SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

152007

University of Miami

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

CONTRIBUTED PROGRAM LIBRARY SUBMITTAL FORM
(for IBM S/360, 1130 and 1800)

SHARE Program Library Agency (PID)
Triangle Universities Computation Center
P. O. Box 12076 .A.
Research Triangle Park, N. C. 27709

This form should be completed and submitted with the program package to PID at the address shown above. Standards and instructions for submitting programs are in your *User Group Reference Manual* or the *Contributed Program Submittal Standards Manual* available from PID.

(1) **Program Order Number** *(to be filled in by PID)* 15.2.007

(2) **System Type** *(machine)* S/360

(3) **Search Key** . . . . . REVISED SIMPLEX
PRODUCT/FORM/ /OF/
LINEAR /PROGRAMMING /

(4) **Name of Author** *(if different than submitter's)* . . . . . Joel Shwimer

(5) **Submitter's Name** *(direct technical inquiries to)* . . . . Joe Walters, Jr.

(6) **Submitter's Address** . . . . . TECHNICAL ASSISTANCE
CURRENTLY NOT AVAILABLE.

(7) **Title of Program** . . . . MFOR 360 LINEAR PROGRAMMING CODE

(8) **Submitter's User Group Affiliation Code and Installation Code** . . . . . S

(9) **Submitter's Own Program Identification and Suffix** *(optional)* . . . . .

(10) **Primary Subject Code** 15.2

(11) **Secondary Subject Codes** . . . . .

(12) **Operating or Monitor System Required** OS/360

(13) **New or Revision Code** *(if revision, show prior Program Order Number in item 1)* . . . . . N

(14) **Year Completed** 67

(15) **Date of Submittal** 03.20.68

(16) **Documentation** *(number of original pages submitted)* 95

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

CONTRIBUTED PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide

a. Purpose
b. Programming Language used
c. Version and modification level or release number of IBM Programming System used, or program order number for non-IBM authored program used
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements
g. Engineering Changes (EC) level of equipment (if pertinent)

ABSTRACT

MFOR 360 is an independent routine which uses the revised simplex method with the product form of the inverse to solve the linear programming problem in standard form. The program is written in FORTRAN IV (G level) with one subroutine in 360 Assembler Language. MFOR 360 has been compiled and tested using OS Version 11 on an S/360 Model 65.

The routine is an all in core routine; therefore no secondary storage is needed. Symbolic control cards direct the operation of MFOR 360.

No special machine requirements are needed.

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

Permission to Publish

"I hereby give anyone permission to reprint, reproduce, and distribute this program to anyone else."

(18) **Signature of Submitter and Date** Joe Walters, Jr. 2/27/68

(19) **Signature of Installation Addressee** J. A. Steinberg 3/20/68

T4SF

TABLE OF CONTENTS

1. CARD DECK KEY

Deck # 1 360 Assembler Deck, sequence 10 through 810 in cc 77-80; UTIL in cc 73-76; 81 cards.

Deck # 2 Fortran IV Deck containing main program and twenty-three subroutines. All routine names are left justified in cc 73-76, and all sequence numbers are right justified in cc 77-80.

BOS	sequence	10	through	1390
CAP		10		1000
DEL		10		200
ERR		10		590
FIN		10		690
GET		10		240
HOT		10		290
INP		10		1870
JMY		10		300
KRT		10		190
LOT		10		190
MIN		10		300
NEW		10		680
OUP		10		410
PIV		10		420
ROW		10		530
SOT		10		270
SS		10		150
TAP		10		690
TIME		10		100
UOUT		10		370
VER		10		1310
XCH		10		210
YXB		10		220

Total of 1261 cards in Deck #2.

Deck # 3 Sample Problem Deck, sequence 10 through 85 in cc 77-80; SMPL in cc 73-76; 85 cards.

2. INTRODUCTION

MFOR 360 is a computer program designed to solve the linear programming problem in standard form. It is a modification of the product form all-in-core linear programming code RS MFOR written primarily in FORTRAN II with FAP subroutines and implemented on the IBM 7090/7094 [1]. MFOR 360 is written in FORTRAN IV (G Level) with one subroutine in 360 Assembler Language and implemented on the IBM 360 Model 65.

The linear programming problem in standard form is :

Find $x_1, x_2, \ldots, x_n$

which minimize

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

subject to the constraints

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i \qquad i = 1, 2, \ldots, m$$

where c_j, a_{ij}, and b_i are given constants. MFOR 360 uses the revised simplex method with product form of the inverse [4,5] to solve this problem. In Phase I the program starts with a full basis and uses the composite simplex procedure to minimize the sum of the infeasibilities [6,3]. In Phase II the program starts with a feasible solution and uses the composite simplex procedure to minimize the objective function.

The present version of MFOR 360 will handle up to 2,000 variables, 511 constraints, and 12,000 non-zero entries, given a maximum of 772 transformations and 15,000 non-zero transformation entries before a reinversion is necessary. This version requires 240,000 bytes of core storage of which approximately 40,000 are taken up by the program. The program can be easily modified, however, to handle larger or smaller problems requiring more or less storage.

Suboptimization, which allows more than one vector to be brought into the basis with only one calculation of the price vector, is the normal mode of operation. The two most negative reduced costs are saved with their associated columns and then if suboptimization is being used, after the first most negative of these is brought into the basis, the second one is tried. If it can be brought in without creating any more infeasibility, it too is brought in. Whether or not suboptimization is used is controlled by the two control cards NSOP and SOPT, with suboptimization being used if nothing to the contrary is indicated.

The number of iterations is defined to be the number of times that the price vector is calculated. The number of steps is defined to be the number of pivots taken exclusive of those in crashing and inverting. Thus, if suboptimization is being performed, the number of steps can and usually will be larger than the number of iterations but, never more than twice the number of iterations. A pivot is counted for every transformation vector created.

Crashing is a method of replacing any artificial variables in the basis by real ones which do not cause any infeasibilities. It uses rules to choose among the possible candidates such as trying the shortest column first. When crashing and inverting, since rules such as the above are used, the code is almost independent of the ordering of the columns in the matrix.

It is possible to start a problem with a specified basis or to restart the solution of a problem from where a previous run had left off by use of BASS and AROW control cards. In the present version they are printed rather than punched by the program. The BASS cards indicate which variables (columns) are to be in the basis, and the AROW cards show which rows still had artificial variables in at the time of the punching of this set of BASS and AROW cards. These are rows in which it is not desired to pivot in any of the columns in the basis if it is desired to start where the other run left off.

Two possible changes which one might wish to make in this program are the size of the arrays used and the format of the input data. Instructions for changing the size of the arrays used are included in the discussion of the Storage System. To change the format for input data only two cards must be modified. These two cards are CAP 980 and TAP 680, which are the format cards for reading in the data associated with the MTRX, ALTA, ALTB, and RHS control cards. Included in the current version as comment cards, CAP 970 and TAP 670, are the format statements which allow standard SHARE input cards to be used for the matrix, objective function and right hand side (with the one exception that the names of rows, columns, etc. must be only 4 characters long; the last 4 of the 6 character SHARE field being taken).

MPOR 360 was written by J. Shwimer under the direction of Dr. C. D. MacRae of the Massachusetts Institute of Technology. This work was done in part at the M. I. T. Computation Center. The program is one part of a research effort to develop computer algorithms for the optimal allocation of scarce resources over time and space. This research was supported in part by funds from the General Motors Corp-

oration under the General Motors Grant for Highway Transportation.

3. CONTROL CARD SCHEME

All control cards use columns 1-4 to determine the nature of the control card. There are, following RS MFOR, three types of control cards: Type 0, Type 1 and Type N. Type 0 control cards have no data cards following them. There is one Type 1 control card, PRMD, which is followed by one associated card containing other necessary information. Type N cards are followed by an arbitrary number of data cards. The data cards must have blanks in columns 1-4 and the data for the Type N control card ends as soon as a punch occurs in columns 1-4. An END card, which marks the end of the data set for the Type N control card, aids the user in visualizing his input deck setup. In any case, the control card following the data set for Type N control cards must have blanks in columns 9-80. All control cards must be punched, left justified, as in indicated on the following pages. For example, the RHS control card must be punched with RHS in columns 1-3 and column 4 left blank. All "O"'s that appear in control card identifiers are the letter O and not the number zero.

All row and column names are input as 4 character fields. All BCD characters (including blanks) may be used. Note that names are considered different unless they agree exactly, character by character, including blanks (hence left or right justification must be consistent). A name used for a row may also be used for a column.

4. CONTROL CARD DESCRIPTIONS

TYPE 0 CONTROL CARDS

BEGN — The first card of each run. It initializes the various constants to nominal values, sets suboptimization to be the mode of operation and restarts the "internal relative timer" to zero.

NSOP — Causes no suboptimization to be performed, that is, not more than one pivot step will be taken per iteration.

SOPT — Restores mode of operation to include suboptimization, that is, up to two pivot steps will be taken per iteration.

ROW — The row name appearing in columns 5-8 will be the objective row. If there is more than one objective row, the OBJ control card must be used before the ROW card. When there is only one objective row and no OBJ card is used, then the ROW card must be used before the matrix is entered.

NEWX — This is used when one wants to solve a problem with more than one right-hand side to a single matrix. When a problem has been solved, and then a new RHS is read, this control card is used to update the solution vector. If, when a new RHS is read, the INVT control card is also used, then the NEWX is not necessary since inversion automatically updates the

solution vector.

SNGL — Either this control word or SLCK or BASS should be used when starting a problem after reading in the matrix and right-hand side. This will cause a basis to be inserted which contains all of the feasible singletons together with the proper inverse. A singleton is a column having exactly one non-zero entry in the set of constraint rows, and the sign of this entry is the same as the corresponding right hand side entry.

SLCK — This causes a basis to be set up which contains only positive slacks. A positive slack is a column having only one non-zero entry in all of the constraint rows and this value must be a +1. A slack is put into the basis regardless of the sign of the corresponding right-hand side entry. Note that no pivoting need be done.

INVT — The current basis will be inverted.

CRSH — This will cause an attempt to increase the size of the basis to capacity while maintaining the present level of feasibility. If BASS has been used to input a partial basis for this run, INVT must be used before CRSH is used.

GO — This will cause the current problem to be attempted. This means optimization in the objective row specified by the ROW control card is tried. If BASS has been used to input a partial basis for this run, or if the matrix has been changed by an ALTA card after solving a previous problem with this matrix, INVT must be used before GO may be used. When a terminating condition is reached, the appropriate output will be given and then error output, as done by ERRS, will automatically be done.

SOLV — This is equivalent to the three successive control cards SNGL, CRSH, and GO. This would be used only if there was no starting basis.

GOGO — This is the same as GO except that all objectives are solved for, in the order in which they were read.

ERRS — This will cause row and possible column errors to be computed and printed. See the subroutine ERR for a more complete description.

OUTP — This causes output with condition No. 6.

PNCH — This causes punching of the BASS and AROW card for the present basis. In the present version they are printed.

SINV — This causes the inversion frequency to be set to (M/10) +6 where M is the number of rows in the problem. This control card should be used only after the matrix has been read in.

TIME — This causes the time in seconds since the last re-setting of the "internal relative time", by either the start of execution or the BEGN control card.

STOP — Indicates the end of all the control cards. Stop the job.

END — No action is caused at all. It is most useful to terminate data sets for type N control cards.

Comment Card — If columns 1-4 are not blank and do not contain any of the defined control words, this card will be treated as a comment and will be printed out.

TYPE 1 CONTROL CARD

PRMD — The one data card following this control card specifies all of the desired output options. On the data card in the first 8 columns are punched eight integers. The card setup is as follows:

COLUMN	OCCURRENCE OF OUTPUT	NOMINAL VALUE
1	Every pivot step, just before step is taken	0
2	When feasibility is declared	0
3	When optimality is declared	9

COLUMN	OCCURRENCE OF OUTPUT	NOMINAL VALUE
4	When it is determined that no feasible solution exists	9
5	When an infinite solution is found	9
6	Special condition specified by OUTP card	6
7	When space for trans-formations is exceeded	9
8	When cutoff due to time or iteration limit exceeded	9

The meaning of the punched values are as follows:

VALUE	MEANING
0	No output
1	Short output only
2	Short and row output
3	Short and column output
4	Short, row, and column output
5	Basis card output only
6	Short and basis card output
7	Short, row and basis card output
8	Short, column and basis card output
9	Short, row, column and basis card output

TYPE N CONTROL CARDS

OBJ — The data cards for this control card contain the names of the objective rows. Any number of data cards may follow the OBJ card. Each card must have columns 1-4 blank. In columns 5-48 there

are eleven 4 character fields (each field ending in a column which is a multiple of 4). Any field which is blank is ignored. Any non-blank field is placed into storage as the name of an objective row. When a problem has more than one objective, these objectives must be read in before the matrix or right-hand side. Note that any row which is not a constraint should be entered as an objective row. The actual row that is used as the objective is determined by the control card ROW.

MTRX

If a name is to be associated with this matrix it should be placed in columns 5-8 on the MTRX card. Each data card contains one matrix entry. The format is:

Col. 1-4	Blank
Col. 5-8	Column name
Col. 9-12	Row name
Col. 13-24	Numerical entry. If decimal point is not punched it is assumed to be between columns 18 and 19.

All entries with the same column name must be adjacent in the deck. No restriction is made on the ordering of the row names. Zero entires may, and usually should be omitted. However, if a zero entry is going to be changed later by an ALTA card, it must be entered in the initial matrix.

RHS

If a name is to be associated with this right-hand side, it should be placed in columns 5-8 on the RHS card. This control card zeroes out the right-hand side vector and then enters the appropriate numbers as read from the data cards which follow. Each data card contains one right-hand side entry in the following format:

Col. 1-4	Blanks
Col. 5-8	Ignored
Col. 9-12	Row name
Col. 13-24	Right hand side entry. If not punched, the decimal point is assumed to be between columns 18 and 19.

No restriction is made on the ordering of the row names. RHS may be used before or after MTRX. When RHS is used before MTRX, the ordering of the row names appearing in the RHS data cards will be preserved for the row output. RHS may be used more than once for the same matrix.

FRST

Same as RHS

BASS

The data card format is the same as the OBJ data cards. Each name entered in a data card is the name of a column which will be entered into the basis, if possible, when an inversion is performed. If a name appears which is not a column name, an error message is printed, but execution continues. An AROW control card must be used with BASS even if the set of artificial rows is empty.

AROW — The data card format is the same as for the OBJ data cards. This card is used with BASS to indicate which rows, if any, in which it is not desirable to pivot.

ALTA — The data card format is the same as for the MTRX data cards. This may be used to change an entry that had been read in under MTRX. However, the entry must have been previously read in under MTRX or else the run will be skipped.

ALTB — The data card format is the same as for the RHS data cards. This may be used after a RHS has been read in for this matrix to change a right-hand side entry.

TOLR — Each data card contains the information needed for the input of one floating point constant in the following format:

Col. 1-4	Blanks
Col. 5-8	Constant name
Col. 9-16	Numerical value in E8.1 format

BEGN sets these quantities to the following nominal values:

COL. 5-8	MEANING	NOMINAL VALUE
PIV	Minimum element size considered for pivoting	1.0E-05
REJ	Pivot rejection tolerance	1.0E-03

COL. 5-8	MEANING	NOMINAL VALUE
COST	Reduced cost considered negative below this value	-1.0E-03
RSET	Tolerance for setting x to zero	1.0E-05
ENTR	Minimum element size in new transformation	1.0E-07
TMLT	Minimum element size considered for applying transformation	1.0E-10
MIX	Mixed pricing ratio	0.0
TIMX	Maximum time since last resetting of internal timer in seconds. Job will be terminated if this time limit is exceeded.	1.0E+03

FREQ — Each data card contains the information to input one fixed point constant in the following format:

Col. 1-4	Blank
Col. 5-8	Name of constant
Col. 9-16	Numerical value (right-justified integer).

BEGN sets these quantities to nominal values of 10,000:

Col. 5-8	Meaning
OUTP	Frequency of type 6 output
INVT	Reinversion frequency
CTOF	Maximum number of interactions allowed

The type 0 control card SINV replaces the constant input by the above INVT data card, and vice versa. The output frequency may not occur exactly as prescribed because an iteration on which a reinversion

occurs (because inverse is too large) and the final iteration of a solution are not counted.

5. ORDERING OF THE CONTROL CARDS

Input for different runs may be stacked, where a run is considered to be all the computation that is done with one set of matrix data. If an ALTA card is used to change a matrix, it is still, however, considered to be the same run.

The BEGN card denotes the start of a run. Before the matrix or right-hand side may be read in either an OBJ or a ROW control card must be entered. If there is more than one objective for a given matrix, this must be indicated by the OBJ card, whereas only one objective can be specified by either the OBJ or ROW card. On any given run, the OBJ card cannot appear after the matrix and/or initial right-hand side have been read in, but a ROW card can (and may have to in the case of multiple objective problems).

The order of MTRX and RHS is arbitrary. More than one RHS card may be used with the same matrix; i.e., multiple right-hand side problems, but an ALTB card may not be used until after an RHS card.

The following control cards may only be used after the matrix has been read in.

BASS	INVT	GOGO
AROW	CRSH	NEWX
ALTA	SINV	ERRS
SLCK	GO	OUTP
SNGL	SOLV	PNCH

All other control cards not mentioned may be used at any place in the run. These include the following control words whose effect is to control data. (The data these cards read in is in

effect until replaced by data read in from another control card of the same type or until a BEGN card replaces it with the nominal values).

SOPT	PRMD	TOLR
NSOP		FREQ

The control card TIME may appear at any place, as can the END card and comment cards. It is suggested that all sets of data for Type N control cards be ended with an END card.

The last card in the entire input deck, possibly after the input for a number of runs, must be a STOP card. This is the signal to the program that the job has been completed.

6. STORAGE SYSTEM

MFOR 360 uses an intricate storage system to obtain a fast, efficient code that is capable of solving large problems. This storage system enables the problem to be solved while remaining entirely within the core of the computer. Although knowledge of this storage scheme is not essential for every user, it is clearly necessary for anyone who wants to understand the coding.

The storage system is designed to take advantage of the fact that the coefficient matrices of large linear programming problems are usually quite sparse. That is, only a small percentage of the a_{ij}'s will be non-zero. Therefore space is only reserved for these non-zero entries, or for those entries which, though presently zero, may be changed to a non-zero value, via an ALTA card, during the same execution of the program. For simplicity in the following discussion, both of the above types of entries shall be called non-zero entries.

A number of lists are used in carrying out the storage scheme, and although these do appear in Appendix B, they are also summarized in the chart below.

Lists for Storage Words

Name	Use
A	Coefficient matrix entries
IA	Coefficient matrix entry row numbers
KL	Column locators for the coefficient matrix entries
KN	Column names
NR	Row names
B	Right hand side
IP	Row pivoted for transformation
LE	Column locator for transformation entries
IE	Transformation entry row numbers
E	Transformation entries

A list of the column names, KN, and row names, NR, is compiled, adding a name to the proper list whenever it is first encountered. Each column and row is thus assigned an internal number, the 10th name in either list being assigned the number 10, which serves to identify that particular column, row, during this execution of the program.

The non-zero matrix entries are all read into the A list. They must be entered one column at a time, with the row order within each column being completely immaterial. Note that the order of the columns as read in is the same as their order in the column name list. To identify to which column a given A list entry belongs, a list associated with the column names list, KL, is created. Corresponding to each entry in the column name list, KN, the number of the entry in the A list at which this column starts, is placed in the KL list. Thus, if the fifth column in the matrix consists of the fiftieth thru the sixty-third non-zero matrix entries, the number fifty would appear as the fifth entry in the KL list.

To completely specify what an A list entry is,the row must also be known. This is accomplished by creating a list, IA, associated with the A list. Each entry in the IA list is the internal number of the row to which the corresponding A list entry belongs.

In specifying the problem to be solved, the right hand side of each linear equation, i. e. each b_i, must also be entered. These are placed in the B list. The coefficients c_j in the objective function are not entered in a separate list. They are considered to be a row, or rows, of the coefficient matrix, and as such, are entered in the A list. The particular row, or rows, which contain the objective function is indicated by the OBJ or ROW control card.

MFOR 360 uses the product form of the revised simplex method, in which the inverse from the previous step is premultiplied by an elementary matrix, i.e. one differing from the identity matrix in only one column. Rather than carrying out this multiplication immediately, the needed information will be saved for use later. To use up as little space in the computer as possible in storing this elementary matrix, only the one "different" column, called the transformation vector, and the location of where this column appears in the elementary matrix will be saved. There is always a starting inverse, namely the identity matrix which corresponds to having the basis consist of all artificial variables, one for each equation.

Since the original coefficient matrix is usually quite sparse and the basis inverse matrix is derived from the coefficient matrix, it is natural to expect that these transformation vectors will also often have a large number of zero entries. This, in fact, is the case. To take advantage of this, a storage scheme almost identical to the one described above for the coefficient matrix will also be used here. An E list, corresponding to the A list above, is created to hold the actual transformation entries. The entries are entered, transformation by transformation, as created, and only the non-zero entries need be stored. The information as to where in the elementary matrix this particular column belongs is stored in the IP list. It has one entyy for each transformation which is the number of the row for which that transformation is to be pivoted. An LE list, corresponding to the KL list above, is formed to indicate where the entries

for each transformation start in the E list. Finally an IE list, corresponding to the IA list, is made containing the number of the row to which each transformation entry in the E list belongs.

This completes the general discussion of the storage system. Other lists are used during the solution of the problem and these appear together with their use in Appendix B. Some additional comments are in order about the present implementation of the storage system as described above.

In RS MFOR a relocation system was used so that once the matrix entries and the right hand sides had been read in, these lists could be adjusted to take up only the needed amount of storage. Thus as much room as possible was left for the transformation entries and the associated information that had to be stored with them.

To simplify conversion and to avoid certain problems caused by the present stage of the compiler being used, the relocation part of the storage scheme, however, had to be discarded. Instead, all of the lists were placed in COMMON with fixed boundaries set for the maximum possible number of entries. To save space halfwords were used wherever possible, namely for the IA, KL, LE, and IE lists.

The calculation of the space necessary for these lists, all those in the table above and also the JH, KB, P, X, and Y lists, can be made as follows:

Let S be the maximum number of non-zero matrix entries allowed,
M be the maximum number of rows (including objective row or rows),
N be the maximum number of columns,

T be the maximum number of transformations before a reinversion is necessary, and
L be the maximum number of transformation entries before a reinversion is necessary.

Then the storage required for the needed lists is

$$6S + 24M + 10N + 6T + 6L \text{ bytes.}$$

In addition, 536 bytes of COMMON are needed for other constants or quantities used in solving the problem (see Appendix B).

Some typical values for these parameters that might be used together with the total amount of storage they require are as follows:

	Set 1	Set 2
S	12,000	2,000
M	511	250
N	2,000	750
T	772	375
L	15,000	6,000
Total Bytes used including other constants (base 10)	199,432	64,286
Total Bytes used including other constants (base 16)	30B08	FB1E

In the present version, which is implemented on an IBM 360 Model 65, the first set of values listed above were used. On this machine the program including the FORTRAN routines it uses under Operating System took up 40,664 bytes (base 10), or 9ED8 bytes (base 16).

If it is desired to change the size of any of the above storage parameters (i.e. S,M,N,T, or L), the following two things must be done:

1. Change the COMMON statements in all of the FORTRAN subroutines;

2. Change the appropirate value in the KM list by modifying the statements on cards FIN 410 through FIN 450 where KM(6), Km(20), KM(21), KM(36), and KM(40) are set.

If this program were to be run on a 128K machine, by using the values given above under Set 2, the program and COMMON would take up 104,950 bytes (base 10) or 199F6 bytes (base 16) leaving 26,122 bytes (base 10) or 660A bytes (base 16) for the monitor system.

7. SUBROUTINE DESCRIPTIONS

BOS

This is the master computing routine. Its work consists mainly of carrying out the following six steps:

1) Check for feasibility using routine XCK.
2) Calculate the appropriate prices using routine GET.
3) Find the non-basic column with the minimum reduced cost using routine MIN. If the minimum reduced cost is non-negative, testing for termination must be done.
4) Obtain the column to be entered into the basis in its updated form by using routine JMY to apply any previous transformations to it.
5) Choose a pivoting row by using routine ROW. Checking for an infinite solution is done now as explained in the description of routine ROW.
6) Update the inverse and the solution vector by using routine PIV. Return to step one.

This routine also performs a suboptimization procedure if it is desired. It causes a "re-inversion" of the basis at the appropriate times, using routine VER. It tests if various types of limits; e.g., space, total iterations, etc., have been exceeded and takes the proper action.

This routine completes its job by finding an optimal solution, an infinite solution, no feasible solution, or one of the limits exceeded. It causes the appropriate output to be printed, using routine UOUT, and returns control to routine INP for the reading of the next control card.

CAP

This routine is used to enter the right hand sides of the constraints by RHS. It also contains the mechanism for changing matrix entries, by ALTA, or right hand sides, by ALTB. Specification of which rows can be used as objective rows, by OBJ, which columns should be entered initially into the basis, by BASS, and which rows it is desired not to pivot, by AROW, are also accomplished in this routine.

DEL

The reduced cost for a given column is calculated by multiplying that column by an appropriate price vector which is determined by routine GET.

ERR

The errors on each row and, if the present solution is feasible, each column are calculated. The maximum row error, where it occurs, and the sum of the absolute values of the row errors are printed. The same procedure is carried out for columns. Note that the objective row is included.

On rows, the error is defined to be the amount by which the equation, Ax - b, is not satisfied. On all columns corresponding to variables in the basis, the error is defined to be the difference of the reduced cost from zero. The error is not calculated for any other column.

FIN

This is the main routine which is used at the start of each run and whenever the control card BEGN is encountered. It initializes constants, calls the major input routine, INP, and also calls the routine TAP for reading in the coefficient matrix.

GET

This routine sets up the price vector. If the present solution is still infeasible, the objective row entry is set to the mixed pricing ration, entries corresponding to rows with negative solutions are set to positive one, +1, and entries for rows with artificial variables are set to negative one, -1. All other entries are set to zero. Updating of this vector by previous transformations is accomplished by routine KRT.

If the solution has already become feasible, the entry corresponding to the objective row is set to positive one, +1, with all other entries being set to zero. Updating by the routine KRT occurs as in the infeasible case.

HOT

This routine prints out the column dependent quantities. In the present version, these quantities are the name, value, true cost, and reduced cost of each variable. The WRITE statement for printing the row in which each variable occurs in the present basis is also in this routine. In the present version, however, the WRITE statement and its FORMAT are comment cards.

INP

This is the major input routine. It reads in a control card, calls upon the SEARCH portion of the utility package to determine which type of control card has been read in, and then transfers to the appropriate places to carry out the operations indicated by the control card. This may involve executing just a few lines of coding or may be as involved as calling several other routines. Once the desired actions have been completed, this routine takes care of reading in the next control card.

JMY

This routine generates one transformed matrix column. It takes the original column from the coefficient matrix and then multiplies it by all the transformation vectors which have been previously generated in the order they were created.

KRT

The routine applies the transformations previously created to the price vector through multiplication. The order that these transformations are applied is the reverse of that in which they were created.

LOT

This routine prints out the row dependent quantities. In the present version, these quantities are the row name, the right hand side, the associated price, and either the row error, indicated by

ERRS, or corresponding entry of the last column to be transformed, indicated by the name of the column. The WRITE statement for printing the variable which appears in each row of the basis and its value is also in this routine, but again the WRITE and FORMAT statements are comment cards.

MIN

This routine finds the minimum reduced cost among the non-basic variables, saving the reduced cost and the name of the associated variable for both the minimum and the second lowest reduced cost for suboptimizing.

NEW

This routine pivots in the singletons. It may also select slacks to pivot in. The choice of which group is to be entered is determined by a parameter set before this routine is called. Each column is checked to see if there is only one non-zero entry among the constraint rows. If the objective row entry is non-zero we can still have a singleton but not a slack. Once such a column is found, a check is made to make sure that the one non-zero entry has the same sign as the corresponding right hand side. If singletons are being entered and the signs are not the same, this column is rejected. Having found a legitimate slack or singleton which is to be entered, it is checked to see if the entry is a positive one. If yes, the row and column are simply interchanged. If it is not a +1, a transformation must be created and routine PIV

must be used. At the end, the total number, by type, of the variables entered is printed.

OUP

This routine prints the BASS and AROW control cards and the associated data naming the specified column names.

ROW

This routine selects the pivot row. In making the selection, the updated version Y, of the column already chosen to enter by MIN, and the present solution vector X are of great importance. The routine looks first at all rows whose associated X values are zero. If any of these variables are artificial, it chooses the row with the largest absolute value of Y. If no artificial variables are at zero level, it chooses from among the real zero level variables, the one having the largest positive Y value. If a row has not been found yet, it looks at the rows associated with positive X values. It selects the one having the minimum Y value. It also looks at the rows with negative X values. If there are any with X/Y less than the minimum X/Y from the "positive" rows, it chooses the one with the largest absolute value of Y. Otherwise it keeps the choice found above.

If no suitable pivot row is found, routine BOS takes care of seeing whether an infinite solution has been discovered; i.e., if mixed pricing ratio is presently zero.

PIV

This routine adds a transformation to the inverse. This is accomplished when using the produce inverse form, as this program does, by adding a vector to premultiply the inverse with. If column $a_j = (a_{1j}, a_{2j}, \ldots, a_{mj})$ is to be brought into the basis as $e_i = (0,0,0,\ldots,1_i,0,0,0)$, then premultiply the inverse with $d = \left(\frac{a_{1j}}{a_{ij}}, \frac{a_{2j}}{a_{ij}}, \ldots, \frac{a_{mj}}{a_{ij}}\right)$. The solution vector is updated in the manner normal to simplex methods.

SOT

This routine prints out what is considered a "short output." This consists of a condition number, the name of the objective row, the number of infeasibilities, the names of the matrix and right hand side if any were given, the iteration number, the number of steps and the number of pivots that have been performed, the present value of the objective function, the sum of the infeasibilities, the value of the determinant, the minimum reduced cost among the non-basic variables, the name of the column just brought into the basis, the one just removed, and the name of the pivot row.

TAP

This routine enters the matrix of coefficients. It also checks that size limits on this matrix have not been exceeded and that a matrix is not read in at the wrong time.

UOUT

This is the output control routine. It determines the type of output desired under the present conditions and calls the appropriate routines to provide this output; i.e., one or more of HOT, LOT, OUP, SOT. It also performs page ejection when necessary.

VER

This routine performs the reinversion of the basis and will also carry out a process called feasible crashing. Which of these two operations is to be performed is determined by a parameter set before this routine is called. In the present version, the same messages are printed whether reinversion or crashing is being done.

Reinversion is performed either when there is no room left to store transformations or when it is felt the round-off error can be kept down by reinverting at that time.

Crashing is a method of replacing any artificial variables in the basis by admissable variables which do not cause any infeasibilities. It uses rules to choose among the possible candidates such as trying the column with the most zero entries first.

XCK

This routine checks the present solution vector setting to zero any element falling inside a given tolerance limit. It then checks for feasibility setting flags to indicate the presence of any artificial variables or negative values in the present solution.

YXB

This routine generates a new solution vector from the right hand side by applying to it the transformations that have been created. The order of application is the same as the order of their creation.

UTILITY PACKAGE

This is the only routine not written in FORTRAN IV. It contains a searching algorithm, SEARCH, used by routine INP to determine the type of control card just read in. It has four dummy entry points, EXT1, EXT2, EXT3, EXT4, which can be used for future extensions of this program. Also included in this utility package is a supporting routine for the FORTRAN IV coded function TIME.

TIME

This routine provides a method of measuring relative time. An argument of zero causes the clock, as far as this program is concerned, to be set to zero. A non-zero argument causes the time in seconds, with two meaningful digits after the decimal point, since the last resetting of the clock to be returned.

SS

This function controls whether optimal reinversion should be performed. This type of reinversion is performed whenever the average time per iteration becomes less than the average time spent on the last four iterations, the number 4 being an arbitrary choice.

8. NUMERICAL EXAMPLE

Formulation and Problem Setup

Take the problem[1] of minimizing the cost of a feed mixture for dairy cows. The mixture must meet certain minimum requirements for nutrients. The following data are given.

Feed	Total Nutrients	Proteins	Calcium	Phosphorus	Cost $/100 lbs.
Corn	78.6%	6.5	0.02	0.27	2.40
Oats	70.1	9.4	0.09	0.34	2.52
Milo maize	80.1	8.8	0.03	0.30	2.18
Bran	67.2	13.7	0.14	1.29	2.14
Flour middlings	78.9	16.1	0.09	0.71	2.44
Linseed meal	77.0	30.4	0.41	0.86	3.82
Cottonseed meal	70.6	32.8	0.20	1.22	3.55
Soybean meal	78.5	37.1	0.26	0.59	3.70
Gluten feed	76.3	21.3	0.48	0.82	2.60
Hominy feed	84.5	8.0	0.22	0.71	2.54
Minimum Requirements	74.2	19.9	0.21	0.67	

Let the various types of feed be the variables $x, x_2, \ldots x_{10}$. Four slack variables will be added since the constraints are of the greater-than type. The problem may now be stated as:

Minimize,

$$2.40x_1 + 2.52x_2 + 2.18x_3 + 2.14x_4 + 2.44x_5 + 3.82x_6 + 3.55x_7 + 3.70x_8 + 2.60x_9 + 2.54x_{10}$$

Subject to

$$78.6x_1 + 70.1x_2 + 80.1x_3 + 67.2x_4 + 78.9x_5 + 77.0x_6 + 70.6x_7 + 78.5x_8 + 76.3x_9 + 84.5x_{10} - SLK1 = 74.2$$

[1]This is problem number 28, page 114, [4].

$$6.5x + 9.4x_2 + 8.8x_3 + 13.7x_4 + 16.1x_5 + 30.4x_6 + 32.8x_7 + 37.1x_8 + 21.3x_9 + 8.0x_{10} - SLK2 = 19.9$$

$$02x + .09x_2 + .03x_3 + .14x_4 + .09x_5 + .41x_6 + .2x_7 + .26x_8 + .48x_9 + .22x_{10} - SLK3 = .21$$

$$.27x + .34x_2 + .30x_3 + 1.29x_4 + .71x_5 + .86x_6 + 1.22x_7 + 0.59x_8 .82x_9 + .71x_{10} - SLK4 = .67$$

$$x + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} = 1.0$$

For the purpose of identifying the rows, the following names were chosen.

NUTR - for total nutrient

PROT - for protein

CAL - for calcium

PHOS - for phosphorus

UNIT - for unit constraint

COST - for objective function

The following column names were also used:

CORN - for corn

OATS - for oats

MAIZ - for milo maize

BRAN - for bran

FLOU - for flour middlings

LINS - for linseed meal

COTT - for cottonseed meal

SOY - for soybean meal

GLUT - for gluten meal

HOMI - for hominy feed

SLK1 - slack variable

SLK2 - slack variable

SLK3 - slack variable

SLK4 - slack variable

The card input was set up as shown in the Input Example.

INPUT DECK


```
BEGN                                                    SMPL  10
FREQ                                                    SMPL  20
    CTOF       25                                       SMPL  30
END                                                     SMPL  40
TOLR                                                    SMPL  50
    REJ   1.0E-4                                        SMPL  60
END                                                     SMPL  70
ROW CCST                                                SMPL  80
RHS REQ                                                 SMPL  90
        NUTR  74.20                                     SMPL 100
        PROT  19.90                                     SMPL 110
        CAL     .21                                     SMPL 120
        PHOS    .67                                     SMPL 130
        UNIT   1.00                                     SMPL 140
END                                                     SMPL 150
MTRX-A-                                                 SMPL 160
    CORNNUTR  78.60                                     SMPL 170
    CORNPROT   6.50                                     SMPL 180
    CORNCAL     .02                                     SMPL 190
    CORNPHOS    .27                                     SMPL 200
    CORNUNIT   1.00                                     SMPL 210
    CORNCOST   2.40                                     SMPL 220
    OATSNUTR  70.10                                     SMPL 230
    OATSPROT   9.40                                     SMPL 240
    OATSCAL     .09                                     SMPL 250
    OATSPHOS    .34                                     SMPL 260
    OATSUNIT   1.00                                     SMPL 270
    OATSCOST   2.52                                     SMPL 280
    MAIZNUTR  80.10                                     SMPL 290
    MAIZPROT   8.80                                     SMPL 300
    MAIZCAL     .03                                     SMPL 310
    MAIZPHOS    .30                                     SMPL 320
    MAIZUNIT   1.00                                     SMPL 330
    MAIZCOST   2.18                                     SMPL 340
    BRANNUTR  67.20                                     SMPL 350
    BRANPROT   3.70                                     SMPL 360
    BRANCAL     .14                                     SMPL 370
    BRANPHOS   1.29                                     SMPL 380
    BRANUNIT   1.00                                     SMPL 390
    BRANCOST   2.14                                     SMPL 400
    FLOUNUTR  78.50                                     SMPL 410
    FLOUPROT  16.10                                     SMPL 420
    FLOUCAL     .09                                     SMPL 430
    FLOUPHOS    .71                                     SMPL 440
    FLOUUNIT   1.00                                     SMPL 450
    FLOUCOST   2.44                                     SMPL 460
    LINSNUTR  77.00                                     SMPL 470
    LINSPROT  30.40                                     SMPL 480
    LINSCAL     .41                                     SMPL 490
    LINSPHOS    .86                                     SMPL 500
    LINSUNIT   1.00                                     SMPL 510
    LINSCOST   3.82                                     SMPL 520
    COTTNUTR  70.60                                     SMPL 530
    COTTPROT  32.80                                     SMPL 540
```

```
    COTTCAL      .20
    COTTPHOS    1.22
    COTTUNIT    1.00
    COTTCOST    3.55
    SOY NUTR   78.50
    SOY PROT   37.10
    SOY CAL      .26
    SOY PHOS     .59
    SOY UNIT    1.00
    SOY COST    3.70
    GLUTNUTR   76.30
    GLUTPROT   21.30
    GLUTCAL      .48
    GLUTPHOS     .82
    GLUTUNIT    1.00
    GLUTCOST    2.60
    HOMINUTR   84.50
    HOMIPROT    8.00
    HOMICAL      .22
    HOMIPHOS     .71
    HOMIUNIT     .00
    HOMICOST    2.54
    SLK1NUTR   -1.00
    SLK2PROT   -1.00
    SLK3CAL    -1.00
    SLK4PHOS   -1.00
END
TIME
SOLV
TIME
STOP
```

```
SMPL 550
SMPL 560
SMPL 570
SMPL 580
SMPL 590
SMPL 600
SMPL 610
SMPL 620
SMPL 630
SMPL 640
SMPL 650
SMPL 660
SMPL 670
SMPL 680
SMPL 690
SMPL 700
SMPL 710
SMPL 720
SMPL 730
SMPL 740
SMPL 750
SMPL 760
SMPL 770
SMPL 780
SMPL 790
SMPL 800
SMPL 810
SMPL 820
SMPL 830
SMPL 840
SMPL 850
```


Analysis of Output

This sample problem was run at the MIT Computation Center on an IBM System/360 Model 65 with an Associated Support Processor (IBM System/360 Model 40).

The first six lines give us the controls that were entered to change the cutoff value and the rejection tolerance level from the nominal values. Next we see that a ROW card and PHS card were entered naming the objective function as COST and the right hand side as REQ.

The MTRX card is now printed with the matrix name, -A-. In the input deck the matrix entries follow MTRX. Immediately following MTRX we find a one line summary of the matrix. In this case 6 rows (including the objective function), 14 columns, and 64 entries. The TIME card shows that .67 seconds have been used reading in the data and storing it. SOLV was the next card in the input deck. Remember that SOLV generates the sequence SNGL, CRSH, and GO. SNGL will try to generate a basis and in this case we see that it was not able to insert either slacks or non-slacks. The reinversion is a result of CRSH. Note that no iterations of the simplex algorithm have been used and that no transformations have been created. The type number is explained in Appendix C under KM(35). During inversion five transformations were created. The poor columns are columns in which all available entries were below the pivot tolerance. The rest of the output up to the TIME and STOP messages is the result of GO. First a message is printed indicating the point at which the solution became feasible. In

this case we see that the output from CRSH was feasible. The next two lines tell us that an optimal solution has been found and give the relative time. The BASS and AROW output gives the current basis and may be used to restart the program with the basis. See the section on the BASS and AROW cards for information on how to restart using this output.

The short output is preceded by (3) indicating an optimal status. This output indicates that the value of the objective function is 2.51526. There are no infeasibilities and the value of the determinant is 7.5445. In addition, the last variable pivoted into the basis was SLK1, which replaced SOY. The latest pivot row chosen was PHOS.

The "row output" follows this short output. Each row is listed with its right hand side and shadow price. The last column is the contents of temporary storage, in this case SLK1. Next comes the "column output", listing every column and its value in the current solution. In addition, the true cost (i.e., that cost appearing in the objective function) and the reduced cost is given for each column. The last two lines of the GO output are the maximum error for rows and columns. The final few lines are the result of the last two cards TIME and STOP. Note that the total time used was 1.02 seconds.

```
FREQ
        CTOF       25
END
TOLR
        REJ    0.1000E-03
END
ROW COST
RHS REQ
MTRX-A-
     PROBLEM HAS     6 ROWS,  14 COLUMNS, AND     64 MATRIX ENTRIES.
TIME
THE TIME IS NOW       0.67 SECONDS
SOLV
**INSTALLED      0 SLACKS,      0 NON-SLACKS
REINVERTING AFTER      0TH ITERATION.     0 TRANSFORMATIONS WITH      0 ENTRIES, TIME=         0.70 TYPE 0
*INVERSION COMPLETED     0 SLACKS,     8 POOR COLS,     5 TRANSFORMATIONS WITH     30 ENTRIES, TIME=         0.72

*FEASIBLE ON ITERATION    0,      0 STEPS
*OPTIMAL SOLUTION
THE TIME IS        0.78 SECONDS.
BASS  -A-    REQ  ITERN     4
   SLK3SLK4GLUTSLK1BRAN
END
AROW
END
(3) MATRIX R.H.S. ITER STEPS PIV, OBJECTIVE COST        0 INFEAS  DETERMINANT MIN. R/COST  NEW COL  OLD COL  PIV ROW
      -A-     REQ      4      5   10           2.51526        0.0    7.59995E  0        0.06053      SLK1       SOY       PHOS

     ROW           RHS              PRICE             SLK1


   COST          0.0               1.000000  -0.359435E-02
   NUTR         74.199997          0.000000  -0.554385E-01
   PROT         10.899994         -0.060526   0.446875E-01
   CAL           0.210000          0.0       -0.184165E 00
   PHOS          0.670000          0.0        0.598144E-01
   UNIT          1.000000         -1.310790   0.124351E 00


  NAME          VALUE                TRUE     /COST/      REDUCED

  CORN**********                    2.400000              0.695790
  OATS**********                    2.520000              0.640264
  MAIZ**********                    2.179999              0.336579
  BRAN          0.184210            2.139999              0.0
  FLOU**********                    2.440000              0.154737
  LINS**********                    3.820000              0.669211
  COFF**********                    3.549999              0.253946
  SOY ***********                   3.700000              0.143684
  GLUT          0.815790            2.599999              0.000001
  HOMI**********                    2.540000              0.745002
  SLK1          0.423669            0.0                  -0.000000
  SLK2**********                    0.0                   0.060526
  SLK3          0.207368            0.0                   0.0
  SLK4          0.236579            0.0                   0.0
```

```
MAX ERROR ON ROW NUTR= -0.30518E-04, SUM= 0.54576E-04
MAX ERROR ON COL GLUT=  0.95367E-06, SUM= 0.07975E-06


TIME
THE TIME IS NOW       1.02 SECONDS
STOP
THE END HAS COME.
```

APPENDIX

A. SUMMARY OF CONTROL CARDS

Columns 1-4	Type	Function	Format of Data
BEGN	0	Initialize	
OBJ	N	List of objective names	(4X, 11A4)
ROW	0	Objective row name in columns 5-8	
MTRX	N	Matrix entries, one per card	(4X, 2A4, F12.6)
RHS	N	Right hand side entries	(8X, A4, F12.6)
RRST	N	Same as RHS	(8X, A4, F12.6)
PREQ	N	Fixed point constants	(4X, A4, I8)
TOLB	N	Tolerances and mixed pricing ratio	(4X, A4, F8.1)
PRMD	1	Output controls	(8I1)
BASS	N	Names of columns in basis	(4X, 11A4)
AROW	N	Names of rows which are artificial	(4X, 11A4)
ALTA	N	Change entries in matrix	(4X, 2A4, F12.6)
ALTB	N	Change right-hand side entry	
NEWX	0	Calculates X from B and transformations	
SNGL	0	Installs singletons in basis	
SLCK	0	Installs slacks in basis	
INVT	0	Invert	
SOPT	0	Causes suboptimization to be used	
SINV	0	Sets inversion frequency to (m/10)+6	
NSOP	0	Causes suboptimization not to be used	
CRSH	0	Force in as complete a basis as possible	
GO	0	Solve problem for a single objective	
GOGO	0	Same as GO, but solve for all objectives	

STOP	0	End of job
SOLV	0	Equivalent to SNGL, CRSH, GO
ERRS	0	Calculate and output errors
OUTP	0	Output condition 6
PNCH	0	Punch (it now prints BASS and AROW cards)
TIME	0	Output relative time
END	0	No action
EXT1	0	Reserved for future subroutine
EXT21	0	Reserved for future subroutine
EXT31	0	Reserved for future subroutine
EXT41	0	Reserved for future subroutine
	0	Blank control card is an error-skip to next BEGN
ALL OTHERS	0	Comment card

APPENDIX

B. LIST OF STORAGE USED IN COMMON

Symbol	Dimension in words	Use
M	1	Number of rows in problem
MA	1	Not used
MB	1	Not used
MC	1	Number of the row which is the objective
MD	1	Not used
ME	1	Number of transformations
MF	1	Number of row which is the first constraint
N	1	Number of columns
KM	50	Fixed point information
KP	8 by 2	"PRMD" information
Z	20	Floating point information
T	10	Temporary floating point information
KBCD	30	Storage for input
A	S	Matrix entries
IA	S (halfwords)	Matrix entry row numbers
KL	N (halfwords)	Column locators for matrix entries
KN	N	Column names
NR	M	Row names
B	M	Right hand sides
JH	M	Column pivoted for row
KB	N	Row pivoted for column
P	M	Prices

X	M	Solution vector
Y	M	Pivot column expanded
IP	T	Row pivoted for transformation
LE	T (halfwords)	Column locator for transformation entries
IE	L (halfwords)	Transformation entry row numbers
E	1	Transformation entries

Where

S = maximum number of non-zero matrix entries

M = maximum number of rows

N = maximum number of columns

T = maximum number of transformations

L = maximum number of transformation entries

APPENDIX

C. DESCRIPTION OF FIXED POINT INFORMATION IN COMMON

KM(01)	Name associated with matrix
KM(02)	Iteration number
KM(03)	Name associated with right-hand side
KM(04)	Infeasibility flag (non zero if infeasible)
KM(05)	No. 6 output frequency (nominally 10,000)
KM(06)	Maximum number of non-zero coefficient matrix entries
KM(07)	Suboptimization flag (1 implies on) (nominally 1)
KM(08)	Inversion frequency (nominally 10,000)
KM(09)	Cutoff frequency (nominally 10,000)
KM(10)	Name of column removed from basis
KM(11)	Name of pivot row
KM(12)	Name of column which is in "Y" storage or "ERRS"
KM(13)	Name of column brought into basis
KM(14)	Eject control
KM(15)	
KM(16)	Condition number
KM(17)	Number of artificials in basis
KM(18)	Number of matrix entries
KM(19)	Number of transformation entries
KM(20)	Maximum number of rows in coefficient matrix
KM(21)	Maximum number of columns in coefficient matrix
KM(22)	AROW control (non zero if rows have been designated real, (nominally 1)

DESCRIPTION OF FIXED POINT INFORMATION IN COMMON--2

KM(23)	CRSH control (0 = invert, 1 = crash)
KM(24)	E matrix storage exceeded flag (=1 if exceeded)
KM(25)	Iterations since last No. 6 output
KM(26)	Pivot counter
KM(27)	Second best (lowest reduced cost) column for suboptimization
KM(28)	Iterations since last inversion
KM(29)	Number of infeasibilities
KM(30)	Negative X flag (non zero when there are negative X'x)
KM(31)	Exponent of determinant
KM(32)	Iteration number of last inversion
KM(33)	Alphabetic constant "ERRS"
KM(34)	
KM(35)	Inversion type (1=inversion frequency, 2=optimal reinversion, 3=reinversion because of space, 0=all others)
KM(36)	Maximum number of transformations allowed
KM(37)	Number of column rejected if pivot was rejected on this iteration
KM(38)	Alphabetic constant "BEGN"
KM(39)	Alphabetic constant "STOP"
KM(40)	Maximum number of transformation entries allowed
KM(41)	
KM(42)	Pivot step counter
KM(43)	Singleton control (0 = slacks, 1 = singletons)
KM(44)	Number of slacks in inversion
KM(45)	Temporary storage
KM(46)	Temporary storage
KM(47)	Pivot rejection flag
KM(48)	4 EBCDIC blanks
KM(49)	Matrix flag (=0 if not in yet)
KM(50)	

APPENDIX

D. DESCRIPTION OF FLOATING POINT INFORMATION IN COMMON

Z(1)	Pivot tolerance
Z(2)	Set x to zero tolerance
Z(3)	Cost tolerance
Z(4)	Mixed pricing ratio
Z(5)	JMY tolerance, TMLT
Z(6)	Transformation entry tolerance
Z(7)	Pivot rejection tolerance
Z(8)	Current pivot ratio
Z(9)	Mantissa of determinant
Z(10)	Objective value
Z(11)	Time limit in seconds since last resetting
Z(12)	
Z(13)	Minimum reduced cost in non-basic variables
Z(14)	Second lowest reduced cost for suboptimizing
Z(15)	
Z(16)	
Z(17)	Current pivot element
Z(18)	
Z(19)	
Z(20)	

APPENDIX

E. SUMMARY OF SUBROUTINES

Name of Routine	
BOS	Master computing routine
CAP	To input RHS, ALTA, ALTB, OBJ, BASS & AROW data
DEL	Multiplies prices by one column of the matrix
ERR	Calculates errors
FIN	The main routine
GET	Sets up initial conditions for pricing
HOT	Does column output
INP	Reads control cards
JMY	Generates one transformed matrix column'
KRT	Applies transformations to get prices
LOT	Does row output
MIN	Finds columns with minimum reduced costs
NEW	Pivots in singletons and slacks
OUP	Does punch output
PIV	Adds a transformation to the inverse
ROW	Selects the pivot row
SOT	Does short output
SS	Controls optimal reinversion
TAP	Inputs matrix cards
UOUT	Controls output
VER	Does crashing and inverting
XCK	Tests for feasibility and applies X tolerance
YXB	Generates X from RHS and transformations
UTIL	Assembler language utility package including a search routine and part of the timing routine
TIME	Sets up and reads "internal relative timer"

REFERENCES

1. Clasen, R.J., "Linear Programming Routine RS MFOR," SHARE Distribution Agency, #1379. (1962)

2. Clasen, R.J., "Using Linear Programming As a Simplex Subroutine." (The RAND Corporation, P-3267, November, 1965)

3. Cutler, L. and Wolfe, P., "Experiments in Linear Programming," in Graves, R.L. and Wolfe, P., eds., Recent Advances in Mathematical Programming. (New York: McGraw-Hill Book Company, Inc., 1963)

4. Dantzig, G.B., Linear Programming and Extensions. (Princeton, New Jersey: Princeton University Press, 1963)

5. Hadley, G., Linear Programming. (Reading, Mass.: Addison-Wesley Publishing Co., Inc., 1963)

6. Wolfe, P., "The Composite Simplex Algorithm," SIAM Review. Vol. 7, no. 1 (1965)