             IF (X(I) .LE. (XLB(ICOL) - ZTOLZE)) GO TO 10
             IF (X(I) .GE. (XUB(ICOL) + ZTOLZE)) GO TO 20
             Y(I) = 0.
             GO TO 30
10         Y(I) = 1.
             MSTAT = QI
             GO TO 30
20         Y(I) = -1.
             MSTAT = QI
30         CONTINUE
      IF (MSTAT.EQ.QF) Y(IOBJ) = 1.
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE PRICE
C
C        PRICE OUT NONBASIC COLUMNS; CHOOSE PIVOT COLUMN JCOLP FOR
C         CURRENT PRIMAL SIMPLEX ITERATION. JCOLP=0 ==> DUAL FEASIBLE.
C        SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C         BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPAR1,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      CMIN = 1.E10
      CMAX = -1.E10
      DO 1000 J=1,NCOL
             IF (KINBAS(J) .GT. 0) GO TO 1000
             IF ((XUB(J) - XLB(J)) .LT. ZTOLZE) GO TO 1000
C    CALCULATE DPRICE = PRICE OF BRINGING COLUMN J INTO THE BASIS
             DPRICE = 0.
             LL = LA(J)
             KK = LA(J+1) - 1
             DO 500 I = LL,KK
                IR = IA(I)
                DE = A(I)
500           DPRICE = DPRICE + (DE * Y(IR))
             IF (KINBAS(J) .EQ. -1) GO TO 600
             IF (DPRICE .GE. CMIN) GO TO 1000
             CMIN = DPRICE
```

```
                JCOL1 = J
                GO TO 1000
600         IF (DPRICE .LE. CMAX) GO TO 1000
                CMAX = DPRICE
                JCOL2 = J
1000        CONTINUE
C
C     CHOOSE PIVOT COLUMN JCOLP BASED ON PRICES
C
      IF (CMIN .LE. -ZTCOST) GO TO 1500
      IF (CMAX .GE. ZTCOST) GO TO 2000
      JCOLP = 0
      RETURN
1500  IF (CMAX .GE. ZTCOST) GO TO 2500
1600  JCOLP = JCOL1
      RETURN
2000  JCOLP = JCOL2
      RETURN
2500  IF (ABS(CMIN) - CMAX) 2000,2000,1600
      END
C-----------------------------------------------------------------------
      SUBROUTINE CHUZR
C
C         PERFORM MIN-RATIO TEST FOR PIVOT COLUMN JCOLP DETERMINED IN
C          SUBROUTINE PRICE, THEN SELECT PIVOT ROW IROWP FOR CURRENT
C          PRIMAL SIMPLEX ITERATION.
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIS1/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,NSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
C    VECTOR Y CONTAINS FORWARD TRANSFORM OF COLUMN JCOLP.
C    DP = MIN. RATIO SO FAR. IT IS PASSED TO UPBETA FOR THE PIVOT STEP.
C    ALSO PASSED TO UPBETA IS DE = X(IROWP) AFTER PIVOT STEP 1.
C    NPIVOT = 0 IFF JCOLP IS NONBASIC (AT OPPOSITE BOUND) AFTER PIVOT.
C
      IF (KINBAS(JCOLP) .EQ. -1) GO TO 1000
C
C    INCOMING VARIABLE AT LOWER BOUND; COMPUTE MIN RATIO DP
C
      DP = 1.E10
      DO 500 I=1,NROW
             IF (I .EQ. IOBJ) GO TO 500
             ICOL = JH(I)
             IF (Y(I) .GT. ZTOLPV) GO TO 100
             IF (Y(I) .LT. -ZTOLPV) GO TO 200
```

```
            GO TO 500
C       POSITIVE COEFFICIENT A(I,JCOLP)
100       IF (X(I) .LT. (XLB(ICOL) - ZTOLZE)) GO TO 500
            DE = (X(I) - XLB(ICOL))/Y(I)
            IF (DE .GE. DP) GO TO 500
            IPTYPE = 0
            GO TO 250
C       NEGATIVE COEFFICIENT A(I,JCOLP)
200       IF (X(I) .GT. (XUB(ICOL) + ZTOLZE)) GO TO 500
            DE = (X(I) - XUB(ICOL))/Y(I)
            IF (DE .GE. DP) GO TO 500
            IPTYPE = -1
250       DP = DE
            IROWP = I
500       CONTINUE
      DE = DP + XLB(JCOLP)
      IF (DE .LT. XUB(JCOLP)) GO TO 600
      DP = XUB(JCOLP) - XLB(JCOLP)
      NPIVOT = 0
      RETURN
600   NPIVOT = 1
      RETURN
C
C   INCOMING VARIABLE AT UPPER BOUND; COMPUTE MAX RATIO DP
C
1000  DP = -1.E10
      DO 1500 I=1,NROW
            IF (I .EQ. IOBJ) GO TO 1500
            ICOL = JH(I)
            IF (Y(I) .GT. ZTOLPV) GO TO 1100
            IF (Y(I) .LT. -ZTOLPV) GO TO 1200
            GO TO 1500
C       POSITIVE COEFFICIENT A(I,JCOLP)
1100      IF (X(I) .GT. (XUB(ICOL) + ZTOLZE)) GO TO 1500
            DE = (X(I) - XUB(ICOL))/Y(I)
            IF (DE .LE. DP) GO TO 1500
            IPTYPE = -1
            GO TO 1250
C       NEGATIVE COEFFICIENT A(I,JCOLP)
1200      IF (X(I) .LT. (XLB(ICOL) - ZTOLZE)) GO TO 1500
            DE = (X(I) - XLB(ICOL))/Y(I)
            IF (DE .LE. DP) GO TO 1500
            IPTYPE = 0
1250      DP = DE
            IROWP = I
1500      CONTINUE
      DE = DP + XUB(JCOLP)
      IF (DE .GT. XLB(JCOLP)) GO TO 1600
      DP = XLB(JCOLP) - XUB(JCOLP)
      NPIVOT = 0
      RETURN
1600  NPIVOT = 1
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE WRETA
C
C         FORM NEW ETA-VECTORS FOR PRODUCT FORM OF BASIS INVERSE
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
```

```
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      NELEM = NELEM + 1
      IE(NELEM) = IROWP
      E(NELEM) = Y(IROWP)
      DO 1000 I = 1,NROW
            IF (I .EQ. IROWP) GO TO 1000
            IF (DABS(Y(I)) .LE. ZTOLZE) GO TO 1000
            NELEM = NELEM + 1
            IE(NELEM) = I
            E(NELEM) = Y(I)
1000     CONTINUE
      NETA = NETA + 1
      LE(NETA+1) = NELEM + 1
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE SHIFTR(IOLD,INEW)
C
C        REARRANGE DATA STORAGE; USED BY SUBROUTINE INVERT
C        SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C         BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C        ***DESCRIPTION OF PARAMETERS***
C        IOLD,INEW = PARAMETERS INDEXING STORAGE LOCATIONS IN WHICH
C                    DATA IS TO BE TRANSFERRED (INPUT)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      DIMENSION BARRAY(240)
      EQUIVALENCE (BARRAY(1),B(1))
      IFO = (IOLD - 1) * NRMAX
      IFN = (INEW - 1) * NRMAX
      DO 1000 I = 1,NROW
            BARRAY(IFN + I) = BARRAY(IFO + I)
1000     CONTINUE
      RETURN
      END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE INVERT
C
C        COMPUTE INVERSE OF CURRENT BASIS BY LU DECOMPOSITION
C        SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C         BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,F-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOPJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/TIMERS/ITOT,TSTORE,TIMFLP,TIMEDR,TIMEDC,TIMINV
C
      INTEGER MREG,HREG,VREG
      DIMENSION MREG(60),HREG(60),VREG(60)
      EQUIVALENCE (MREG(1),YTEMP(1)),(HREG(1),YTEMP(31)),(VREG(1),X(1))
C
D     ITIME = IHPTIM(1)
C
C   SET PARAMETERS
C
      NETA = 0
      NLETA = 0
      NUETA = 0
      NELEM = 0
      NLELEM = 0
      NUELEM = 0
      NABOVE = 0
      LE(1) = 1
      KR1 = 0
      LR4 = NROW + 1
C
C   PUT SLACKS AND ARTIFICIALS IN PART 4 AND REST IN PART 1
C
      DO 100 I = 1,NROW
             IF (JH(I) .GT. NROW) GO TO 50
             LR4 = LR4 - 1
             MREG(LR4) = JH(I)
             VREG(LR4) = JH(I)
             GO TO 90
50        KR1 = KR1 + 1
             VREG(KR1) = JH(I)
90        HREG(I) = -1
             JH(I) = 0
100       CONTINUE
      KR3 = LR4 -1
      LR3 = LR4
      DO 200 I = LR4,NROW
             IR = MREG(I)
             HREG(IR)= 0
             JH(IR) = IR
             KINBAS(IR) = IR
```

```
200       CONTINUE
C
C    PULL OUT VECTORS BELOW BUMP AND GET ROW COUNTS
C
      NBNONZ = NROW - LR4 + 1
      IF (KR1 .EQ. 0) GO TO 1190
      J = 1
210   IV = VREG(J)
      LL = LA(IV)
      KK = LA(IV+1) -1
      IRCNT = 0
      DO 220 I = LL,KK
              NBNONZ = NBNONZ + 1
              IR = IA(I)
              IF (HREG(IR) .GE. 0) GO TO 220
              IRCNT = IRCNT + 1
              HREG(IR) = HREG(IR) - 1
              IRP = IR
220       CONTINUE
      IF (IRCNT - 1) 230,250,300
230   WRITE(21,8000)
8000  FORMAT(16HMATRIX SINGULAR )
      KINBAS(IV) = 0
      VREG(J) = VREG(KR1)
      KR1 = KR1 - 1
      IF (J .GT. KR1) GO TO 310
      GO TO 210
C
250   VREG(J) = VREG(KR1)
      KR1 = KR1 - 1
      LR3 = LR3 - 1
      VREG(LR3) = IV
      HREG(LR3) = IRP
      HREG(IRP) = 0
      JH(IRP) = IV
      KINBAS(IV) = IRP
      IF (J .GT. KR1) GO TO 310
      GO TO 210
300   IF (J .GE. KR1) GO TO 310
      J = J+1
      GO TO 210
C
C    PULL OUT REMAINING VECTORS ABOVE AND BELOW THE
C     BUMP AND ESTABLISH MERIT COUNTS OF COLUMNS
C
310   NVREM = 0
      IF(KR1 .EQ. 0) GO TO 1190
      J = 1
320   IV = VREG(J)
      LL = LA(IV)
      KK = LA(IV+1) - 1
      IRCNT = 0
      DO 800 I = LL,KK
              IR = IA(I)
              IF(HREG(IR) .NE. -2) GO TO 400
C
C    PIVOT ABOVE BUMP (PART OF L)
C
              NABOVE = NABOVE + 1
              IROWP = IR
```

```
            CALL UNPACK(IV)
            CALL WRETA
            NLETA = NETA
            JH(IR) = IV
            KINBAS(IV) = IR
            VREG(J) = VREG(KR1)
            KR1 = KR1 - 1
            NVREM = NVREM + 1
            HREG(IR) = IV
            GO TO 940
C
400      IF (HREG(IR) .GE. 0) GO TO 800
            IRCNT = IRCNT + 1
            IRP = IR
800      CONTINUE
C
      IF (IRCNT - 1) 810,900,1000
810   WRITE(21,8000)
      KINBAS(IV) = 0
      VREG(J) = VREG(KR1)
      NVREM = NVREM + 1
      KR1 = KR1 - 1
      IF (J .GT. KR1) GO TO 1010
      GO TO 320
C
C   PUT VECTOR BELOW BUMP
C
900   VREG(J) = VREG(KR1)
      NVREM = NVREM + 1
      KR1 = KR1 - 1
      LR3 = LR3 - 1
      VREG(LR3) = IV
      HREG(LR3) = IRP
      HREG(IRP) = 0
      JH(IRP) = IV
      KINBAS(IV) = IRP
C
C   CHANGE ROW COUNTS
C
940   DO 950 II = LL,KK
            IIR = IA(II)
            IF (HREG(IIR) .GE. 0) GO TO 950
            HREG(IIR) = HREG(IIR) + 1
950      CONTINUE
      IF (J .GT. KR1) GO TO 1010
      GO TO 320
1000  IF (J .GE. KR1) GO TO 1010
      J = J+1
      GO TO 320
1010  IF(NVREM .GT. 0) GO TO 310
C
C   GET MERIT COUNTS
C
1020  IF (KR1 .EQ. 0) GO TO 1190
      DO 1100 J = 1,KR1
            IV = VREG(J)
            LL = LA(IV)
            KK = LA(IV+1) - 1
            IMCNT = 0
            DO 1050 I = LL,KK
```

```
                  IR = IA(I)
                  IF (HREG(IR) .GE. 0) GO TO 1050
                  IMCNT = IMCNT - (HREG(IR) +1)
1050          CONTINUE
               MREG(J) = IMCNT
1100      CONTINUE
C
C    SORT COLUMNS INTO MERIT ORDER USING SHELL SORT
C
       ISD = 1
1106   IF (KR1 .LT. 2*ISD) GO TO 1108
       ISD = 2*ISD
       GO TO 1106
1108   ISD = ISD - 1
C    END OF INITIALIZATION
1101   IF (ISD .LE. 0) GO TO 1107
       ISK = 1
1102   ISJ = ISK
       ISL = ISK + ISD
       ISY = MREG(ISL)
       ISZ = VREG(ISL)
1103   IF (ISY .LT. MREG(ISJ)) GO TO 1104
1105   ISL = ISJ + ISD
       MREG(ISL) = ISY
       VREG(ISL) = ISZ
       ISK = ISK + 1
       IF ((ISK + ISD) .LE. KR1) GO TO 1102
       ISD = (ISD - 1) / 2
       GO TO 1101
1104   ISL = ISJ + ISD
       MREG(ISL) = MREG(ISJ)
       VREG(ISL) = VREG(ISJ)
       ISJ = ISJ - ISD
       IF (ISJ .GT. 0) GO TO 1103
       GO TO 1105
1107   CONTINUE
C
C    END OF SORT ROUTINE
C    PUT OUT BELOW BUMP ETAS (PART OF U)
C
1190   NSLCK = 0
       NBELOW = 0
       NELAST = NEMAX
       NTLAST = NTMAX
       LE(NTLAST + 1) = NELAST + 1
       LR = LR3
       IF (LR3 .GT. LR4) LR = LR4
       IF (LR .GT. NROW) GO TO 2050
       JK = NROW + 1
       DO 2000 JJ= LR,NROW
            JK = JK - 1
            IV = VREG(JK)
            I = MREG(JK)
            NBELOW = NBELOW + 1
            IF (IV .GT. NROW) GO TO 1200
            NSLCK = NSLCK + 1
1200      LL = LA(IV)
            KK = LA(IV+1) -1
            IF (KK .GT. LL) GO TO 1300
1250      IF (ABS(A(LL) - 1.) .LE. ZTOLZE) GO TO 2000
```

```
1300      NUETA = NUETA + 1
              DO 1400 J = LL,KK
                 IR = IA(J)
                 IF (IR .EQ. I)GO TO 1390
                 IE(NELAST) = IR
                 E(NELAST) = A(J)
                 NELAST = NELAST - 1
                 NUELEM = NUELEM + 1
                 GO TO 1400
1390          EP = A(J)
1400          CONTINUE
              IE(NELAST) = I
              E(NELAST) = EP
              LE(NTLAST) = NELAST
              NELAST = NELAST - 1
              NTLAST = NTLAST - 1
              NUELEM = NUELEM + 1
2000      CONTINUE
2050   IF(KR1 .EQ. 0) GO TO 3500
C
C    DO L-U DECOMPOSITION OF BUMP
C
       DO 3000 J = 1,KR1
              IV = VREG(J)
              CALL UNPACK(IV)
              CALL FTRAN(2)
              IROWP = 0
              IRCMIN = -999999
              DO 2100 I = 1,NROW
                 IF (DABS(Y(I)) .LE. ZTOLPV) GO TO 2100
                 IF (HREG(I) .GE.0) GO TO 2100
                 IF (HREG(I) .LE. IRCMIN) GO TO 2100
                 IRCMIN = HREG(I)
                 IROWP = I
2100          CONTINUE
              IF (IROWP .GT. 0) GO TO 2150
              WRITE(21,8000)
              KINBAS(IV) = 0
              GO TO 3000
2150      INCR = HREG(IROWP) + 3
C
C    WRITE L AND U ETAS
C
              IF (J .EQ. KR1) GO TO 2160
              NELEM = NELEM + 1
              IE(NELEM) = IROWP
              E(NELEM) = Y(IROWP)
2160      DO 2300 I = 1,NROW
                 IF (I .EQ. IROWP) GO TO 2300
                 IF(DABS(Y(I)) .LE. ZTOLZE) GO TO 2300
                 IF (HREG(I) .GE. 0) GO TO 2200
C
C    L ETA ELEMENTS
C
                 NELEM = NELEM + 1
                 IE(NELEM) = I
                 E(NELEM) = Y(I)
                 GO TO 2300
C
C    U ETA ELEMENTS
```

```
C
2200         IE(NELAST) = I
                 E(NELAST) = Y(I)
                 NELAST = NELAST - 1
                 NUELEM = NUELEM + 1
2300         CONTINUE
C
              JH(IROWP) = IV
              KINBAS(IV) = IROWP
              NUETA = NUETA + 1
              IE(NELAST) = IROWP
              IF (J .NE. KR1) GO TO 2330
              E(NELAST) = Y(IROWP)
              GO TO 2340
2330      E(NELAST) = 1.
              NETA = NETA + 1
              LE(NETA+1) = NELEM + 1
2340      NUELEM = NUELEM + 1
              LE(NTLAST) = NELAST
              NELAST = NELAST - 1
              NTLAST = NTLAST - 1
C
C   UPDATE ROW COUNTS
C
              DO 2350 I = 1,NROW
                 IF (DABS(Y(I)) .LE. ZTOLZE) GO TO 2350
                 IF (HREG(I) .GE. 0) GO TO 2350
                 HREG(I) = HREG(I) - INCR
                 IF (HREG(I) .GE. 0) HREG(I) = -1
2350         CONTINUE
              HREG(IROWP) = 0
3000      CONTINUE
C
C   MERGE L AND U ETAS
C
3500   NLETA = NETA
       NETA = NLETA + NUETA
       NLELEM = NELEM
       NELEM = NLELEM + NUELEM
       IF (NUELEM .EQ. 0) GO TO 3550
       CALL SHFTE
C
C   INSERT SLACKS FOR DELETED COLUMNS
C
3550   DO 3600 I = 1,NROW
              IF (JH(I) .NE. 0) GO TO 3600
              JH(I) = I
              IROWP = I
              CALL UNPACK(IROWP)
              CALL FTRAN(1)
              CALL WRETA
3600      CONTINUE
C
C   UPDATE X
C
       CALL UPDATX
D      ITIME2 = IHPTIM(1)
D      TIMINV = TIMINV + (ITIME2-ITIME)/100000.
       RETURN
       END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE UNPACK(IV)
C
C         EXPAND COMPRESSED MATRIX COLUMN AND STORE IN VECTOR Y
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C         ***DESCRIPTION OF PARAMETERS***
C         IV = PARAMETER INDEXING COLUMN TO BE EXPANDED (INPUT)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      DO 100 I = 1,NROW
            Y(I) = 0.
100       CONTINUE
      LL = LA(IV)
      KK = LA(IV+1) - 1
      DO 200 I = LL,KK
            IR = IA(I)
            Y(IR) = A(I)
200       CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE SHFTE
C
C         SUBROUTINE FOR INVERT
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
C   SHIFT IE AND E OF U ELEMENTS
C
      NF = NEMAX - NUELEM + 1
      INCR = 0
      DO 1000 I = NF,NEMAX
            INCR = INCR + 1
            IE(NLELEM + INCR) = IE(I)
            E(NLELEM + INCR) = E(I)
1000      CONTINUE
```

```
C
      IDIF = NEMAX - NLELEM - NUELEM
      NF = NTMAX - NUETA + 1
      INCR = 0
      DO 2000 I = NF,NTMAX
              INCR = INCR + 1
              LE(NLETA + INCR) = LE(I) - IDIF
2000      CONTINUE
      LE(NETA+1) = NELEM + 1
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE UPBETA
C
C         UPDATE RIGHT-HAND SIDES TO REFLECT NEW BASIS RESULTING FROM
C          CURRENT SIMPLEX PIVOT
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      DO 1000 I=1,NROW
1000      X(I) = X(I) - Y(I)*DP
      IF (NPIVOT .EQ. 1) GO TO 2000
      KINBAS(JCOLP) = -(KINBAS(JCOLP) + 1)
      RETURN
2000  A(IROWP) = DE
      IVOUT = JH(IROWP)
      KINBAS(JCOLP) = IROWP
      KINBAS(IVOUT) = IPTYPE
      JH(IROWP) = JCOLP
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE NORMAL(ITSINV)
C
C         THIS IS THE MASTER PROGRAM FOR LINEAR PROGRAMMING COMPONENT
C          (REVISED,PRIMAL-SIMPLEX METHOD) OF BRANCH-AND-BOUND ROUTINE.
C         SUBROUTINE ADAPTED FROM LINEAR PROGRAMMING CODE LPM-1, WRITTEN
C          BY J. A. TOMLIN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C         ***DESCRIPTION OF PARAMETERS***
C         ITSINV = NUMBER OF SIMPLEX ITERATIONS SINCE LAST BASIS
C                  INVERSION (INPUT/OUTPUT)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
```

```
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
C
      IF (ITSINV .LT. INVFRQ) GO TO 1500
1000  CALL INVERT
      ITSINV = 0
C
C   SIMPLEX CYCLE
C
1500  CALL FORMC
      CALL BTRAN
      CALL PRICE
      IF (JCOLP .GT. 0) GO TO 3000
      IF (MSTAT .EQ. QI) GO TO 2000
      MSTAT = QBL
      RETURN
2000  MSTAT = QN
      RETURN
C
C   PIVOT ON COLUMN JCOLP.
C
3000  CALL UNPACK(JCOLP)
      CALL FTRAN(1)
      CALL CHUZR
      CALL UPBETA
      ITCNT = ITCNT + 1
      ITSINV = ITSINV + 1
      IF (NPIVOT .EQ. 0) GO TO 4010
      IF (NELEM .GT. (NEMAX-NROW)) GO TO 1000
      CALL WRETA
4010  IF (ITSINV .GE. INVFRQ) GO TO 1000
      IF (ITCNT .GE. ITRFRQ) RETURN
      GO TO 1500
      END
C-----------------------------------------------------------------------
      SUBROUTINE BANDB(INITBD)
C
C         MASTER PROGRAM FOR BRANCH-AND-BOUND INTEGER PROGRAMMING
C          ROUTINE.  ALSO SERVES AS MASTER PROGRAM FOR REOPTIMIZATION
C          VIA REVISED DUAL-SIMPLEX METHOD AFTER A FORWARD BRANCH.
C         SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C          BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C         ***DESCRIPTION OF PARAMETERS***
C         INITBD = INITIAL LOWER BOUND ON MAX. OBJECTIVE VALUE (INPUT)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
```

```
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QP,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
C
      IFEAS = 0
      LISTL = 0
      ISUMC = 0
      INCVAL = INITBD
C
C   TEST FOR FATHOMING
C
      ENTRY BBENTR
100   CALL TESTX
      IF (MSTAT .EQ. QBL) GO TO 200
      IF (MSTAT .EQ. QE) RETURN
C   CURRENT NODE FATHOMED;BACKTRACK TO LAST PROMISING NODE ON LIST
150   CALL BKTRAK
C   IF LIST IS EMPTY, RETURN TO MAIN (COMPUTATIONS COMPLETED)
      IF (LISTL .EQ. 0) RETURN
C   USE PRIMAL SIMPLEX METHOD FOR REOPTIMIZATION AT NEW NODE
      CALL NORMAL(ITSINV)
      IF (ITCNT .GE. ITRFRQ) GO TO 2000
      GO TO 100
C   CURRENT NODE NOT FATHOMED; COMPUTE PENALTIES
C    BRANCHING AT CURRENT NODE IS DONE FROM SUBROUTINE PENLTS
200   CALL PENLTS
      IF (IDIR) 400,150,400
C
C   REINVERT CURRENT BASIS
1000  CALL INVERT
      ITSINV = 0
C
C   DUAL SIMPLEX CYCLE
C
C   CHOOSE PIVOT ROW IROWP
300   CALL DCHUZR
      IF (IROWP .GT. 0) GO TO 400
      MSTAT = QPL
      GO TO 100
C   CHOOSE PIVOT COLUMN JCOLP
400   CALL DCHUZC
      IF (JCOLP .EQ. 0) GO TO 150
C   UPDATE RIGHT-HAND SIDES TO REFLECT NEW BASIS RESULTING FROM
C    CURRENT SIMPLEX PIVOT
      CALL UPBETA
      ITCNT = ITCNT + 1
      IF (ITCNT .GE. ITRFRQ) GO TO 2000
      ITSINV = ITSINV + 1
      IF (NELEM.GT.(NEMAX-NROW) .OR. (ITSINV.GE.INVFRQ)) GO TO 1000
C   WRITE OUT NEW ETA-VECTOR FOR CURRENT SIMPLEX PIVOT
      CALL WRETA
      GO TO 300
```

```
C
2000  LISTL = 0
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE DCHUZR
C
C         SELECT PIVOT ROW IROWP FOR CURRENT DUAL-SIMPLEX ITERATION.
C          SET IROWP=0 IF CURRENT BASIS IS OPTIMAL.  OTHERWISE, CHOOSE
C          IROWP TO BE THE ROW WITH GREATEST PRIMAL INFEASIBILITY.
C         SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C          BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,KELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/TIMERS/ITOT,TSTORE,TIMELP,TIMEDR,TIMEDC,TIMINV
C
D     ITIME = IHPTIM(1)
      IROWP = 0
      DP = -1.E10
      DO 1000 I=1,NROW
            IF (I .EQ. IOBJ) GO TO 1000
            ICOL = JH(I)
            IF (X(I) .LT. (XLB(ICOL) - ZTOLZE)) GO TO 100
            IF (X(I) .GT. (XUB(ICOL) + ZTOLZE)) GO TO 200
            GO TO 1000
C
C      BASIC VARIABLE ON ROW I FALLS BELOW ITS LOWER BOUND
100     DE = XLB(ICOL) - X(I)
            IF (DE .LE. DP) GO TO 1000
            IPTYPE = 0
            GO TO 250
C
C      BASIC VARIABLE ON ROW I EXCEEDS ITS UPPER BOUND
200     DE = X(I) - XUB(ICOL)
            IF (DE .LE. DP) GO TO 1000
            IPTYPE = -1
C
250     IROWP = I
            DP = DE
1000    CONTINUE
D     ITIME2 = IHPTIM(1)
D     TIMEDR = TIMEDR + (ITIME2-ITIME)/100000.
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE DCHUZC
```

```
C
C        SELECT PIVOT COLUMN JCOLP FOR CURRENT DUAL-SIMPLEX ITERATION.
C         SET JCOLP = 0 IF LP-PROBLEM AT CURRENT NODE IS INFEASIBLE,
C         OTHERWISE CHOOSE JCOLP TO MAINTAIN PRIMAL-OPTIMALITY.
C        SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C         BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/TIMERS/ITOT,TSTORE,TIMELP,TIMEDR,TIMEDC,TIMINV
C
D     ITIME = IHPTIM(1)
      JCOLP = 0
      IF (IPTYPE .EQ. -1) GO TO 1000
C
C   LEAVING VARIABLE FALLS BELOW ITS LOWER BOUND; COMPUTE MAX RATIO DP
C
      DP = -1.E10
      DO 500 J=1,NCOL
             IF (KINBAS(J) .GT. 0) GO TO 500
             IF ((XUB(J) - XLB(J)) .LE. ZTOLZE) GO TO 500
             K = J
             CALL UNPACK(K)
             CALL FTRAN(1)
             IF (KINBAS(J) .EQ. -1) GO TO 200
             IF (Y(IROWP) + ZTOLPV) 225,225,500
200       IF (Y(IROWP) - ZTOLPV) 500,225,225
225       DE = Y(IOBJ)/Y(IROWP)
             IF (DE .LE. DP) GO TO 500
             JCOLP = J
             DP = DE
500       CONTINUE
C
C   STORE PIVOT COL JCOLP IN Y; STORE CHANGE IN INCOMING VAR. ICOL IN DP
      IF (JCOLP .EQ. 0) RETURN
      CALL UNPACK(JCOLP)
      CALL FTRAN(1)
      ICOL = JH(IROWP)
      DP = (X(IROWP) - XLB(ICOL))/Y(IROWP)
      GO TO 2000
C
C   LEAVING VARIABLE EXCEEDS ITS UPPER BOUND; COMPUTE MIN RATIO DP
C
1000  DP = 1.E10
      DO 1500 J=1,NCOL
             IF (KINBAS(J) .GT. 0) GO TO 1500
             IF ((XUB(J) - XLB(J)) .LE. ZTOLZE) GO TO 1500
```

```
            K = J
            CALL UNPACK(K)
            CALL FTRAN(1)
            IF (KINBAS(J) .EQ. -1) GO TO 1200
            IF (Y(IROWP) - ZTOLPV) 1500,1225,1225
1200      IF (Y(IROWP) + ZTOLPV) 1225,1225,1500
1225      DE = Y(IOBJ)/Y(IROWP)
            IF (DE .GE. DP) GO TO 1500
            JCOLP = J
            DP = DE
1500      CONTINUE
C
C   STORE PIVOT COL JCOLP IN Y; STORE CHANGE IN INCOMING VAR. ICOL IN DP
      IF (JCOLP .EQ. 0) RETURN
      CALL UNPACK(JCOLP)
      CALL FTRAN(1)
      ICOL = JH(IROWP)
      DP = (X(IROWP) - XUB(ICOL))/Y(IROWP)
C
2000  IF (KINBAS(JCOLP) .EQ. 0) DE = DP + XLB(JCOLP)
      IF (KINBAS(JCOLP) .EQ. -1) DE = DP + XUB(JCOLP)
      NPIVOT = 1
D     ITIME2 = IMPTIM(1)
D     TIMEDC = TIMEDC + (ITIME2-ITIME)/100000.
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE TESTX
C
C         TEST LP-OPTIMAL SOLUTION AT CURRENT NODE FOR FATHOMING.
C         FATHOMING OCCURS IF:
C          (1) LP PROBLEM AT CURRENT NODE IS INFEASIBLE (MSTAT = QN);
C          (2) LP-OPT OBJ. VALUE + OBJ. VALUE FOR PREVIOUS PERIODS +
C               OBJ. BOUND ON SUCCEEDING PERIODS <= OBJ. OF INCUMBENT
C          (3) LP-OPT SOL. SATISFIES INTEGER RESTRICTIONS AND NS = NP
C         IF THE LP-OPT. SOL. IS INTEGER BUT FATHOMING DOES NOT OCCUR,
C          BRANCH ON (FIX) ALL NONSLACK VARS AND STORE SUBPROBLEM NS IN
C          PREPARATION FOR A FORWARD STEP. SET MSTAT = QE TO FLAG THIS.
C         SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C          BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
      COMMON/TIMERS/ITOT,TSTORE,TIMELP,TIMEDR,TIMEDC,TIMINV
C
```

```
C    INCUMB = BEST SOLUTION FOUND SO FAR; INCVAL IS ITS OBJECTIVE VALUE
C    ICURX CONTAINS CURRENT VALUES FOR VARIABLES IN SUBS. 1,...,NS-1
C    DFPART(I) [IPART(I)] IS THE FRACTIONAL [INTEGRAL] PART OF X(I)
C    IVAL = LP-OPT. OBJECTIVE VALUE FOR SUBPROBLEM NS ON CURRENT BRANCH
C    JBOUND = BOUND ON MAX. OBJECTIVE VALUE (USING MAXC)
C    ISUMC = OBJECTIVE VALUE FOR SUBPROBLEMS 1,...,NS-1 ON THIS BRANCH
C    IFFAS = 1 IFF AN INTEGRAL SOL. WITH INCVAL > INITBD HAS BEEN FOUND
C
C    TEST FOR FATHOMING IN WAYS (1) AND (2)
C
      IF (MSTAT .EQ. ON) GO TO 2000
      DP = X(IOBJ) + ZTOLZE
      IVAL = IDINT(DP)
      IF (DP .LT. 0.) IVAL = IVAL - 1
      IF ((MAXC2(NS)+IVAL+ISUMC) .LE. INCVAL) GO TO 2000
      IF (NS.EQ.NP) GO TO 50
      JBOUND = IVAL + MAXC(NS)
      IF (JBOUND.LT.0) JBOUND = JBOUND - 1
      JBOUND = JBOUND / 2
      IF ((JBOUND + ISUMC) .LE. INCVAL) GO TO 2000
C
C    COMPUTE INTEGER AND FRACTIONAL PARTS OF EACH BASIC VAR.
C
50    DO 100 I=1,NROW
            IPART(I) = IDINT(X(I) + ZTOLZE)
            IF (X(I) .LT. -ZTOLZE) IPART(I) = IPART(I) - 1
            NTEMP1 = IPART(I)
100      DFPART(I) = X(I) - FLOAT(NTEMP1)
C
C    CHECK FOR ALL-INTEGER SOLUTION
C
      DO 200 I=1,NROW
            IF (JH(I) .LE. NROW) GO TO 200
            IF (DFPART(I) .GE. ZTOLZE) RETURN
200      CONTINUE
C
C    SOLUTION ALL-INTEGER: CHECK FOR COMPLETE SOLUTION
C
      IF (NS.LT.NP) GO TO 400
C
C    NEW IMPROVED INTEGER SOLUTION TO ALL PERIODS REACHED.
C     OUTPUT OBJ. VAL. AND  COMPUTATION TIME REQUIRED TO REACH IT.
C
      INCVAL = IVAL + ISUMC
      JTIME = IMPTIM(1)
      TOPT = (JTIME-ITOT)/100000.
      IF (IFEAS.EQ.0) WRITE(21,1)
1     FORMAT ('       INTERMEDIATE SOLUTIONS FOUND')
      WRITE (21,2) TOPT,NBRANC,INCVAL
2     FORMAT (' TIME =',F7.2,' SECONDS;  BRANCHES =',I10,';  INCVAL =',
     1  I10)
      IFEAS = 1
C    STORE NEW INCUMBENT SOLUTION
      K = JFCOL(NS) - 1
      DO 300 J=1,K
300      INCUMB(J) = ICURX(J)
      DO 350 J=1,NCOL
            IF (FINBAS(J)) 320,330,350
320      INCUMB(K+J) = IDINT(XUB(J))
            GO TO 350
```

```
330       INCUMB(K+J) = IDINT(XLB(J))
350       CONTINUE
      DO 360 I=1,NROW
            ICOL = JH(I)
360       INCUMB(K+ICOL) = IPART(I)
      GO TO 2000
C
C    A PARTIAL INTEGER SOLUTION HAS BEEN REACHED. BRANCH ON ALL
C     NONSLACK, UNFIXED VARIABLES AND SAVE SOLUTION IN ICURX.
C
400    K = JFCOL(NS) - 1
C    BRANCH ON ALL NONBASIC NONSLACK VARIABLES FIRST
      DO 500 J=1,NCOL
            IF (KINBAS(J)) 420,440,500
C       VARIABLE NONBASIC AT UPPER BOUND; BRANCH UP
420       ICURX(K+J) = IDINT(XUB(J))
            IF (J.LE.NROW .OR. (XUB(J)-XLB(J)).LE.ZTOLZE) GO TO 500
            IDIR = -1
            REVBND = SNGL(XUB(J))
            GO TO 460
C       VARIABLE NONBASIC AT LOWER BOUND; BRANCH DOWN
440       ICURX(K+J) = IDINT(XLB(J))
            IF (J.LE.NROW .OR. (XUB(J)-XLB(J)).LE.ZTOLZE) GO TO 500
            IDIR = 1
            REVBND = SNGL(XLB(J))
460       ICOL = J
            CALL BRANCH
500       CONTINUE
C
C    STORE AND BRANCH ON ALL NONSLACK BASIC VARIABLES
      DO 600 I=1,NROW
            ICOL = JH(I)
            ICURX(K+ICOL) = IPART(I)
            IF (ICOL.LE.NROW) GO TO 600
            IF ((XUB(ICOL)-XLB(ICOL)).LE.ZTOLZE) GO TO 600
            IF ((X(I)-XLB(ICOL)).LE.ZTOLZE) GO TO 520
            IF ((XUB(ICOL)-X(I)).GT.ZTOLZE) GO TO 550
C       VARIABLE BASIC AT UPPER BOUND; BRANCH UP
            IDIR = -1
            REVBND = SNGL(XUB(ICOL))
            GO TO 590
C       VARIABLE BASIC AT LOWER BOUND; BRANCH DOWN
520       IDIR = 1
            REVBND = SNGL(XLB(ICOL))
            GO TO 590
C       VARIABLE BASIC BETWEEN BOUNDS; BRANCH TO FIX IT
550       IDIR = -1
            REVBND = FLOAT(IPART(I))
            CALL BRANCH
            IDIR = 1
590       CALL BRANCH
600       CONTINUE
C    STORE CURRENT SUBPROBLEM AND THE CURRENT OBJ. VALUE
      CALL STORE
      ISUMC = ISUMC + IVAL
      MSTAT = QE
      RETURN
C
C    CURRENT PROBLEM NO LONGER OF INTEREST
C
```

```
2000  MSTAT = QI
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE PENLTS
C
C        COMPUTE TOMLIN'S IMPROVED UP- AND DOWN- PENALTIES AND THE
C         GOMORY PENALTY FOR EACH NONINTEGER BASIC VARIABLE. THEN CHECK
C         FOR FORCED BRANCHES ON BOTH BASIC AND NONBASIC VARIABLES. IN
C         THE ABSENCE OF FORCED BRANCHES ON BASIC VARIABLES, ADD TO
C         EACH PENALTY THE PRICE OF THE CORRESPONDING OFFDIAGONAL COLUMN
C         OF THE NEXT SUBPROBLEM (NS + 1). THEN CHOOSE AS BRANCHING
C         VARIABLE THE ONE WITH LARGEST ASSOCIATED UP- OR DOWN-PENALTY.
C         TAKE THE FORWARD BRANCH IN THE DIRECTION OPPOSITE TO THIS
C         MAXIMUM PENALTY WHILE ADDING THAT VARIABLE TO THE LIST WITH
C         THE APPROPRIATE BRANCH DIRECTION (IVID) AND BOUND (IOBND).
C         THE BRANCHING PROCESS ITSELF IS DONE IN SUBROUTINE BRANCH.
C        SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C         BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
      REAL PU(60),PD(60),PG(60)
      LOGICAL FORCED
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  QLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/BESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
C
C    PU(I),PD(I),PG(I) ARE THE UP,DOWN, AND GOMORY PENALTIES FOR VAR JH(I)
C    REVBND = REVISED BOUND ON THE BRANCH VAR ICOL; IT IS PASSED TO BRANCH
C    IF IDIR = -1 BRANCH UP, +1 BRANCH DOWN, 0 FATHOMING HAS OCCURRED
C    IVAL = LP-OPT. OBJECTIVE VALUE FOR SUBPROBLEM NS ON CURRENT BRANCH
C    JBOUND = BOUND ON MAX. OBJECTIVE VALUE FOR SUBPROBLEMS NS,...,NP
C    ISUMC = OBJECTIVE VALUE FOR SUBPROBLEMS 1,...,NS-1 ON THIS BRANCH
C    VECTOR Y CONTAINS APPROPRIATE COLUMN OF CURRENT SIMPLEX TABLEAU
C
      DO 10 I=1,NROW
            IF (DFPART(I) .LT. ZTOLZE) GO TO 5
            PU(I) = 1.E6
            PD(I) = 1.E6
            PG(I) = 1.E6
            GO TO 10
5        PU(I) = 0.
            PD(I) = 0.
            PG(I) = 0.
10       CONTINUE
C
C    MAIN LOOP: CALCULATE PENALTIES FOR EACH ELIGIBLE INCOMING VAR. J
```

```
C
      DO 1000 J=1,NCOL
            IF (KINBAS(J) .GT. 0) GO TO 1000
            IF ((XUB(J) - XLB(J)) .LE. ZTOLZE) GO TO 1000
            K = J
            CALL UNPACK(K)
            CALL FTRAN(1)
            IF (KINBAS(J) .EQ. 0) GO TO 30
            DO 20 I=1,NROW
20          Y(I) = -Y(I)
C
C      CHECK FOR FORCED BRANCH ON NONBASIC VARIABLE J
C
30        IF (J .LE. NROW) GO TO 60
            DP = X(IOBJ) - Y(IOBJ) + ZTOLZE
            IVAL = IDINT(DP)
            IF (DP .LT. 0.) IVAL = IVAL - 1
            IF ((MAXC2(NS)+IVAL+ISUMC) .LE. INCVAL) GO TO 50
            IF (NS.EQ.NP) GO TO 60
            JBOUND = IVAL + MAXC(NS)
            IF (JBOUND.LT.0) JBOUND = JBOUND - 1
            JBOUND = JBOUND / 2
            IF ((JBOUND + ISUMC) .GT. INCVAL) GO TO 60
50        IDIR = 2*KINBAS(J) + 1
            IF (IDIR .EQ. -1) REVBND = SNGL(XUB(J))
            IF (IDIR .EQ. 1) REVBND = SNGL(XLB(J))
            ICOL = J
            CALL BRANCH
            GO TO 1000
C
C      COMPUTE PENALTIES ON BASIC VARIABLES JH(I), FOR ICOL = J
C
60        DO 500 I=1,NROW
                IF (JH(I) .LE. NROW) GO TO 500
                IF (DFPART(I) .LT. ZTOLZE) GO TO 500
C         COMPUTE UP PENALTY FOR JH(I)
100          IF (Y(I) .GT. -ZTOLPV) GO TO 200
                DE = Y(IOBJ)*(DFPART(I) - 1.)/Y(I)
                IF (DE .LT. Y(IOBJ)) DE = Y(IOBJ)
                IF (DE .LT. PU(I)) PU(I) = DE
                GO TO 300
C         COMPUTE DOWN PENALTY FOR JH(I)
200          IF (Y(I) .LT. ZTOLPV) GO TO 300
                DE = Y(IOBJ)*DFPART(I)/Y(I)
                IF (DE .LT. Y(IOBJ)) DE = Y(IOBJ)
                IF (DE .LT. PD(I)) PD(I) = DE
C         COMPUTE GOMORY PENALTY FOR JH(I)
300          DP = DABS(Y(I))
                IF (DP .LE. ZTOLZE) GO TO 500
                NTEMP1 = IDINT(DP)
                DP = DP - FLOAT(NTEMP1)
                IF ((DP .GT. ZTOLZE) .AND. (DP .LT. 1.-ZTOLZE)) GO TO 330
                IF (J.NE.I) GO TO 500
                IF (Y(I) .LT. 0.) GO TO 320
                DE = Y(IOBJ)*DFPART(I) / Y(I)
                GO TO 350
320          DE = Y(IOBJ)*(1. - DFPART(I)) / (-Y(I))
                GO TO 350
330          IF (Y(I) .LT. 0.) DP = 1. - DP
                IF (DP .GT. DFPART(I)) GO TO 340
```

```
                DE = Y(IOBJ)*DFPART(I)/DP
                GO TO 350
340           DE = Y(IOBJ)*(1. - DFPART(I))/(1. - DP)
350           IF (DE .LT. PG(I)) PG(I) = DE
500           CONTINUE
1000      CONTINUE
C
C    COMPUTE LARGEST GOMORY PENALTY AND TEST FOR FATHOMING
C
       PEN = 0.
       DO 2000 I=1,NROW
              IF (JH(I) .LE. NROW) GO TO 2000
              IF (PG(I) .GT. PEN) PEN = PG(I)
2000      CONTINUE
       DP = X(IOBJ) - PEN + ZTOLZE
       IVAL = IDINT(DP)
       IF (DP .LT. 0.) IVAL = IVAL - 1
       IF ((MAXC2(NS)+IVAL+ISUMC) .LE. INCVAL) GO TO 2050
       IF (NS.EQ.NP) GO TO 3000
       JBOUND = IVAL + MAXC(NS)
       IF (JBOUND.LT.0) JBOUND = JBOUND - 1
       JBOUND = JBOUND / 2
       IF ((JBOUND + ISUMC) .GT. INCVAL) GO TO 3000
2050   IDIR = 0
       RETURN
C
C    PROBLEM NOT FATHOMED: CHECK FOR FORCED BRANCHES ON BASIC X(I)
C
3000   FORCED = .FALSE.
       DO 3900 I=1,NROW
              IF (JH(I).LE.NROW .OR. DFPART(I).LE.ZTOLZE) GO TO 3900
              IF (PU(I) .GT. PD(I)) GO TO 3600
              DP = X(IOBJ) - PD(I) + ZTOLZE
              NTEMP1 = IDINT(DP)
              IF (DP .LT. 0.) NTEMP1 = NTEMP1 - 1
              IF ((MAXC2(NS)+NTEMP1+ISUMC) .LE. INCVAL) GO TO 3050
              IF (NS.EQ.NP) GO TO 3900
              JBOUND = NTEMP1 + MAXC(NS)
              IF (JBOUND.LT.0) JBOUND = JBOUND - 1
              JBOUND = JBOUND / 2
              IF ((JBOUND + ISUMC) .GT. INCVAL) GO TO 3900
C        FORCED BRANCH UP ON X(I)
3050         IVAL = NTEMP1
              IDIR = -1
              NTEMP1 = IPART(I) + 1
              GO TO 3700
C
3600         DP = X(IOBJ) - PU(I) + ZTOLZE
              NTEMP1 = IDINT(DP)
              IF (DP .LT. 0.) NTEMP1 = NTEMP1 - 1
              IF ((MAXC2(NS)+NTEMP1+ISUMC) .LE. INCVAL) GO TO 3650
              IF (NS.EQ.NP) GO TO 3900
              JBOUND = NTEMP1 + MAXC(NS)
              IF (JBOUND.LT.0) JBOUND = JBOUND - 1
              JBOUND = JBOUND / 2
              IF ((JBOUND + ISUMC) .GT. INCVAL) GO TO 3900
C        FORCED BRANCH DOWN ON X(I)
3650         IVAL = NTEMP1
              IDIR = 1
              NTEMP1 = IPART(I)
```

```
3700      IROWP = I
              ICOL = JH(IROWP)
              REVBND = FLOAT(NTEMP1)
              FORCED = .TRUE.
              CALL BRANCH
3900      CONTINUE
      IF (FORCED) GO TO 5000
C
C   NO FORCED BRANCHES: CHOOSE BRANCHING VAR. AND DIRECTION
C
      PEN = 0.
      IROWP = 0
C   DETERMINE BASIC VARIABLE JH(IROWP) WITH MAX. UP- OR DOWN-PENALTY,
C    ADDING IN A PENALTY FROM THE NEXT SUBPROBLEM (STORED IN PRICE)
      DO 4900 I=1,NROW
              IF (JH(I).LE.NROW .OR. DFPART(I).LE.ZTOLZE) GO TO 4900
              UPPEN = PU(I) +  PRICE(JFCOL(NS) + JH(I) - 1)
              IF (UPPEN .GT. PD(I)) GO TO 4600
              IF (PD(I) .LE. PEN) GO TO 4900
              PEN = PD(I)
              IROWP = I
              IDIR = -1
              NTEMP1 = IPART(I) + 1
              REVBND = FLOAT(NTEMP1)
              GO TO 4900
4600      IF (UPPEN .LE. PEN) GO TO 4900
              PEN = UPPEN
              IROWP = I
              IDIR = 1
              NTEMP1 = IPART(I)
              REVBND = FLOAT(NTEMP1)
4900      CONTINUE
      IF (IROWP .GT. 0) GO TO 4950
C   EACH UP- AND DOWN-PENALTY = 0. (DUAL-DEGENERACY)  CHOOSE ANY
C    NONINTEGER BASIC VARIABLE AS BRANCHING VARIABLE ICOL
      DO 4910 IROWP=1,NROW
              IF (JH(IROWP) .LE. NROW) GO TO 4910
              IF (DFPART(IROWP) .GE. ZTOLZE) GO TO 4920
4910      CONTINUE
4920  IDIR = 1
      NTEMP1 = IPART(IROWP)
      REVBND = FLOAT(NTEMP1)
4950  IF (IDIR .EQ. 1) PEN = PU(IROWP)
      ICOL = JH(IROWP)
      DP = X(IOBJ) - PEN + ZTOLZE
      NTEMP1 = IDINT(DP)
      IF (DP .LT. 0.) NTEMP1 = NTEMP1 - 1
      IF (IVAL .GT. NTEMP1) IVAL = NTEMP1
C   BRANCH ON CHOSEN VARIABLE
      CALL BRANCH
5000  IF (IDIR .EQ. -1) IPTYPE = 0
      IF (IDIR .EQ. 1) IPTYPE = -1
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BRANCH
C
C         BRANCH ON VARIABLE ICOL AS DETERMINED IN SUBROUTINE PENLTS
C         SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C          BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
```

```
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
C
C    ICOL INDEXES BRANCHING VARIABLE CHOSEN
C    IDIR = -1 MEANS BRANCH UP, = +1 MEANS BRANCH DOWN
C    IVID STORES BRANCH VARIABLE, OPPOSITE DIRECTION, AND SUBPROBLEM #
C    IOBND STORES AN OBJECTIVE FUNCTION BOUND ON THE OPPOSITE BRANCH
C    IVBND STORES VARIABLE BOUND XUB OR XLB FOR OTHER BRANCH DIRECTION
C
C    ADD OPPOSITE DIRECTION TO LIST
      NBRANC = NBRANC + 1
      LISTL = LISTL + 1
      IF (IDIR .EQ. -1) IVBND(LISTL) = IDINT(XLB(ICOL) + ZTOLZE)
      IF (IDIR .EQ. 1) IVBND(LISTL) = IDINT(XUB(ICOL) + ZTOLZE)
      IVID(LISTL) = IDIR * (ICOL + (NS * 1000))
      IOBND(LISTL) = IVAL
C
C    REVISE BOUNDS ON BRANCHING VARIABLE FOR FORWARD DIRECTION
      IF (IDIR .EQ. -1) XLB(ICOL) = DBLE(REVBND)
      IF (IDIR .EQ. 1) XUB(ICOL) = DBLE(REVBND)
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BKTRAK
C
C        BACKTRACK TO A PROMISING (UNFATHOMED) NODE FROM THE LIST OF
C         STORED NODES. EMPLOYS LAST-IN-FIRST-OUT (LIFO) SELECTION RULE
C        SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C          BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,R-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION E(2000)
      REAL A(1000)
C
      COMMON/CONSTS/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLSM,NEGINF,NEMAX,NRMAX,QBL,
     1  NLES,NTMAX,QA,QI,QF,QN,QSUB,QB,QC,QE,QH,QL,QO,QR,QM,QG
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
```

```
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
C
      NTEMP3 = 0
C   IF LIST IS EMPTY, RETURN (COMPUTATIONS COMPLETED)
50    IF (LISTL .EQ. 0) RETURN
C   GET NEXT NODE FROM LIST. CHECK ITS SUBPROBLEM NUMBER.
      ICOL = IVID(LISTL)
      ISUBNO = IABS(ICOL)/1000
      IF (ISUBNO.EQ.NS) GO TO 70
C   WE ARE BACKTRACKING TO A PREVIOUS SUBPROBLEM
      NS = ISUBNO
      CALL RESTOR(0)
70    IF ((MAXC2(NS)+IOBND(LISTL)+ISUMC) .LE. INCVAL) GO TO 2000
      IF (NS.EQ.NP) GO TO 80
      JBOUND = IOBND(LISTL) + MAXC(NS)
      IF (JBOUND.LT.0) JBOUND = JBOUND - 1
      JBOUND = JBOUND / 2
      IF ((JBOUND + ISUMC) .LE. INCVAL) GO TO 2000
80    IF (ICOL .LT. 0) GO TO 100
      ICOL = ICOL - (NS * 1000)
C
C   BRANCH DIRECTION WAS DOWN. RESTORE UP DIRECTION BOUNDS.
      NTEMP1 = IDINT(XLB(ICOL) + ZTOLZE)
      NTEMP2 = IVBND(LISTL)
      XLB(ICOL) = XUB(ICOL) + 1.
      XUB(ICOL) = FLOAT(NTEMP2)
      IF (KINBAS(ICOL) .GT. 0) GO TO 1000
      KINBAS(ICOL) = 0
      NTEMP3 = 1
      GO TO 1000
C
C   BRANCH DIRECTION WAS UP. RESTORE LOWER DIRECTION BOUNDS.
100   ICOL = - (ICOL + (NS * 1000))
      NTEMP1 = IDINT(XUB(ICOL) + ZTOLZE)
      NTEMP2 = IVBND(LISTL)
      XUB(ICOL) = XLB(ICOL) - 1.
      XLB(ICOL) = FLOAT(NTEMP2)
      IF (KINBAS(ICOL) .GT. 0) GO TO 1000
      KINBAS(ICOL) = -1
      NTEMP3 = 1
C
C   MARK OLD BRANCH AS FATHOMED
C
1000  IVID(LISTL) = -IVID(LISTL)
      IVBND(LISTL) = NTEMP1
      IOBND(LISTL) = NEGINF
C
C   UPDATE X IF NECESSARY
C
      IF (NTEMP3 .NE. 0) CALL UPDATX
      RETURN
C
C   NODE FATHOMED: UPDATE VAR. BOUNDS AND BACKTRACK AGAIN
C
2000  IF (ICOL .LT. 0) GO TO 2100
      ICOL = ICOL - (NS * 1000)
      NTEMP1 = IVBND(LISTL)
      IF (KINBAS(ICOL)) 2010,2050,2050
```

```
2010  NTEMP3 = 1
      DP = XUB(ICOL) - XLB(ICOL)
      DY = FLOAT(NTEMP1) - XUB(ICOL)
      IF (DP .LT. DY) KINBAS(ICOL) = 0
2050  XUB(ICOL) = FLOAT(NTEMP1)
      GO TO 3000
C
2100  ICOL = -(ICOL + (NS * 1000))
      NTEMP1 = IVBND(LISTL)
      IF (KINBAS(ICOL)) 2150,2110,2150
2110  NTEMP3 = 1
      DY = XLB(ICOL) - FLOAT(NTEMP1)
      DP = XUB(ICOL) - XLB(ICOL)
      IF (DP .LT. DY) KINBAS(ICOL) = -1
2150  XLB(ICOL) = FLOAT(NTEMP1)
C   CONTINUE BACKTRACKING
3000  LISTL = LISTL - 1
      GO TO 50
      END
C-----------------------------------------------------------------------
      SUBROUTINE WRAPUP
C
C         OUTPUT OPTIMAL SOLUTION AND CORRESPONDING OBJECTIVE VALUE.
C         SUBROUTINE ADAPTED FROM INTEGER PROGRAMMING CODE BB, WRITTEN
C          BY GARY A. KOCHMAN (OPERATIONS RESEARCH, STANFORD UNIVERSITY)
C
      IMPLICIT REAL*4 (A,C,E-H,O,P,P-W,Z), REAL*8 (B,D,X,Y),
     1  INTEGER*4 (I-N,Q)
      INTEGER JH,KINBAS,LA,LE,IA,IE
      INTEGER IPART,INCUMB,IVBND,IVID,IOBND
      DOUBLE PRECISION F(2000)
      REAL A(1000)
C
      COMMON/BBLIST/ DFPART(60),REVBND,INCVAL,ICOL,IVAL,IDIR,IPART(122),
     1  INCUMB(130),IVBND(500),IVID(500),IOBND(500),NPIVOT,IPTYPE,IFEAS
      COMMON XLB(122),XUB(122),DE,DP,B(60),X(60),Y(60),YTEMP(60),A,
     1  E,MSTAT,IOBJ,IROWP,ITCNT,INVFRQ,ITRFRQ,JCOLP,
     2  NROW,NCOL,NELEM,NETA,NLELEM,NLETA,NUELEM,NUETA,JH(60),
     3  KINBAS(122),LA(122),LE(502),IA(1000),IE(2000)
      COMMON/GESTLT/PRICE(130),ICURX(130),ISUMC,ITSINV,LISTL,NBRANC,
     1  NS,NP,JFCOL(11),JFROW(11),JFELEM(11),MAXC(10),MAXC2(10)
C
      LASTC = JFCOL(NP+1) - 1
      IF (ITCNT .GE. ITRFRQ) GO TO 20
      IF (IFEAS .EQ. 0) GO TO 10
      WRITE (21,1)
1     FORMAT (/'        OPTIMAL INTEGER SOLUTION ')
      DO 500 I=1,NP
         WRITE (21,2) I
2        FORMAT (' SUBPROBLEM',I4)
            JBEG = JFCOL(I)
            JEND = JFCOL(I + 1) - 1
            WRITE (21,3) (INCUMB(J), J=JBEG,JEND)
3        FORMAT (1515)
500      CONTINUE
      WRITE (21,4) INCVAL
4     FORMAT (' MAX OBJECTIVE VALUE =', I6)
      RETURN
10    WRITE (21,5) INCVAL
5     FORMAT (/' NO FEASIBLE SOLUTION FOUND WITH OBJECTIVE VALUE >',
```

```
     1  I10)
        RETURN
10      WRITE (21,100) ITCNT
100     FORMAT(/' SIMPLEX ITERATIONS =',I8,':  COMPUTATIONS TERMINATED.')
        IF (IFEAS .EQ. 0) GO TO 10
        WRITE (21,101)
101     FORMAT (' BEST INTEGER SOLUTION FOUND IS:')
        DO 600 I=1,NP
                WRITE (21,2) I
                JBEG = JFCOL(I)
                JEND = JFCOL(I + 1) - 1
                WRITE (21,3) (INCUMB(J), J=JBEG,JEND)
600        CONTINUE
        WRITE (21,6) INCVAL
6       FORMAT(' MAX OBJECTIVE VALUE DISCOVERED =',I6)
        RETURN
        END
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER #95	2. GOVT ACCESSION NO. AD-A089 543	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A COMPUTER PROGRAM FOR THE STAIRCASE INTEGER PROGRAMMING PROBLEM		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL REPORT
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) LYNNE J. POLLENZ		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0418
9. PERFORMING ORGANIZATION NAME AND ADDRESS OPERATIONS RESEARCH PROGRAM ONR DEPARTMENT OF OPERATIONS RESEARCH STANFORD UNIVERSITY, STANFORD, CALIFORNIA		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-061
11. CONTROLLING OFFICE NAME AND ADDRESS OFFICE OF NAVAL RESEARCH OPERATIONS RESEARCH PROGRAM CODE 434 ARLINGTON, VA. 22217		12. REPORT DATE July 1980
		13. NUMBER OF PAGES 62
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Also issued as Technical Report No. 80-16, Dept. of Operations Research Stanford University, under National Science Foundation Grant MCS76-81259.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

INTEGER PROGRAMMING

DECOMPOSITION

STAIRCASE STRUCTURE

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report documents the staircase integer programming computer code SDA. This program is intended for use in solving multitime period integer programming problems with general upper and lower bounds on the variables.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)