A COMPUTER PROGRAM FOR QUADRATIC MATHEMATICAL MODELS TO BE USED FOR AIRCRAFT DESIGN AND OTHER APPLICATIONS INVOLVING LINEAR CONSTRAINTS

L. Cutler and D. S. Pass

A Report prepared for

UNITED STATES AIR FORCE PROJECT RAND
This research is supported by the United States Air Force under Project Rand—Contract No. F44620-67-C-0045—Monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of Rand or of the United States Air Force.
A COMPUTER PROGRAM FOR QUADRATIC MATHEMATICAL MODELS TO BE USED FOR AIRCRAFT DESIGN AND OTHER APPLICATIONS INVOLVING LINEAR CONSTRAINTS

L. Cutler and D. S. Pass

A Report prepared for
UNITED STATES AIR FORCE PROJECT RAND
Rand maintains a number of special subject bibliographies containing abstracts of Rand publications in fields of wide current interest. The following bibliographies are available upon request:

Africa • Arms Control • Civil Defense • Combinatorics
Communication Satellites • Communication Systems • Communist China
Computing Technology • Decisionmaking • Delphi • East-West Trade
Education • Foreign Aid • Foreign Policy Issues • Game Theory
Health-related Research • Latin America • Linguistics • Maintenance
Mathematical Modeling of Physiological Processes • Middle East
Policy Sciences • Pollution • Program Budgeting
SIMSCRIPT and Its Applications • Southeast Asia • Systems Analysis
Television • Transportation • Urban Problems • USSR/East Europe
Water Resources • Weapon Systems Acquisition
Weather Forecasting and Control

To obtain copies of these bibliographies, and to receive information on how to obtain copies of individual publications, write to: Communications Department, Rand, 1700 Main Street, Santa Monica, California 90406.
PREFACE

This report describes a computer program, converted from an earlier machine version, that solves quadratic programming problems, i.e., mathematical programming problems involving linear constraints on all variables with a quadratic objective function to be optimized. The development and updating of the program are part of Rand's effort to supply computational tools useful for solving practical problems that arise in a wide variety of Air Force applications.

The motivation for developing this particular program came from the idea of obtaining a least squares smooth fitting to such segments of a three-dimensional surface as may be encountered in fairing of segments of fuselage or wing surfaces. In this case, usually only a finite set of points is given through which the real surface should pass; an approximation for the points of the entire real surface is necessary.

However, the program is general for least squares fitting and the Air Force should find it useful whenever such fits involve additional linear constraints. Many examples for application can be found in such areas as inventory analysis and control, regression analysis (e.g., in reliability studies, such as whether aircraft component failures differ significantly from one aircraft to another), two-dimensional function fitting to weather data, and missile tracking data reduction.

The program, originally written by Leola Cutler for the IBM 7044, is a conversion of QPF4 to the IBM 360 by D. S. Pass.
SUMMARY

This report is a user's guide to a computer program that solves the problem of minimizing a quadratic function of non-negative variables subject to linear constraints. It is a double-precision, all-in-core code, written almost completely in FORTRAN IV for use on the IBM 360/65 computer.

The size of data storage can easily be set by the user. The input data are packed in storage, but the maximum-size problem is usually limited by the size of the inverse of the basis, which is stored in core.

The report describes how the data are input; the symbolic control-card scheme controls the input and the execution of the program. A parametric run on the linear part of the objective function may be run, or part of the input can be entered from tape, or special output options may be desired. A sample problem is given and its output explained. The report includes a listing of the computer program.
ACKNOWLEDGMENTS

We would like to thank the following people, whose contributions made this report possible. Philip Wolfe, formerly of The Rand Corporation, was a co-planner for QPF4, developing the parametric technique used for the linear part of the objective function. The general procedure is essentially by G. B. Dantzig, Ref. 1.

In converting QPF4 to the IBM 360, we adapted many of the subroutines written for MFOR by R. J. Clasen, a Rand consultant.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PREFACE</td>
<td>iii</td>
</tr>
<tr>
<td></td>
<td>SUMMARY</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td>ACKNOWLEDGMENTS</td>
<td>vii</td>
</tr>
<tr>
<td></td>
<td>Section I INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>II. PROGRAM AND INPUT DESCRIPTIONS</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Symbolic Control Card Interpreter</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Row and Column Naming Convention</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Statement of the Problem and the Algorithm</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Quadratic Program Algorithm</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Type 0 Control Cards</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Type 1 Control Cards (Followed by One Data Card)</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Type N Control Cards</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Ordering of the Control Cards</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>III. SAMPLE PROBLEM</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>IV. OUTPUT FOR SAMPLE PROBLEM</td>
<td>26</td>
</tr>
<tr>
<td>Appendix</td>
<td>A. SUMMARY OF CONTROL CARDS</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>B. SUMMARY OF ROUTINES IN QP360</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>C. SUMMARY OF COMMON STORAGE--NOT RELOCATABLE</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>D. SUMMARY OF COMMON STORAGE--RELOCATABLE</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>E. DESCRIPTION OF KM AND KMQ ARRAYS IN COMMON</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>F. FLOATING-POINT VARIABLES</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>G. PROGRAM LISTING WITH INPUT PROBLEMS</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>REFERENCES</td>
<td>113</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

The RS QPF4/360 program solves the problem of minimizing a quadratic function of non-negative variables subject to linear constraints. It is presently a double precision, all-in-core code, written, with one exception, completely in FORTRAN IV. Mnemonic control cards select the paths for the codes.

The program is to be run on an IBM 360 computer and the storage size may be easily set by the compilation of two routines. However, the size of the problem to be solved is limited because the matrix and the inverse of the basis must fit in core.

We assume a familiarity with the jargon used in the "M"-series linear programming codes submitted to SHARE, e.g., RS M1 and RS MFOR.
II. PROGRAM AND INPUT DESCRIPTIONS

SYMBOLIC CONTROL CARD INTERPRETER

All control cards use columns 1 through 6 to determine the nature of the control card. Columns 7 through 80 of the control cards may have any punching. The control cards are of three types, Type 0, Type 1, and Type N.

A Type 0 control card has no data following it.
A Type 1 control card has one data card following it.
A Type N control card has an arbitrary number of data cards following it. These data cards must be blank in columns 1 through 6. The data for a control ends as soon as a card with a punch in columns 1 through 6 occurs. The control card that follows Type N should contain "END" in columns 1 through 6 in order to terminate the data in the proper manner.

All control cards must be punched, starting in column 1, exactly as indicated on the following pages. For example, the control card "EOF" must be punched with "E" in column 1, the letter "O" in column 2, and "F" in column 3. Columns 4 through 6 must be blank.

ROW AND COLUMN NAMING CONVENTION

All row and column names are input as six-character fields. Any EBCDIC character, including blanks, may be used. Names are considered different unless they agree
exactly, character by character, including blanks. A 
name that is used for a column may not be used for a row.

STATEMENT OF THE PROBLEM AND THE ALGORITHM

Minimize the objective function

\[ \lambda p^T x + x^T Q x \]

as a function of \( x \) subject to the linear constraints

\[
\begin{align*}
A_1 x & \leq b_1 \\
A_2 x & \geq b_2 \\
A_3 x & = b_3 \\
x & \geq 0
\end{align*}
\]

where

\( p \) is an \( n \)-component row vector;

\( A_1, A_2, A_3 \) are matrices with \( n \) columns and \( m_1, m_2, m_3 \) rows, respectively;

\( Q \) is a symmetric \( n \times n \) matrix;

\( b_1, b_2, b_3 \) comprise column vectors with \( m_1, m_2, m_3 \) rows, respectively;

\( x \) is an unknown \( n \)-component column vector, whose components are also referred to as the \( x \) variables;

\( \lambda \) is a non-negative scalar. QP360 sets the initial value of \( \lambda, L(0) \), to \( 1.0 \) unless the user specifies this value by a control card.

Each piece of data is identified with a column name and a row name. These names contain six EBCDIC characters, including blanks. The names assigned to \( p \) and to the rows
of $A_1, A_2, A_3$ must be different from the names assigned to their columns. Column names correspond to the $x$ variables. Data are entered by columns. Complete columns are not required and the order within columns is not crucial. Table 1 is a pictorial representation of the symbolic scheme of the input.

The rows of the $Q$ matrix also take the names of the $x$ variables. For example, assume that the quadratic portion of the objective function reads

$$2x_1^2 - 3x_1x_3 + x_2^2 + 5x_2x_1 + \ldots$$

and the names assigned to the columns in the $A$ matrices are $X_1$, $X_2$, and $X_3$. Then, the data for the first two columns of $Q$ could be entered as:

<table>
<thead>
<tr>
<th>Column</th>
<th>Row</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>$X_1$</td>
<td>2</td>
</tr>
<tr>
<td>$X_1$</td>
<td>$X_3$</td>
<td>-3    (Note that $q_{31} = q_{13} = -3/2$)</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$X_2$</td>
<td>1</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$X_1$</td>
<td>5     (Note that $q_{21} = q_{12} = 5/2$)</td>
</tr>
</tbody>
</table>

The column and row names may be interchanged so that, in place of the second line above, $-3x_1x_3$ could be entered with the column for $X_3$ as:

| $X_3$  | $X_1$ | -3    |

Only non-zero values must be entered. For the $Q$ matrix, the user need not enter all of the data since $q_{ij} = q_{ji}$. If $i \neq j$ and values are entered for both terms, QP360 stores
Table 1—Full tableau of data as used internally by QP360

<table>
<thead>
<tr>
<th>Primal variables</th>
<th>Dual variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$x_+$</td>
</tr>
<tr>
<td>$p$</td>
<td></td>
</tr>
<tr>
<td>$A_1$</td>
<td>$I_2$</td>
</tr>
<tr>
<td>$-A_2$</td>
<td>$I_2$</td>
</tr>
<tr>
<td>$A_3$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dual Data</th>
<th>$-Q$</th>
<th>$I$</th>
<th>$-A_1'$</th>
<th>$A_2'$</th>
<th>$-A_3'$</th>
<th>$-p'$</th>
<th>$L(0)$</th>
<th>$p'$</th>
</tr>
</thead>
</table>

- $x$ — Column names of $A$; as input with data
- $x_+$ — Slacks created; column names same as row names for $A_1$ matrix
- $x_-$ — Slacks created; column names same as row names for $A_2$ matrix
- $y$ — Slacks created; column names same as for $x$ variables (or $Q$ rows which are equivalent)
- $y_+$ — Transpose of rows done by code, column names are the same as the original rows
- $y_-$ — Transpose of rows done by code, column names are the same as the original rows
- $\lambda$ — $L(0)$ Column and name are created by the code

ON ALL OUTPUT DUAL VARIABLES ARE IDENTIFIED AS SUCH

$^{a}xy = 0; x_+y_+ = 0; x_-y_- = 0$

$x, x_+, x_-, y, y_+, y_- \geq 0$

$\pi$ are unrestricted variables; may be negative

$^{a}xy = 0; x_+y_+ = 0; x_-y_- = 0$

$x, x_+, x_-, y, y_+, y_- \geq 0$

$\pi$ are unrestricted variables and may be negative.
the sum of the values corresponding to $q_{ij}$ and $q_{ji}$. Thus, the values entered might be $q_{ij}/2$ and $q_{ji}/2$.

The user must provide QP360 with a list of names both for $p$ and for the $A$-matrix rows. These cards include a special character that identifies the type of row.

- for the linear form $p$;
- for rows in $A_1$, maximum constraints $A_1 x \leq b_1$;
- for rows in $A_2$, minimum constraints $A_2 x \geq b_2$;
- blank or zero for rows in $A_3$, equality constraints $A_3 x = b_3$.

QP360 generates the appropriate slack variable for each row of $A_1$ and $A_2$ by using the row name as the name of the slack. The original $x$ variables plus the slacks are the primal variables of the problem. QP360 also forms a set of dual variables. There is an identity matrix associated with $Q$. The names assigned to the columns of $Q$ are the same as the names of the rows. In addition, each row of the $A$ matrices is transposed and takes the name of its column name. QP360 also generates a variable called $L-L(0)$, used for the parametric computation. In effect, the same column name is used in primal and dual variables. On all output, dual variables are identified as such.

**Quadratic Program Algorithm**

1. Find a feasible solution for the primal subproblem:
\[ A_1 x \leq b_1 \]
\[ A_2 x \geq b_2 \]
\[ A_3 x = b_3 \]

2. Extend the basis to include the appropriate dual variables:
   a. For every nonbasic \( x \), insert the \( y \) (slack) variable associated with it.
   b. For every nonbasic \( x_+, x_- \), insert the appropriate \( y_+, y_- \) variable.
   c. Insert all of the \( \pi \) (unrestricted) variables in the basis.

3. Using all of the data, create the inverse and solution for the new basis.

4. Find the optimal solution to the quadratic problem for the initial value \( L(0) \). Maintain all \( x, x_+, x_- \) variables non-negative.
   a. Find \( y_s = \text{minimum } y \) for \( y < 0 \).
      If all \( y \geq 0 \), solution is optimal.
   b. Introduce column \( x_s \) into solution. Save the name of \( y_s \). Choose pivot among the rows that contain \( x, x_+, x_- \) variables and \( y_s \). All other \( y \) variables and \( \pi \) may be negative.
   c. If column \( y_s \) is dropped, go to step 4a and repeat method.
   d. Introduce column \( y_r \) into the solution. If \( y_s \) is dropped from the basis, go to step 4a. If
an \( x_{ri} \) variable is dropped, introduce \( y_{ri} \) into the basis.

5. Parametric run on \( \lambda px + x'Qx \).

\( L(0) \) is the initial value of \( \lambda \). The code sets this to one so that the usual quadratic problem minimizes \( px + x'Qx \). The user has the option to set \( L(0) \) to any value at input time. After the problem is optimal, parametric programming will increase the value of \( \lambda \) and maintain an optimum solution.

a. First step is to introduce the variable \( L-L(0) \).

b. Choose the pivot row so that all \( x \) and \( y \) type variables remain positive or zero.

c. The new column that enters in the basis is the complement of the one that was dropped. Thus, if \( x_{ri} \) is dropped, the new column would be \( y_{ri} \). If \( y_{ri} \) is dropped, then the new column is \( x_{ri} \).

d. Repeat step c until the solution becomes unbounded or the limit for \( \lambda \) has been reached.

**TYPE 0 CONTROL CARDS**

Columns 1 through 6

BEGIN

Function

Is the first card of a run. It zeroes out the data storage area, assigns values to "FREQUE," "TOLERA," "PRMODE," "L(0)," and "LIMIT," then ejects the page for output.
Columns 1 through 6

Function

END

Should be used to terminate the data that follow Type N control cards. It causes no action, but prevents possible format trouble when the next control card is read. Leave columns 7 through 80 blank.

EOF

Instructs the code to read from the standard FORTRAN input-unit 5. Code will continue to read this unit until it receives the control card "TAPE" and switches to FORTRAN logical-unit 11. The last card on unit 11 should be an "EOF" control card.

ERRORS

Calls the subroutine ERR to compute the row errors. The row error is the total amount by which the variables fail to satisfy the equations. The subroutine computes the sum of the absolute values of the errors and also retains the largest absolute error and the row in which it occurs. ERR prints these three pieces of information.

GO

Calls BOS, which controls the calculations for the basic changes needed to reach the optimal solution for the original problem.
Columns 1 through 6

Function
or the solutions for new values of \( \lambda \) in the parametric mode. Normally, this control is used only after INVERT.

INVERT

Calls VER to create the inverse for the current basis.

OUTPUT

Calls UOUT to output for condition 6.

PARAME

(PARAMETRIC) Sets \( KMQ(11) = -1 \), then calls BOS. The code introduces the column \( L-L(0) \) into the solution. It makes basis changes that maintain an optimum solution and increase the value of \( L-L(0) \). It continues to do so until \( \lambda = (L-L(0) + L(0) \) reaches the LIMIT that was set or becomes unbounded.

PUNCH

Calls OUP to punch out cards for restart purposes. It punches the control card BASIS* and then cards for the basis and artificial rows, if any. If the subprogram is feasible, another card is punched. It contains the names of the variables in \( KMQ(14) \) and \( KM(15) \) and a word to tell the code if it is looking for the optimal solution or is in the parametric mode. The last card is the END control card.
<table>
<thead>
<tr>
<th>Columns 1 through 6</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETINV</td>
<td>Sets the inversion frequency to ((m/10) + 6), our normal recommendation, where (m) is the number of rows in the full problem. Use this control card only after the matrix data have been read.</td>
</tr>
<tr>
<td>SOLVE</td>
<td>Is the normal way to start a problem. The code sets (KM(23) = 1), then calls VER, which performs a crash, i.e., enters as many vectors as possible into the basis, ignoring feasibility. After VER has created the inverse, the code calls BOS.</td>
</tr>
<tr>
<td>TAPE</td>
<td>Instructs the code to read all input from FORTRAN-unit 11. It continues to do so until it encounters an EOF control card.</td>
</tr>
<tr>
<td>EXIT</td>
<td>Instructs the code to eject a page and exit.</td>
</tr>
<tr>
<td>(blank card)</td>
<td>If a control card has no punching, i.e., all blanks -- in columns 1 through 6, QP360 prints an error message and then skips to the next BEGIN card.</td>
</tr>
<tr>
<td>(comment card)</td>
<td>If columns 1 through 6 contain nonblank punching that is different from the acceptable set of controls, the card is treated</td>
</tr>
</tbody>
</table>
Columns 1 through 6 as a comment card. Columns 1 through 6 of this card are stored in KM(41). When a "MATRIX" or "RHS" control is read, the information in KM(41) identifies it so that the comment card that precedes "MATRIX" or "RHS" (if any) is used for identification.

TYPE 1 CONTROL CARDS (FOLLOWED BY ONE DATA CARD).

L(0) Reads a data card that has a starting value for \( \lambda \) in \( \lambda px \) of the objective. The code initially sets the value of \( \lambda \) to be 1.0. The data card is blank in columns 19 through 30. QP360 reads it with an F12.6 format and assumes that the decimal point is between columns 24 and 25 if none is punched. (The third letter in the control card may be either a zero or the letter 0).

LIMIT Reads a data card that sets the upper limit for the value of \( \lambda \) in \( \lambda px \) of the objective function. The code initially sets the limit to be 1.0E20. The data card format is the same as for L(0).

PRMODE The associated data card determines the kind of output QP360 will produce for each of the 10
possible conditions that call for output. The kind of output is defined by an integer value from 0 to 9, punched in the first 10 columns. A blank is treated as zero. The conditions correspond to card columns 1 through 10. The following lists describe the conditions, the values set initially by QP360, and the kinds of output called for by each value.

<table>
<thead>
<tr>
<th>Condition or Column</th>
<th>Occurrence of Output</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>After every pivot step, except during parametric runs.</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>When feasibility is declared.</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>When an optimum solution is declared.</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>When it is determined that no feasible solution exists.</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>When infinite solution is found.</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>When &quot;OUTPUT&quot; control card is read or when code reaches the frequency specified by the &quot;OUTPUT&quot; data card read by &quot;PREQUE&quot; control card.</td>
<td>6</td>
</tr>
<tr>
<td>Condition or Column</td>
<td>Occurrence of Output</td>
<td>Initial Value</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>7</td>
<td>When inverse is too big for core. After this output is done, the job skips to the next &quot;BEGIN&quot; card.</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>When cut off because time or iteration limit is exceeded.</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>After every pivot step during the parametric run.</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>When the limit for ( \lambda ) has been reached or exceeded.</td>
<td>9</td>
</tr>
</tbody>
</table>

The meaning of the punched values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No output</td>
</tr>
<tr>
<td>1</td>
<td>Short output only</td>
</tr>
<tr>
<td>2</td>
<td>Short and row output</td>
</tr>
<tr>
<td>3</td>
<td>Short and primal-dual output</td>
</tr>
<tr>
<td>4</td>
<td>Short, row, and primal-dual output</td>
</tr>
<tr>
<td>5</td>
<td>Card output for BASIS*</td>
</tr>
<tr>
<td>6</td>
<td>Short output and BASIS*</td>
</tr>
<tr>
<td>7</td>
<td>Short and row output and BASIS*</td>
</tr>
<tr>
<td>8</td>
<td>Short and primal-dual output and BASIS*</td>
</tr>
<tr>
<td>9</td>
<td>Short, row, and primal-dual output and BASIS*</td>
</tr>
</tbody>
</table>
TYPE N CONTROL CARDS

ALTERA  Changes data in the matrix. Each change is entered, one per card, in the same format as matrix data. This control cannot be used to add a new entry to the matrix. The run is skipped if the data have not originally been loaded with the matrix. Of course, a value of zero may be used as a new entry or as an altered entry.

ALTERB  Changes or adds a new entry in the right-hand side (RHS). Each change is entered, one per card, in the same format as RHS data.

BASIS    Enters the names of primal basis variables and artificial rows, if any. The code extends the basis on the assumption that the solution is feasible. The matrix data must be read before the basis can be entered. The code deletes any basis variable that is not in the matrix. INVERT should be the first computational control card for the run. This control is used if a basis is known. Basis column names are entered from 1 to 10 per card in the following format:
Columns 1-6   Blank.
Columns 7-12   Name of column for the
13-18   basis punched exactly as
   :    it appears in the matrix;
   61-66   or a blank field.
Columns 67-72   Blank.

Artificial row names are entered from 1 to 10
per card in the following format:

Columns 1-6   Blank.
Columns 7-12   Name of row punched exactly
13-18   as it appears in the
   :    matrix; or a blank
   61-66   field.
Columns 67-72   "ARTIF." These six characters must
   must be punched in order to
   identify the card as containing
   artificial rows.

BASIS*

This control plus all of the data and the "END"
card are punched by the code when the "PUNCH"
control card is used or when a condition calls
for output with a value of five or greater.
The data cards are the list of basis column
names with a marker of "..DUAL" to identify
dual variables. The cards include artificial
rows, if any, and special cards if the
solution is feasible. These cards are a card with "DUAL.." in columns 67 through 72 and information for KMQ(14) and KM(15), needed when looking for an optimal solution. If the code is operating in the parametric mode, columns 67 through 72 will contain "PARAM." This package of data is used to restart a problem from the point where it stopped.

(FREQUENCY OR FREQUENCIES) This control card changes either the special-output frequency, the matrix-inversion frequency, or the maximum number of iterations. QP360 initializes each of these values to 10,000. The card format requires that columns 1 through 6 be blank, columns 7 through 12 contain the name of the variable, and columns 13 through 20 contain the integer value, right justified. The frequencies and their meanings are:

<table>
<thead>
<tr>
<th>Column 7 through 12</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>Frequency for calling condition; 6 special output.</td>
</tr>
<tr>
<td>INVERT</td>
<td>Reinversion frequency.</td>
</tr>
<tr>
<td>CUTOFF</td>
<td>Maximum number of iterations allowed for the matrix.</td>
</tr>
</tbody>
</table>
The Type 0 control card, "SETINV," and the "INVERT" data card set the same frequency. The last one read is retained.

The output frequency may not occur precisely on the iteration as specified by the "OUTPUT" data card.

If the inverse exceeds storage area, the code transfers to reinvert the solution before it tallies for output.

**MATRIX**

Each data card contains one matrix entry in the following format:

- **Columns 1-6**  Blank.
- **Columns 7-12** Column name.
- **Columns 13-18** Row name.
- **Columns 19-30** Numerical entry in FORTRAN FL2.6 format. The code assumes the decimal point is between columns 24 and 25 if none is punched.

All entries with the same column name must be adjacent in the deck. No restriction is made on the ordering of the row names. A data entry of zero is usually omitted; however, zero or non-zero values must be input for a
position that is going to be changed later by using the "ALTERA" control card.

**RHS**

This control card zeroes out the RHS vector and then reads the data cards. Each card contains one non-zero RHS entry. The card format is the same as for matrix data; the column-name field is ignored.

**ROWS**

Each data card contains the name of a row and a special punch to indicate if the row is the linear form p or belongs to the $A_1$, $A_2$, or $A_3$ matrix. (Row names for the Q matrix are not entered.)

<table>
<thead>
<tr>
<th>Columns</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-11</td>
<td>Blank</td>
</tr>
<tr>
<td>12</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>$+$</td>
</tr>
<tr>
<td></td>
<td>$-$</td>
</tr>
<tr>
<td>13-18</td>
<td>Name of row</td>
</tr>
</tbody>
</table>
TOLERA (TOLERANCE or TOLERANCES) Reads data cards that input floating-point tolerances. Columns 1 through 6 are blank, columns 7 through 12 contain the name of the tolerance, and columns 13 through 20 contain its numerical value, which is read in with a format of E8.1.

<table>
<thead>
<tr>
<th>Columns 7 through 12</th>
<th>Meaning</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIVOT</td>
<td>Minimum element size considered for a pivot.</td>
<td>1.0E-05</td>
</tr>
<tr>
<td>REJECT</td>
<td>Pivot rejection tolerance.</td>
<td>1.0E-03</td>
</tr>
<tr>
<td>COST</td>
<td>Reduced cost considered -1.0E-05 negative if below this value.</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>Tolerance for setting x equal to zero.</td>
<td>1.0E-05</td>
</tr>
<tr>
<td>ENTRY</td>
<td>Minimum element size in new transformation.</td>
<td>1.0E-07</td>
</tr>
<tr>
<td>TMULT</td>
<td>Minimum element size considered for applying transformation.</td>
<td>1.0E-10</td>
</tr>
</tbody>
</table>

ORDERING OF THE CONTROL CARDS

Runs may be stacked in QP360. A run is defined as the total computation for one set of matrix data, including any changes made to the matrix by "ALTERA" data cards. The "BEGIN" control card denotes a new run.
The "ROWS" control card and data must be read before "MATRIX" or "RHS" may be used. The ordering of "RHS" and "MATRIX" in the deck is arbitrary.

"ALTERB" may be used only after the RHS data has been loaded.

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

ALTERA
BASIS
BASIS*
ERRORS
GO
INVERT

OUTPUT
PARAMETRIC
PUNCH
SETINV
SOLVE

The "L(0)" control should be loaded before "INVERT" or "SOLVE" is used.

All other control cards may be read at any time.

The "BEGIN" control resets data to initial values; therefore, the following controls would have to be reloaded if special values are used:

FREQUE
L(0)
LIMIT

PRMODE
TOLERA

Control cards "TAPE," "EOF," and comment cards may be used at any time.

The standard problem is one that uses all parameters and tolerances as initialized by the program. The data and
control card order for this problem would be:

BEGIN
ROWS
data
END

comment card for identification of the matrix
(optional)

MATRIX
data
END

comment card for identification of the RHS
(optional)

RHS
data
END

SOLVE
III. SAMPLE PROBLEM

Solve the quadratic programming problem

Minimize \( 2x_1 - 3x_2 + 2x_3 + [20x_1^2 + 5x_2^2 + 5.5x_3^2 + x_2x_3] \)

under the constraints

\[ \begin{align*}
5x_1 & = 2x_3 \leq 2 \\
4x_1 + 2x_2 + x_3 & \geq 3 \\
2x_1 + x_2 + x_3 & = 2
\end{align*} \]

For this problem

\[ p = (2, -3, 2) \]

and

\[ Q = \begin{bmatrix}
20 & 0 & 0 \\
0 & 5.0 & 0.5 \\
0 & 0.5 & 5.5
\end{bmatrix} \]

where \( p \) is the linear part and \( Q \) is the quadratic part of the objective function. The objective form in general notation is \( \lambda p x + x'Q x \). The solution is needed for all combinations of this form. Therefore, the initial solution is found for \( \lambda = 0 \); then the parametric feature increases the value of \( \lambda \). (The standard mode of operation would be for the program to initialize \( \lambda \) to equal one and solve the problem for \( px + x'Q x \).)

The card input and the machine output for the problem are given in Fig. 1. The names assigned to the rows and columns are:

\[ \begin{array}{ll}
\text{LINEAR} & \text{for the linear form } p, \\
\text{MAXIM} & \text{for the first constraint.}
\end{array} \]
BEGIN
PRM0DE USE 002222234 FOR PRINT CONTROL
002222234
SAMPLE PROBLEM FROM RSEQ1 OPTTEST PROB 2
ROWS
  EQUAL*
+MAXIM
$LINEAR
-MIN
END
PROB 2
RHS
  MAXIM  2.0
  MIN    3.0
  EQUAL* 2.0
END
OPTTEST
MATRIX
  X1  LINEAR  2.0
  X1  MAXIM   5.0
  X1  MIN    -4.0
  X1  EQUAL*  2.0
  X1  X1     20.0
  X2  LINEAR -3.0
  X2  MIN    2.0
  X2  EQUAL*  1.0
  X2  X2     5.0
  X3  LINEAR  2.0
  X3  MAXIM  -2.0
  X3  MIN    1.0
  X3  EQUAL*  1.0
  X3  X2     1.0
  X3  X3     5.5
END
SETINV
L(0)
SET L(0) TO ZERO
0.0
SOLVE
PARAMETRIC

Fig. 1--Input to Sample Problem
MIN for the second constraint.
EQUAL* for the third constraint.
X1 for variable x₁.
X2 for variable x₂.
X3 for variable x₃.

The rows of the Q matrix take the variable names X1, X2, and X3. Note that the entry for the X₂X₃-term is input with a coefficient of 1.0 for the variable (column) X3 and row X2.
IV. OUTPUT FOR SAMPLE PROBLEM

Figure 2 contains the output from the sample problem as written on output-unit 6. In some instances, two pages of output have been combined into one page in order to conserve space.

The control cards -- except for the END cards -- and comments cards are listed. Note that the field beyond column 6 of the control card may be used for comments. Columns 1 through 6 of the comment that precedes the RHS have been saved for its identification. Columns 1 through 6 of the comment card that precedes the MATRIX have been saved for MATRIX identification. In this problem, the matrix is called OPTEST, and the RHS is called PROB 2. The statement following MATRIX gives the dimensions that are set after the matrix data are read. In this case, 9 rows have been reserved. The first 2 are used by the program. The next row contains the linear portion of the objective function; the next 3 are the primal constraints; the last 3 contain 3 columns for the quadratic portion of the objective function. The 3 columns are equal to the 3 input variables and the 15 matrix entries are the 15 pieces of data that were entered.

The message after SOLVE indicates that the inversion routine has been entered and the subsequent one gives the result after inversion. In this case, inversion has
BEGIN
PRMODE USE 0022222234 FOR PRINT CONTROL
SAMPLE PROBLEM FROM RSQPI QPTEST PROB 2
ROWS PROB 2 RHS QPTEST MATRIX

PROBLEM HAS 9 ROWS, 3 COLUMNS, AND 15 MATRIX ENTRIES.
SETINV L(0) SET L(0) TO ZERO
SOLVE

REINVERTING AFTER ITERATION 0, 0 TRANSFORMATIONS WITH 0 ENTRIES, TYPE 0
*INVERSION COMPLETED 2 SLACKS, 0 POOR COLS, 1 TRANSFORMATIONS WITH 4 ENTRIES

*FEASIBLE AT ITERATION 1, 1 STEPS

REINVERTING AFTER ITERATION 1, 2 TRANSFORMATIONS WITH 8 ENTRIES, TYPE 0
*INVERSION COMPLETED 2 SLACKS, 0 POOR COLS, 4 TRANSFORMATIONS WITH 23 ENTRIES

MAXIMUM ERROR IS ON ROW = -7.110-15
SUM OF ABSOLUTE ERRORS ON ROWS = 1.270-14

Fig. 2---Sample Problem Output
=OPTIMAL SOLUTION

(3) MATRIX R.H.S. ITER PIVS 0 INFEAS DETERMINANT NEW COLUMN OLD COLUMN PIVOT ROW
QPTEST PROB 2 3 8 0.0 1.1960E3 MAXIM MAXIM .. DUAL XI

OBJECTIVE TOTAL = 7.290970 = L * LINEAR (PX) + QUADRATIC (X'QX)

0.0 * -0.073579 + 7.290970

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>ROW</th>
<th>RHS</th>
<th>MAXIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>0.073579</td>
<td>LINEAR</td>
<td>0.0</td>
<td>0.174892</td>
</tr>
<tr>
<td>MIN</td>
<td>0.364548</td>
<td>MAXIM</td>
<td>2.000000</td>
<td>0.113420</td>
</tr>
<tr>
<td>X3</td>
<td>0.397993</td>
<td>MIN</td>
<td>-3.000000</td>
<td>0.216450</td>
</tr>
<tr>
<td>MAXIM</td>
<td>0.602007</td>
<td>EQUAL*</td>
<td>2.000000</td>
<td>-0.216450</td>
</tr>
<tr>
<td>X2</td>
<td>1.381271</td>
<td>XI</td>
<td>0.0</td>
<td>-1.035498</td>
</tr>
<tr>
<td>X3</td>
<td>0.668896</td>
<td>X2</td>
<td>0.0</td>
<td>-0.010390</td>
</tr>
<tr>
<td>EQUAL*..DUAL</td>
<td>-7.290970</td>
<td>X3</td>
<td>0.0</td>
<td>0.320346</td>
</tr>
</tbody>
</table>

MAXIMUM ERROR IS ON ROW = -7.11D-15
SUM OF ABSOLUTE ERRORS ON ROWS = 1.33D-14

EXIT

Fig. 2--Continued
(9) MATRIX R.H.S.   ITER  PIVS
QPTEST PROB 2   4   9
0 INFEAS  DETERMINANT  NEW COLUMN  OLD COLUMN  PIVOT ROW
       0.0  2.56000E 2  L-L(0) . DUAL  X3  EQUAL*

OBJECTIVE TOTAL = 1,210937 = L * LINEAR (PX) + QUADRATIC (X'QX)
2.812500 * -4.250000 + 13.164062

PRIMAL VARIABLES SOLUTION
NAME   VALUE
X1     0.218750
X2     1.562500
MAXIM  0.906250
MIN    1.000000

DUAL VARIABLES SOLUTION
NAME   VALUE
EQUAL* -7.187500
L-L(0)  2.812500

(9) MATRIX R.H.S.   ITER  PIVS
QPTEST PROB 2   5   10
0 INFEAS  DETERMINANT  NEW COLUMN  OLD COLUMN  PIVOT ROW
       0.0  8.000000E 0  X3  . . DUAL  X1  MAXIM

OBJECTIVE TOTAL = -10.000000 = L * LINEAR (PX) + QUADRATIC (X'QX)
5.000000 * -6.000000 + 20.000000

PRIMAL VARIABLES SOLUTION
NAME   VALUE
X2     2.000000
MAXIM  2.000000
MIN    1.000000

DUAL VARIABLES SOLUTION
NAME   VALUE
X3     7.000000
EQUAL* -5.000000
L-L(0)  5.000000

Fig. 2--Continued
=FINAL STEP, PARAMETRIC VARIABLE L IS UNBOUNDED

<table>
<thead>
<tr>
<th>(5) MATRIX R.H.S.</th>
<th>ITER</th>
<th>PIVS</th>
<th>0 INFEAS</th>
<th>DETERMINANT</th>
<th>NEW COLUMN</th>
<th>OLD COLUMN</th>
<th>PIVOT ROW</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPTEST</td>
<td>PROB 2</td>
<td>5</td>
<td>10</td>
<td>0.0</td>
<td>8.00000E0</td>
<td>X1</td>
<td>X1</td>
</tr>
</tbody>
</table>

OBJECTIVE TOTAL = L * LINEAR (PX) + QUADRATIC (X'QX)

-10.000000 = 5.000000 * -6.000000 + 20.000000

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>ROW</th>
<th>RHS</th>
<th>X1</th>
<th>..DUAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>X3 ..DUAL</td>
<td>6.000000</td>
<td>LINEAR</td>
<td>0.0</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>7.000000</td>
<td>MAXIM</td>
<td>2.000000</td>
<td>-0.625000</td>
<td></td>
</tr>
<tr>
<td>L-L(0) ..DUAL</td>
<td>1.000000</td>
<td>MIN</td>
<td>-3.000000</td>
<td>-0.000000</td>
<td></td>
</tr>
<tr>
<td>MAXIM</td>
<td>5.000000</td>
<td>EQUAL*</td>
<td>2.000000</td>
<td>-0.125000</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>2.000000</td>
<td>X1</td>
<td>0.0</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>EQUAL* ..DUAL</td>
<td>-5.000000</td>
<td>X3</td>
<td>0.0</td>
<td>-0.375000</td>
<td></td>
</tr>
</tbody>
</table>

(5) MATRIX R.H.S. | ITER | PIVS | 0 INFEAS | DETERMINANT | NEW COLUMN | OLD COLUMN | PIVOT ROW |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>QPTEST</td>
<td>PROB 2</td>
<td>5</td>
<td>10</td>
<td>0.0</td>
<td>8.00000E0</td>
<td>X1</td>
<td>X1</td>
</tr>
</tbody>
</table>

OBJECTIVE TOTAL = L * LINEAR (PX) + QUADRATIC (X'QX)

-10.000000 = 5.000000 * -6.000000 + 20.000000

<table>
<thead>
<tr>
<th>PRIMAL VARIABLES SOLUTION</th>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>2.000000</td>
<td></td>
</tr>
<tr>
<td>MAXIM</td>
<td>2.000000</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>1.000000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DUAL VARIABLES SOLUTION</th>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X3</td>
<td>7.000000</td>
<td></td>
</tr>
<tr>
<td>EQUAL*</td>
<td>-5.000000</td>
<td></td>
</tr>
<tr>
<td>L-L(0)</td>
<td>5.000000</td>
<td></td>
</tr>
</tbody>
</table>

MAXIMUM ERROR IS ON ROW = -6.440-15
SUM OF ABSOLUTE ERRORS ON ROWS = 9.450-15

PARAMETRIC

Fig. 2--Continued
performed a crash, that is, searched for a starting basis. It installed two slack variables and one regular column and generated a total of four entries for the inverse. Type 0 is the usual type for inversion.

The significance of the other types of inversions is explained in Appendix E by example KM(35). The "poor column" count during an inversion of a given basis is the number of columns for which all possible pivot entries are below the pivot tolerance. These columns are removed from the basis.

After the inversion information, the iteration on which the problem became feasible is printed, followed by the number of pivot steps taken. Following the feasible solution, the program calls inversion to extend the basis to include the dual variables that may be entered. The errors are computed and printed. See the ERRORS control card, p. 9, for an explanation.

The next output is for the optimal solution to the initial objective form. The first line is the terminating condition, in this case "OPTIMAL SOLUTION." Following this is the "short output," beginning with the number "(3)" and ending with "PIVOT ROW." The (3) is the output condition number (see "PRMODE"). The next two columns are the identification names. We then state that the output is after three iterations and eight pivots (including those taken in inversion). There are no infeasibilities in the result.
The determinant of the basis is listed. The new column MAXIM was the last column introduced into the basis. It replaced the dual of MAXIM and is located in row X1. MAXIM is the slack variable associated with the constraint named MAXIM. MAXIM.. DUAL is the dual column created by transposing the constraint MAXIM. The row X1 is the first row of the dual portion of the constraints.

The next headings and values are the components that make up the objective function. The total objective is equal to $\lambda$ times px plux x'Qx. In this run, $\lambda$ is equal to zero. The value for the linear part of the objective is -0.073579 (see p. 4). The value for the quadratic term is 7.290970. Since $\lambda$ is zero for this problem, we found the solution for the minimum of the quadratic portion of the objective. The first five lines of the output constitute the "short output."

Following this is the "row" output. The column headed "NAME" lists the variables that are in the solution (basis). The next column, "VALUE," is the value of each variable. In this problem, the optimal basis contains the primal variables X1, X2, X3, the slacks for the rows MAXIM and MIN, and the dual variable EQUAL*, which was obtained by transposing the constraint EQUAL*. Note that this variable is equal to -7.290970. A negative solution is permissible since the transposition of an equality constraint becomes an unrestricted variable and may be negative. The column labelled
"ROW" gives the names of all the rows of the problem. The column "RHS" lists the RHS value for each of the rows. The program has changed the sign for the row MIN. The last column, "MAXIM," is merely the transformation of the column and may be ignored.

The errors are computed and printed after the optimal solution. Normally, they would start a new page. The next control card is "PARAMETRIC." The "primal-dual" type of output is printed for each iteration during the parametric run. The short output is the same as for the row output. Note that after one parametric step, \( \lambda \) changes from 0 to 2.812500. The variables in the solution are printed in two sections. On the left half of the page, the primal variables are listed. The dual variables are on the right half. The dual variable "L-L(0)" is created by the program for use in the parametric computation. It is equal to the current value of \( \lambda \) less the value of L(0), which is zero in this problem.

After iteration 5, the variable \( \lambda \) is unbounded. This is the normal termination for a parametric run if the control card "LIMIT" has not been used. At this termination, the program automatically does row output, primal-dual output, and punch cards.

The last column of the row output, in this example, "X1 ..DUAL", may be used to calculate the solution for larger values of \( \lambda \). Let \( \Theta \) equal the desired value of \( \lambda \).
less the current value of \( \lambda \). The new solution values are obtained by adding 0 times "X1 .. DUAL" to the column "VALUE." In this problem and in most cases, there will be no change in the value of the primal variables. This is because the linear part of the objective is at its minimum value.

The cards' output is the BASIS* card, which contains the identification for the matrix and the RHS, the iteration number, and the condition number. The next two cards are the list of variables in the basis. The next card contains information for the parametric run. The last card is an "END" card.

The cards are:

\begin{verbatim}
BASIS*  QPTEST  PROB 2  ITERATION  5  (5)
X3 ..DUALMIN
L-L(0) ..DUALMAXIM  X2
EQUAL*..DUAL
PARAM
END
\end{verbatim}

A primal variable is followed by a blank field, a dual variable is followed by ..DUAL.
Appendix A

SUMMARY OF CONTROL CARDS

<table>
<thead>
<tr>
<th>Col(s)</th>
<th>Type</th>
<th>Function</th>
<th>Format of Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-6</td>
<td>N</td>
<td>Change entries in matrix</td>
<td>(6X,A6,A6,F12.6)</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Change entries in right-hand side</td>
<td>(12X,A6,F12.6)</td>
</tr>
<tr>
<td>BASIS</td>
<td>N</td>
<td>Names of columns in basis</td>
<td>(6X,11A6)</td>
</tr>
<tr>
<td>BASIS*</td>
<td>N</td>
<td>Basis that was punched by code</td>
<td>(6X,11A6)</td>
</tr>
<tr>
<td>BEGIN</td>
<td>O</td>
<td>Zero data-core area, sets nominal</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>values</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>O</td>
<td>No action, should be used</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>to terminate data for type N control</td>
<td></td>
</tr>
<tr>
<td>EOF</td>
<td>O</td>
<td>Resume inputting from the</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>system input unit (5)</td>
<td></td>
</tr>
<tr>
<td>ERRORS</td>
<td>O</td>
<td>Calculate and output errors</td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>O</td>
<td>Terminate program normally</td>
<td></td>
</tr>
<tr>
<td>FREQUE</td>
<td>N</td>
<td>Fixed-point frequencies</td>
<td>(6X,A6,I8)</td>
</tr>
<tr>
<td>GO</td>
<td>O</td>
<td>Continue computation from current</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>basis</td>
<td></td>
</tr>
<tr>
<td>INVERT</td>
<td>O</td>
<td>Invert current basis</td>
<td></td>
</tr>
<tr>
<td>L(O)</td>
<td>1</td>
<td>Read value for ( \lambda ), i.e.,</td>
<td>(13X,F12.6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L(O)</td>
<td></td>
</tr>
<tr>
<td>Cols 1-6</td>
<td>Type</td>
<td>Function</td>
<td>Format</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>LIMIT</td>
<td>1</td>
<td>Read value for the limit for λ. Stop run when this value is exceeded.</td>
<td>(18X,F12.6)</td>
</tr>
<tr>
<td>MATRIX</td>
<td>N</td>
<td>Read matrix entries, one per card.</td>
<td>(6X,A6,A6,F12.6)</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>O</td>
<td>Print output using condition 6 in UOUT.</td>
<td></td>
</tr>
<tr>
<td>PARAME</td>
<td>O</td>
<td>Start parametric computation.</td>
<td></td>
</tr>
<tr>
<td>PRMODE</td>
<td>1</td>
<td>Output controls.</td>
<td>(10I1)</td>
</tr>
<tr>
<td>PUNCH</td>
<td>0</td>
<td>Punch BASIS* cards.</td>
<td></td>
</tr>
<tr>
<td>RHS</td>
<td>N</td>
<td>RHS entries.</td>
<td>(6X,A6,A6,F12.6)</td>
</tr>
<tr>
<td>ROWS</td>
<td>N</td>
<td>List of row names with identifiers for types of constraint.</td>
<td>(6X,2A6)</td>
</tr>
<tr>
<td>SETINV</td>
<td>0</td>
<td>Set inversion frequency.</td>
<td></td>
</tr>
<tr>
<td>SOLVE</td>
<td>0</td>
<td>Find a starting basis, then optimize.</td>
<td></td>
</tr>
<tr>
<td>TAPE</td>
<td>0</td>
<td>Begin inputting from FORTRAN-unit 11.</td>
<td></td>
</tr>
<tr>
<td>TOLERA</td>
<td>N</td>
<td>Tolerances.</td>
<td>(6X,A6,E8.1)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Blank control word is an error; skip to next BEGIN card.</td>
<td></td>
</tr>
<tr>
<td>All Others</td>
<td>0</td>
<td>Comment card.</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B
SUMMARY OF ROUTINES IN QP360

<table>
<thead>
<tr>
<th>Routine Name</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>Computes the number of storage words from a given variable location to the end of common storage.</td>
</tr>
<tr>
<td>BOS</td>
<td>Master computing routine.</td>
</tr>
<tr>
<td>CAP</td>
<td>Inputs RHS, ALTEA, ALTERB, BASIS, BASIS*, and ROWS data.</td>
</tr>
<tr>
<td>DEL</td>
<td>Multiplies prices by one column of the matrix.</td>
</tr>
<tr>
<td>ERR</td>
<td>Calculates errors by multiplying matrix by the solution.</td>
</tr>
<tr>
<td>EXP</td>
<td>Picks up all entries of a column and stores expanded column in Y for use by JMY. (Internally, the code stores only half of the Q matrix and the original primal constraints. This subroutine generates the other half of Q when needed and also finds the transpose of a row.)</td>
</tr>
<tr>
<td>(MAIN)</td>
<td>The main routine; relocates input data.</td>
</tr>
<tr>
<td>GET</td>
<td>Sets up initial conditions for pricing.</td>
</tr>
<tr>
<td>Routine Name</td>
<td>Use</td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
</tr>
<tr>
<td>IMAGE</td>
<td>Finds two words of four Hollerith characters each in a list and stores them away.</td>
</tr>
<tr>
<td>INP</td>
<td>Reads the control cards and calls the appropriate routine.</td>
</tr>
<tr>
<td>JMY</td>
<td>Generates one transformed matrix column.</td>
</tr>
<tr>
<td>KRT</td>
<td>Applies transformations to get prices.</td>
</tr>
<tr>
<td>LOT</td>
<td>Does row output.</td>
</tr>
<tr>
<td>MIN-</td>
<td>Finds column with minimum reduced cost.</td>
</tr>
<tr>
<td>OUP</td>
<td>Punches BASIS* data cards.</td>
</tr>
<tr>
<td>PDOT</td>
<td>Prints primal and dual solutions.</td>
</tr>
<tr>
<td>PIV</td>
<td>Adds a transformation to the inverse.</td>
</tr>
<tr>
<td>QOT</td>
<td>Controls output.</td>
</tr>
<tr>
<td>ROW</td>
<td>Selects pivot row.</td>
</tr>
<tr>
<td>SEARCH</td>
<td>Matches two, four-Hollerith-character words with a given list, exits with their position in the list. The two words are considered as a string of eight Hollerith characters.</td>
</tr>
<tr>
<td>SOT</td>
<td>Does short output.</td>
</tr>
<tr>
<td>TAP</td>
<td>Inputs matrix data.</td>
</tr>
<tr>
<td>Routine Name</td>
<td>Use</td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
</tr>
<tr>
<td>TAPENO</td>
<td>Assigns logical unit numbers to the input, output, and reserved input units.</td>
</tr>
<tr>
<td>UOUT</td>
<td>Calls QOT for outputting.</td>
</tr>
<tr>
<td>VER</td>
<td>Finds a starting basis, inverts the current basis, or extends the basis from a primal to dual one and then inverts.</td>
</tr>
<tr>
<td>XCK</td>
<td>Tests for feasibility and applies X tolerance.</td>
</tr>
<tr>
<td>IGETAD</td>
<td>360 Assembly-language coded, FORTRAN callable, function routine that obtains the core address of the argument.†</td>
</tr>
<tr>
<td>NOP</td>
<td>Dummy routine.</td>
</tr>
<tr>
<td>TIE</td>
<td>Dummy routine.</td>
</tr>
<tr>
<td>ENDTIE</td>
<td>This routine is called if an error occurs. The program exits.</td>
</tr>
</tbody>
</table>

(MAIN) is the entry to the program.

†IGETAD is a Rand version of an extended FORTRAN function. The assembly cards are included in the listing. (IGETAD is used in MAIN and BEGIN.)
Appendix C

SUMMARY OF COMMON STORAGE—NOT RELOCATABLE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Dimension</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>19</td>
<td>Location of relocatable symbols A, B, E, IA, IP, JH, KB, KL, KN, LE, NR, P, X, Y, KN2, NR2 with respect to A.</td>
</tr>
<tr>
<td>M</td>
<td>1</td>
<td>Number of rows in problem.</td>
</tr>
<tr>
<td>MA</td>
<td>1</td>
<td>Marker for KM(13) new column.</td>
</tr>
<tr>
<td>MB</td>
<td>1</td>
<td>Marker for KM(10) old column.</td>
</tr>
<tr>
<td>MC</td>
<td>1</td>
<td>Dual marker for KM(12) l, dual. 0, nondual.</td>
</tr>
<tr>
<td>MD</td>
<td>1</td>
<td>Not used.</td>
</tr>
<tr>
<td>ME</td>
<td>1</td>
<td>Number of transformations.</td>
</tr>
<tr>
<td>MF</td>
<td>1</td>
<td>Number of the row that is the first constraint.</td>
</tr>
<tr>
<td>N</td>
<td>1</td>
<td>Number of columns.</td>
</tr>
<tr>
<td>KM</td>
<td>50</td>
<td>Fixed point and EBCDIC information.</td>
</tr>
<tr>
<td>KM2</td>
<td>50</td>
<td>If KM(I) contains EBCDIC information, then KMZ(I) contains the last two EBCDIC characters.</td>
</tr>
<tr>
<td>KP</td>
<td>16</td>
<td>&quot;PRMODE&quot; information.</td>
</tr>
<tr>
<td>Z</td>
<td>20</td>
<td>Floating-point information.</td>
</tr>
<tr>
<td>T</td>
<td>10</td>
<td>Temporary floating-point information.</td>
</tr>
<tr>
<td>NTAP</td>
<td>4</td>
<td>Unit numbers set by subroutine TAPENO.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Dimension</td>
<td>Use</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NTAP (1) Set by program to the current input unit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NTAP (2) = 5, system input unit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NTAP (3) = 6, system output unit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NTAP (4) = 11, reserved input unit.</td>
</tr>
<tr>
<td>KBCD</td>
<td>20</td>
<td>Temporary storage</td>
</tr>
<tr>
<td>KBCD2</td>
<td>20</td>
<td>If KBCD (I) contains EBCDIC information, KBCD2 (I) contains the last two EBCDIC characters.</td>
</tr>
<tr>
<td>MP</td>
<td>1</td>
<td>Number of rows in primal subproblem.</td>
</tr>
<tr>
<td>NQ</td>
<td>1</td>
<td>Total number of columns including two dummy variables.</td>
</tr>
<tr>
<td>PLAMDA</td>
<td>1</td>
<td>L(0) value, initially 1.0. User may change this value.</td>
</tr>
<tr>
<td>PLIMIT</td>
<td>1</td>
<td>The limiting value of ( \lambda ) in the parametric run. The code sets this equal to 1.0E+20; the user may change this value.</td>
</tr>
<tr>
<td>KMQ</td>
<td>15</td>
<td>Fixed-point information.</td>
</tr>
</tbody>
</table>
Appendix D

SUMMARY OF COMMON STORAGE—RELOCATABLE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Dimension</th>
<th>Relocation</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S</td>
<td>A(N1)</td>
<td>Matrix entries.</td>
</tr>
<tr>
<td>IA</td>
<td>S/4 + 2</td>
<td>A(N4)</td>
<td>Matrix entry row numbers. (half-word integers)</td>
</tr>
<tr>
<td>KL</td>
<td>N/4 + 2</td>
<td>A(N9)</td>
<td>Column locator for matrix, row numbers for slacks and dual variables. (half-word integers)</td>
</tr>
<tr>
<td>KN</td>
<td>N/2 + 2</td>
<td>A(N10)</td>
<td>Column names—KN(I) contains the four high-order characters.</td>
</tr>
<tr>
<td>KN2</td>
<td>N/2 + 2</td>
<td>A(N18)</td>
<td>Column names—KN2(I) contains the two low-order characters.</td>
</tr>
<tr>
<td>NR</td>
<td>M/2 + 2</td>
<td>A(N12)</td>
<td>Row names—NR(I) contains the four high-order characters.</td>
</tr>
<tr>
<td>NR2</td>
<td>M/2 + 2</td>
<td>A(N19)</td>
<td>Row names—NR2(I) contains the two low-order characters.</td>
</tr>
<tr>
<td>B</td>
<td>M + 1</td>
<td>A(N2)</td>
<td>RHS entries.</td>
</tr>
<tr>
<td>KLT</td>
<td>N/4 + 2</td>
<td>A(N16)</td>
<td>Count of total entries for a column. (half-word integers)</td>
</tr>
<tr>
<td>JH</td>
<td>M/4 + 2</td>
<td>A(N7)</td>
<td>Column pivoted for row. (half-word integers)</td>
</tr>
</tbody>
</table>

†The resultant of the divisions is the largest integer not greater than the floating-point division.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Dimension</th>
<th>Relocation</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>KB</td>
<td>N/4 + 2</td>
<td>A(N8)</td>
<td>Row pivoted for column. (half-word integers)</td>
</tr>
<tr>
<td>P</td>
<td>M + 1</td>
<td>A(N13)</td>
<td>Prices.</td>
</tr>
<tr>
<td>X</td>
<td>M + 1</td>
<td>A(N14)</td>
<td>Solution vector.</td>
</tr>
<tr>
<td>Y</td>
<td>M + 1</td>
<td>A(N15)</td>
<td>Pivot column expanded.</td>
</tr>
<tr>
<td>IP</td>
<td>(3M/2 + 1)/4 + 3</td>
<td>A(N6)</td>
<td>Row pivoted for transformation. (half-word integers).</td>
</tr>
<tr>
<td>LE</td>
<td>(3M/2 + 1)/4 + 3</td>
<td>A(N11)</td>
<td>Column locator for transformation entries. (half-word integers)</td>
</tr>
<tr>
<td>IE</td>
<td>L/5</td>
<td>A(N5)</td>
<td>Transformation entry row numbers. (half-word integers)</td>
</tr>
<tr>
<td>E</td>
<td>4(L/5)</td>
<td>A(N3)</td>
<td>Transformation entries, non-zero values only.</td>
</tr>
</tbody>
</table>

NOTES: 1. The above dimensions assume that the program is double precision. For single dimension, multiply all the integer arrays by 2 to find the new dimensions.

2. M is number of rows. N is 2* (number of columns) + 2* (number of less-than and greater-than constants) + the number of equality constraints. S is number of matrix entries. L is amount of storage left after everything is assigned except E and IE. E is the packed matrix that is the inverse of the basis.
3. The rest of the storage adds up to:

(double precision) \((5/4)S + (7/4)N + 6M + 29\)

(single precision) \((3/2)S + (7/2)N + 8M + 40\)

The dimension of A in the main routine and BEGIN must be sufficient for the above storage and the E and IE arrays.

Before the program reads the input, the maximum number of matrix entries, N and M, are set. These values are KDIM(1), KDIM(2), and KDIM(3), respectively. Check to see if these are big enough for the input problem and that the dimension of A is at least:

(double precision) \((5/4)KDIM(1) + (7/4)KDIM(2) + 2KDIM(3) + 12\)

(single precision) \((3/2)KDIM(1) + (7/2)KDIM(2) + 3KDIM(3) + 22\)

4. On the H-compiler, the double-precision program uses approximately 70K bytes of storage plus the A-array storage.
Appendix E

DESCRIPTION OF KM AND KMQ ARRAYS IN COMMON

<table>
<thead>
<tr>
<th>KM(01)</th>
<th>Matrix identification word (EBCDIC).</th>
</tr>
</thead>
<tbody>
<tr>
<td>KM(02)</td>
<td>Iteration number.</td>
</tr>
<tr>
<td>KM(03)</td>
<td>RHS identification (EBCDIC).</td>
</tr>
<tr>
<td>KM(04)</td>
<td>Infeasibility flag (non-zero if infeasible).</td>
</tr>
<tr>
<td>KM(05)</td>
<td>'QOT(6) frequency.</td>
</tr>
<tr>
<td>KM(06)</td>
<td>Tally for number of times problem has gone infeasible.</td>
</tr>
<tr>
<td>KM(07)</td>
<td>Count of artificial at zero level at end of phase one.</td>
</tr>
<tr>
<td>KM(08)</td>
<td>Inversion frequency.</td>
</tr>
<tr>
<td>KM(09)</td>
<td>Cutoff frequency.</td>
</tr>
<tr>
<td>KM(10)</td>
<td>Name of column removed from basis (EBCDIC).</td>
</tr>
<tr>
<td>KM(11)</td>
<td>Name of pivot row (EBCDIC).</td>
</tr>
<tr>
<td>KM(12)</td>
<td>Name of column that is in &quot;P&quot; storage, or BCD word &quot;ERRORS&quot; or &quot;PRICE&quot; (EBCDIC).</td>
</tr>
<tr>
<td>KM(13)</td>
<td>Name of column brought into basis (EBCDIC).</td>
</tr>
<tr>
<td>KM(14)</td>
<td>Eject control.</td>
</tr>
<tr>
<td>KM(15)</td>
<td>Position in KN list of the column dropped from basis during quadratic and parametric.</td>
</tr>
<tr>
<td>KM(16)</td>
<td>Condition number.</td>
</tr>
<tr>
<td>KM(17)</td>
<td>Number of artificial.</td>
</tr>
<tr>
<td>KM(18)</td>
<td>Number of matrix entries.</td>
</tr>
<tr>
<td>KM(19)</td>
<td>Number of transformation entries.</td>
</tr>
<tr>
<td>KM(20)</td>
<td>6 EBCDIC blanks.</td>
</tr>
<tr>
<td>KM(21)</td>
<td>'DUAL(EBCDIC).</td>
</tr>
</tbody>
</table>
KM(22)  Artificial row control -- non-zero if rows have been
designated real.

KM(23)  Crash control (0 = invert, 1 = crash).

KM(24)  Capacity of core storage exceeded flag.

KM(25)  Number of iterations since last QOT(6).

KM(26)  Pivot counter (tallied by PIV, includes inversion).

KM(27)  Not used.

KM(28)  Iterations since last inversion.

KM(29)  Number of infeasibilities.

KM(30)  Negative X flag (non-zero when there are negative
X).

KM(31)  Exponent of determinant.

KM(32)  Iteration number of last inversion.

KM(33)  Not used.

KM(34)  Not used.

KM(35)  Inversion type (1 = inversion frequency, 2 =
inverted because of running out 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)  Length of common.

KM(39)  Amount of common left over after all storage
assigned except E and IE.

KM(40)  Maximum number of transformation entries allowed.

KM(41)  First word of last remark.

KM(42)  Pivot-step counter.

KM(43)  Set equal to one for solve.
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) Address (contains location of IA storage).
KM(50) Address (contains location of IE storage).
KMQ(1) Number of x variables plus one. Space is needed for one dummy variable in KL list. Set by TAP.
KMQ(2) Number of rows in $A_1$ matrix. Set by CAP.
KMQ(3) Number of rows in $A_2$ matrix. Set by CAP.
KMQ(4) Number of rows in $A_3$ matrix. Set by CAP.
KMQ(5) Location of first $x_1$ slack in KN list; zero if there are none. Set by TAP.
KMQ(6) Location of first $x_2$ slack in KN list; zero if there are none. Set by TAP.
KMQ(7) Location of first $y$ slack in KN list. Set by TAP.
KMQ(8) Location of first $t$ variable in KN list; zero if there are none. Set by TAP.
KMQ(9) Difference between the location of $y$ and $x$ type variables; equal to KMQ(1) + KMQ(2) + KMQ(3).
KMQ(10) Indicator for phase and size of problem
1 Find a feasible solution for the subproblem.
-1 Subproblem is feasible but artificialis are in basis at zero level. Attempt to drop these artificialis from basis.
0 Working on full problem.
KMQ(11)  Indicator for parametric run
0   Find the optimal solution for the initial value of \( L(0) \).
-1  Set by INF when PARAMETRIC control is read.
    Signal to code to introduce vector \( L-L(0) \)
    into the solution to start parametric run.
  1   In parametric run.

KMQ(12)  Position in basis list of the \( y_s \) variable that should be dropped from solution.

KMQ(13)  Indicator to note if RHS vector has been extended to include dual data, namely \( L(0) \cdot p \).
0   RHS has not been extended.
  1   RHS has been extended.

KMQ(14)  Position in KN list of the \( y_s \) variable that should be dropped from the solution.
Appendix F
FLOATING-POINT VARIABLES

Z(1)  Pivot tolerance
Z(2)  Set X to zero tolerance
Z(3)  Cost tolerance
Z(4)  Mixed pricing ratio
Z(5)  JMY tolerance, TMULT
Z(6)  Transformation entry tolerance
Z(7)  Pivot rejection tolerance
Z(8)  Current pivot ratio
Z(9)  Mantissa of determinant
Z(10) Total objective value
Z(11) Linear part of objective
Z(12) Quadratic part of objective
Z(13) Minimum reduced cost in nonbasic variables
Z(14) Not used
Z(15) Beginning time of last inversion
Z(16) End time of last inversion
Z(17) Current pivot element
Z(18) Not used
Z(19) Not used
Z(20) Not used
Appendix G

PROGRAM LISTING WITH INPUT PROBLEMS

// EXEC FORTCLG,PARM,FORT='MAP',REGION,GO=104K
//FORT,SYSIN DD =
IMPLICIT REAL*R (A-H,O-Z),INTEGER (I-N)
MODIFIED FOR THE 360
RSMFOR MODIFIED FOR OP

THE MFOR MAIN ROUTINE

DIMENSION L(1),KDIM(3)
COMMON / XBLANK / NN(19),M,MA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
NTAP(4),KBDCD(40),MP,NO,KMO(15),Z(20),T(10),PLM,DA,PLIMIT
A(3000)
A(6000),IA(6004),KL(1104),KN(1102),KN2(1102),NR(512),NR2(512),
B(511),KL(1104),KRN(1104)
INTEGER*2 IA,IL,KLT,KRN

THE COMPLETE COMMON IS ONLY IN SUBROUTINE BEGIN. THE OTHER
ROUTINES HAVE ONLY THE FIRST TWO LINES.

DIMENSION KM(50),KBDCD(20)
DIMENSION NFSAVE(19)
EQUIVALENCE (KM(51),KM2(1)),(KBDCD(21),KBDCD(1))
EQUIVALENCE (A(1),L(1)),(NN(1),N1),(NN(2),N2),(NN(3),N3),
(NN(4),N4),(NN(5),N5),(NN(6),N6),(NN(7),N7),
(NN(8),N8),(NN(9),N9),(NN(10),N10),(NN(11),N11),
(NN(12),N12),(NN(13),N13),(NN(14),N14),(NN(15),N15),
(NN(16),N16),(NN(17),N17)
DATA CON1/1,E-5/,CON2/1,E-7/,CON3/1,E-3/,CON4/1,E20/,ONE/1.0/
EQUIVALENCE (NN(18),N18),(NN(19),N19)
DIMENSION L2(1)
EQUIVALENCE (A(1),L2(1))
INTEGER*2 L2
LOGICAL FLGDBL
DATA IBLANK,IDL,IDL2/1H,4H,DU,2HAL/
DATA MINDM1,MINDM2,MINDM3/200,100,100/
DATA FLGDBL/.TRUE./

IF THE PROGRAM IS TO BE DOUBLE PRECISION, DECLARE THE VARIABLES,
CHANGE ABS TO DABS, AND SET FLGDBL (FOR STORAGE).

CALL BEGIN(IGETAD(NN(1)),K3)
K3 = K3/4
K33 = K3
KD1 = MINDM1
KD1 = MINDM1
KD1 = MINDM1
KD1 = MINDM1

SET VALUES FOR KD1 FOR MAXIMUM SIZE PROBLEM
KD1 = NO. OF ENTRIES
C KDIM(2) = NO. OF COLUMNS
C KDIM(3) = NO. OF ROWS
C MINDM1,2,3 ARE THE MIN. VALUES FOR KDIM(I). THEY CAN BE READ IN
C HERE, AS FOLLOWS.
C
C READ( 5,200 ) KDIM(1),KDIM(2),KDIM(3)
C 200 FORMAT( 10I8 )
C
C NA = 0
C 5 CONTINUE
C DO 10 I=1,K33
C 10 NN(I) = 0
C CALL TAPEND
C NT6 = NTAP(3)
C IF( NA.EQ.0 ) NA = NTAP(2)
C NTAP(1) = NA
C KM(20) = IBLANK
C KM2(20) = IBLANK
C KM(21) = IDUAL
C KM2(21) = IDUAL2
C KM(48) = KM(20)
C KM2(48) = IBLANK
C KM(1) = KM(48)
C KM2(1) = KM2(48)
C KM(3) = KM(48)
C KM2(3) = KM2(48)
C KM(5) = 10000
C KM(8) = 10000
C KM(9) = 10000
C KM(10) = KM(48)
C KM2(10) = KM2(48)
C KM(11) = KM(48)
C KM2(11) = KM2(48)
C KM(12) = KM(48)
C KM2(12) = KM2(48)
C KM(13) = KM(48)
C KM2(13) = KM2(48)
C KM(38) = K3
C CALL BEGIN( IGETAD(A(1)),KM(39) )
C KP(3) = 9
C KP(4) = 9
C KP(5) = 9
C KP(6) = 6
C KP(7) = 9
C KP(8) = 9
C KP(10) = 9
C KP(9) = 3
C Z(1) = CON1
C Z(2) = CON1
C Z(3) = -CON1
C Z(5) = CON1*CON1
C Z(6) = CON2
C Z(7) = CON3
C Z(9) = ONE
C 40 CONTINUE
C PLAMDA = ONE
PLIMIT = CON4
KMQ(10) = 1
N1 = 1

THE FOLLOWING INSTRUCTIONS SHOULD BE COMPATIBLE WITH THE
DIMENSIONS FOR IA THROUGH KNR

NFAC = 1
IF(FLGDBL)NFAC = 2
IAFAC = NFAC+2

IA HAS TWO INDECES PER WORD, AS DOES THE OTHER INDEXES. KN AND
NR ARE BCD CHARACTERS. IN THE ORIGINAL 7094 VERSION, ONLY
IA AND IE WERE PACKED.

N4 = N1+KDIM(1)
KM(49) = NFAC( A(N4) )
N9 = N4+(KDIM(1)+4)/IAFAC+1
N10 = N9+(KDIM(2)+4)/IAFAC+1
N18 = N10+(KDIM(2)+2)/NFAC+1
N12 = N18+(KDIM(2)+2)/NFAC+1
N19 = N12+(KDIM(3)+2)/NFAC+1
N2 = N19+(KDIM(3)+2)/NFAC+1
50 CONTINUE
N16 = N2+KDIM(3)+1
N17 = N16+(KDIM(2)+4)/IAFAC+1
K = 0

60 CONTINUE
CALL INP( K,A,B,E,IA,IE,IP,JH,KB,KL,KN,LE,NP,LY,KLT,KNR,KN2,NR2 )

TALLY RIGHT HAND SIDE INTO MATRIX COUNT

JF = KMQ(7)+KMQ(1)
JL = JF+KMQ(2)+KMQ(3)+KMQ(4)-1
DO 80 I=4,MP
K = N2+I-1

81 B(I)
IF( A(K) = EQ.0.0 ) GO TO 80
K1 = (N12-I)*NFAC+I
K2 = (N19-I)*NFAC+I
DO 70 IJ=JF+JL
K3 = (N10-I)*NFAC+I
K4 = (N18-I)*NFAC+I
COMPARE KN AND NR

DO 80 I=1,MP
IF ( L(K3) .NE. L(K1) .OR. L(K4) .NE. L(K2) ) GO TO 70

C C C

KLT

K5 = (N16-1)*IAFAC+II
L2(K5) = L2(K5)+1
GO TO 80

70 CONTINUE

KM(14) = 0
WRITE( NT6+100 ) L(K1)+L(K2)+A(K)
100 FORMAT(6OH4--CODE CANNOT FIND COLUMN NAME FOR TRANSPOSE OF ROW NAME )
1ME# A4-A2.20H FOR WHICH THE RHS = +F13.6 )
CALL ENDTIE

80 CONTINUE

DO 85 I=1,19

85 NNSAV(I) = NN(I)

MP0 = M+1
NP0 = N+1
N4 = N1+KM(18)
N9 = N4+(KM(18)+4)/IAFAC+1
N10 = N9+ (N+4)/IAFAC+1
N18 = N10+ (N+2)/NFAC+1
N12 = N18- (N+2)/NFAC+1
N19 = N12- (M+2)/NFAC+1
N2 = N19+ (M+2)/NFAC+1

90 CONTINUE

N16 = N2+MP0
N7 = N16+ (N+4)/IAFAC+1
N8 = N7+ (M+4)/IAFAC+1
N13 = N8+ (N+4)/IAFAC+1
N14 = N13+MP0
N15 = N14+MP0
N6 = N15+MP0
KM(36) = M+M/2+5
N11 = N6+ (KM(36)+4)/IAFAC+1
N5 = N11+ (KM(36)+4)/IAFAC+1
KM(39) = (KM(39)-N5*NFAC*4)/(4*AFAC)
NIE = KM(39)/3
IF( FLGBLB ) NIE = KM(39)/5
KM(40) = 2*(NIE-1)
N3 = N5*NIE+1

SET TRUE VALUES FOR PARAMETERS BEFORE RELOCATING DATA

KM(49) = NOP( A(N4)*0 )
KM(50) = NOP( A(N5)*1 )

RELOCATE DATA

NIA = N9-N4
K1 = NNSAV(4)
K2 = NNSAV(9)-1
DO 110 I=K1,K2
J = N4+I-K1
A(J) = A(I)
110 CONTINUE

K1 = NNSAV(9)
K2 = NNSAV(10) - 1
DO 120 I = K1, K2
   J = N9 + I - K1
   A(J) = A(I)
120 CONTINUE
K1 = NNSAV(10) - 1
K2 = NNSAV(18) - 1
DO 130 I = K1, K2
   J = N10 + I - K1 - 1
   A(J) = A(I)
130 CONTINUE
K1 = NNSAV(18) - 1
K2 = NNSAV(12) - 1
DO 135 I = K1, K2
   J = N18 + I - K1 - 1
   A(J) = A(I)
135 CONTINUE
K1 = NNSAV(12) - 1
K2 = NNSAV(19) - 1
DO 140 I = K1, K2
   J = N12 + I - K1 - 1
   A(J) = A(I)
140 CONTINUE
K1 = NNSAV(19) - 1
K2 = NNSAV(2) - 1
DO 145 I = K1, K2
   J = N19 + I - K1 - 1
   A(J) = A(I)
145 CONTINUE
K1 = NNSAV(2)
K2 = NNSAV(16) - 1
DO 150 I = K1, K2
   J = N2 + I - K1
   A(J) = A(I)
150 CONTINUE
K1 = NNSAV(16) - 1
K2 = NNSAV(17) - 1
DO 160 I = K1, K2
   J = N16 + I - K1 - 1
   A(J) = A(I)
160 CONTINUE
C REDUCE N COUNT TO IGNORE LPX COLUMN FOR PARAMETRIC RUN
C N = N - 1
C CLEAR REMAINDER OF CORE AFTER DATA HAS BEEN RELOCATED
C CALL BEGIN( IGETAD(A(N7)), II )
   II = II / 4
   IF ( FLGDBL ) II = II / 2
   DO 180 J = 1, II
   I = N7 + J - 1
   A(I) = 0.0
180 CONTINUE
GO TO 60
END
SUBROUTINE BOS( A,B,E,I,E,IP,JH,KB,KB,KL,KN,LE,NR,P,X,Y,KLT,KN2,NR2,  
 1 IA )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON /XBLANK/N(19),M,M,MB,MC,MD,ME,DF,N,KN(100),KP(12),
 1 NTAP(4),KBCC(40),MP,NQ,KMQ(15),Z(20),T(10),PLAMDA,PLIMIT

BOS CALLED BY INP
KQ(M) = 1  --  FIND A PRIMAL FEASIBLE SOLUTION
= -1  --  FEASIBLE SOLUTION HAS ARTIFICIALS AT ZERO
       --  LEVEL. CODE ATTEMPTS TO DRIVE THEM OUT OF
       --  THE BASIS.
KQ(M) = 0  AND
KQ(M) = 0
KQ(M) = 0
       --  QUADRATIC ALGORITHM ON FULL PROBLEM
       --  PARAMETRIC RUN ON LAMBDA

DIMENSION A(1),B(1),E(1),IE(1),IP(1),JH(1),KB(1),KL(1),KN(1),
 1 LE(1),NR(1),P(1),X(1),Y(1),KLT(1)
DIMENSION KN2(1),NR2(1)
DIMENSION KM2(50),KBCC2(20)
DIMENSION IA(1)
EQUIVALENCE (KM(51),KM2(1)), (KBCC(21),KBCC2(1))
INTEGER*2 IE,IP,JH,KB,KB,LE,KLT
INTEGER*2 IA

NT6 = NTAP(3)
KK1 = KM(2)+1
KK2 = KM(9)

SET MAJOR DO LOOP TO COMPUTE UP TO KM(9) ITERATIONS
DO 310 KKO=KK1,KK2
  INVERSION FREQUENCY
  IF( KM(28).NE.KM(8) ) GO TO 20
  KM(35) = KM(35)+1
  10 CONTINUE
  CALL VER( A,B,E,IE,IP,JH,KB,KB,KL,KN,LE,NR,X,Y,KLT,KN2,NR2,IA )
  ASSIGN 20 TO KERR
  CALL ERR( A,B,JH,KB,KB,KN,NR,P,X,Y,KN2,NR2,IA )
  20 CONTINUE
  CHECK CHANGE OF PHASE
  CALL XCK( K,JH,X )
  IF( KM(4)=K ) 30,70,60
  30 CONTINUE

SOLUTION HAS GONE INFEASIBLE
WRITE( NT6, 100 ) KM(2)
100 FORMAT(32H0#BECAUSE INFEASIBLE AT ITERATION ,I4 )
KM(14) = 0
IF( KM(28) .NE. 0 ) GO TO 10
IF( KM(6) .GT. 0 ) GO TO 340

RESET PARAMETERS TO FIND A FEASIBLE SOLUTION, INVERT
KM(6) = KM(6)+1
KM(23) = -2
GO TO 10
60 CONTINUE

SOLUTION IS FEASIBLE, CHECK FOR ARTIFICIALS
IF( KMQ(10) .EQ. 0 ) GO TO 120
IF( KMQ(10) .LT. 0 .AND. KM(7) .EQ. 0 ) GO TO 120
IF( KMQ(10) .LT. 0 ) GO TO 70
IF( KM(17) .EQ. 0 ) GO TO 120
KMQ(10) = -1
KM(7) = KM(17)
70 CONTINUE

NO CHANGE IN FEASIBILITY
KM(37) = 0
IF( KMQ(10) .EQ. 0 ) GO TO 130
75 CONTINUE
CALL GET( IR, E, IP, JH, LE, P, Y, IE )
80 CONTINUE
CALL MIN( JM, A, KB, KL, P, Y, IA )
CALL IMAGE( KM(12), KM2(12), 1, 8HPRICE )
MC = 0
IF( KMQ(10) .LT. 0 ) GO TO 90
IF( Z(I3) .LT. Z(3) ) GO TO 180
IF( KM(37) .EQ. 0 ) GO TO 340
GO TO 270
90 CONTINUE
IF( IR .EQ. 0 ) GO TO 120

****** ABS CALL
110 CONTINUE
IF( DABS(Z(I3)) .GE. Z(1) ) GO TO 180
JH(IR) = -1
KM(7) = KM(7)-1
GO TO 75
120 CONTINUE

SOLUTION IS FEASIBLE
WRITE( NT6, 200 ) KM(2), KM(42)
KM(14) = 0
200 FORMAT(32H0#FEASIBLE AT ITERATION ,I4, 1H, , I6, 6H STEPS )
CALL UCGV( 2 )

GO INVERT AND EXTEND BASIS FOR FULL PROBLEM
KM(4) = 0

BOS 0560
BOS 0570
BOS 0580
BOS 0590
BOS 0600
BOS 0610
BOS 0620
BOS 0630
BOS 0640
BOS 0650
BOS 0660
BOS 0670
BOS 0680
BOS 0690
BOS 0700
BOS 0710
BOS 0720
BOS 0730
BOS 0740
BOS 0750
BOS 0760
BOS 0770
BOS 0780
BOS 0790
BOS 0800
BOS 0810
BOS 0820
BOS 0830
BOS 0840
BOS 0850
BOS 0860
BOS 0870
BOS 0880
BOS 0890
BOS 0900
BOS 0910
BOS 0920
BOS 0930
BOS 0940
BOS 0950
BOS 0960
BOS 0970
BOS 0980
BOS 0990
BOS 1000
BOS 1010
BOS 1020
BOS 1030
BOS 1040
BOS 1050
BOS 1060
BOS 1070
BOS 1080
BOS 1090
BOS 1100
BOS 1110
BOS 1120
BOS 1130
BOS 1140
KMQ(10) = 0
KMQ(14) = 0
KM(15) = 0
GO TO 10
130 CONTINUE

TEST FOR PARAMETRIC KMQ(11)=+1 OR -1

IF( KMQ(11) ) 140,160,150
140 CONTINUE

FIRST ITERATION FOR PARAMETRIC ALGORITHM

KMQ(11) = 1
N = NQ+1
JM = N
GO TO 180
150 CONTINUE

CHECK IF LAMDA IS AT ITS LIMIT

IF( PLAMDA+X(1)-PLIMIT-LT.0.0 ) GO TO 170
30 TO 362
160 CONTINUE

QUADRATIC ALGORITHM

IF( KMQ(14).EQ.0 ) GO TO 240
IF( KM(15).EQ.KMQ(14) ) GO TO 235
165 CONTINUE

: = KMQ(14)
KMQ(12) = KB(I)

NEW COLUMN IS COMPLEMENT OF JOLD

170 CONTINUE

JM = KM(15)-KMQ(9)
IF( JM.LE.0 ) JM = KM(15)+KMQ(9)
180 CONTINUE

TRANSFORM SELECTED COLUMN

KM(13) = KN(JM)
KM2(13) = KN2(JM)
MA = 0
IF( JM.GE.KMQ(7) ) MA = 1
CALL EXP( JM, A, B, KL, NR, Y, NR2, IA )
CALL JMY( E, IP, LE, Y, IE )
IF( KMQ(10) ) 270,190,220
190 CONTINUE

TRANSFORMED COLUMN TO P STORAGE IF SOLUTION FEASIBLE

DO 210 I=1,M
210 P(I) = Y(I)
KM(12) = KM(13)
KM2(12) = KM2(13)
MC = MA
220 CONTINUE
   CALL RCW( IR, JH, X, Y )
   ..
   TEST PIVIT
   IF( IR.EQ.0 ) GO TO 350
   Z(17) = Y(IR)
   IF( KMQ(11).NE.0 ) GO TO 270
   ******** ABS CALL
   IF( DABS(Z(17)).GE.Z(7) ) GO TO 270
   CHECK TYPE OF COLUMN BEING REJECTED
   IF( KMQ(10).NE.0 ) GO TO 232
   IF REJECTED COLUMN IS X/S, FIND NEW Y/S
   IF( JM.EQ.KMQ(14)-KMQ(9) ) GO TO 230
   IF NOT X/S, REINVERT OR USE PIVOT
   IF( KM(28).EQ.0 ) GO TO 270
   GO TO 10
230 CONTINUE
   KM(15) = 0
232 CONTINUE
   K = 0
   IF( JM.EQ.KMQ(7) ) K = 1
   WRITE( NT6,300 ) KN(JM),KN2(JM),KM(K+20),KM2(K+20),NR(IR),
   1 NR2(IR),Z(17),KM(2)
300 FORMAT(2H01,A4,A2,A4,A2,1H7,A4,A2,3H) =,1PE10.2,14H REJECTED ITN
   1 I4 )
   KB(JM) = -1
   KM(37) = JM
   KM(47) = KM(37)
   IF( KMQ(10).NE.0 ) GO TO 80
   Y/S WAS DROPPED FROM BASIS, FIND NEW ONE
235 CONTINUE
   KMQ(14) = 0
240 CONTINUE
   T(1) = 0.0
   KM(45) = KMQ(7)+KMQ(9)
   T(2) = -1.0E20
   IF( KM(37).EQ.0 ) GO TO 250
   I = KM(37)+KMQ(9)
   SAVE NAME OF Y/S FOR REJECTED X/S
   KM(46) = I
   I = KB(1)
   STORE VALUE OF Y/S FOR REJECTED COLUMN
   T(2) = X(I)
250 CONTINUE
DO 260 I=1,M
   IF( X(I).GE.T(1) ) GO TO 260
   IF( JH(I).GE.KM(45) ) GO TO 260
   CONTINUE
   CHOOSE VALUE THAT IS LARGER THAN REJECTED Y/S
   IF( T(2).GE.X(I) ) GO TO 260
   T(1) = X(I)
   KMQ(14) = JH(I)
260 CONTINUE
   KM(15) = KMQ(14)
   IF( KMQ(14).NE.0 ) GO TO 165
   IF( KMQ(37).EQ.0 ) GO TO 330
   KMQ(14) = KM(46)

270 CONTINUE
   KM(11) = NR(IR)
   KM(211) = NR2(IR)
   JOLD = JH(IR)
   KM(15) = JOLD
   KM(10) = KN(JOLD)
   KM(210) = KN2(JOLD)
   MB = 0
   IF( JOLD.GE.KM(7) ) MB = 1
   PIVOT
   KM(42) = KM(42)+1
   KB(JM) = IR
   JH(IR) = JM
   IF( JOLD.EQ.0 ) KM(17) = KM(17)-1
   IF( JOLD.NE.0 ) KB(JOLD) = 0
   CALL PIV( IR,E,IE,IP,LE,X,Y )
   KM(2) = KM(2)+1

   TEST FOR PARAMETRIC ITERATION OUTPUT
   IF( KMQ(11).NE.0.AND.KP(9).NE.0 ) CALL UDOUT( 9 )
   IF( KMQ(11).EQ.0.AND.KP(1).NE.0 ) CALL UDOUT( 1 )

   TEST TO SEE IF CORE SPACE IS EXCEEDED
   IF( KM(24).EQ.0 ) GO TO 280
   KM(35) = 1
   KM(28) = KM(8)
   GO TO 310
280 CONTINUE
   KM(28) = KM(28)+1
   KM(25) = KM(25)+1

   TEST QOT(6) FREQUENCY
   IF( KM(25).EQ.KM(5) ) CALL UDOUT( 6 )
310 CONTINUE

   WRITE( 6,500 )
KM(14) = 0

500 FORMAT(26H1=ITERATION LIMIT EXCEEDED )
320 CONTINUE
   K = 8
   ASSIGN 380 TO KERR
   GO TO 370
330 CONTINUE
   WRITE( NT6,600 )
   KM(14) = 0
600 FORMAT( 18H1=OPTIMAL SOLUTION )
   K = 3
   ASSIGN 390 TO KERR
   GO TO 370
340 CONTINUE
   WRITE( 6,700 )
   KM(14) = 0
700 FORMAT( 22H1=NO FEASIBLE SOLUTION )
   K = 4
   ASSIGN 380 TO KERR
   GO TO 370
350 CONTINUE
   IF( KMQ(11).EQ.0 ) GO TO 360
   KP(5) = 9
   WRITE( NT6,800 )
   KM(14) = 0
800 FORMAT(50H1=FINAL STEP, PARAMETRIC VARIABLE L IS UNBOUNDED )
   K = 5
   ASSIGN 390 TO KERR
   GO TO 370
360 CONTINUE
   WRITE( NT6,900 )
   KM(14) = 0
900 FORMAT( 19H1=INFINITE SOLUTION )
   K = 5
   GO TO 364
362 CONTINUE
   WRITE( NT6,1000 )
   KM(14) = 0
1000 FORMAT( 38H1= L IS EQUAL TO OR GREATER THAN LIMIT )
   K = 10
364 CONTINUE
   ASSIGN 390 TO KERR
370 CONTINUE
   KM(14) = 1
   CALL QOUT( K )
375 CONTINUE
   CALL ERR( A,B,JH,KB,KL,KN,NR,P,X,Y,KN2,NR2,IA )
   GO TO KERR, ( 20,380,390 )
380 CALL ENDTIE
390 CONTINUE
RETURN
END

SUBROUTINE INP( KIN,A,B,E,IA,IE,IP,JH,KB,KL,KN,LE,NR,P,X,Y,KLT, INP 0010
   1 KNR,KN2,NR2 )
   INP 0020
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19), M, MA, MB, MC, MD, ME, MF, N, KM(100), KP(12),
1 NTAP(4), KB2CD(40), MP, NQ, KM(15), Z(20), T(10), PLAMCA, PLIMIT

INP CALLED BY FIN - READS CONTROL CARDS

DIMENSION A(1), E(1), IE(1), IP(1), JH(1), KB(1), KL(1), KN(1), LE(1)
DIMENSION NR(1), P(1), X(1), Y(1), IA(1), B(1), KLT(1), KNR(1)
DIMENSION KN2(1), NR2(1)
DIMENSION KM(50), KB2CD2(20)
EQUIVALENCE (KM(51), KM2(1)) * (KB2CD(21), KB2CD2(1))
INTEGER*2 IA, IE, IP, JH, KB, KL, LE, KLT, KNR

K = KIN
NT6 = NTAP(3)
CALL IMAGE( KBEGN, KBEGN2, 1, 0MBEGIN )
IF( K NE 0 ) GO TO 80
ASSIGN 60 TO K
CALL TIE
GO TO KTNIE, ( 60, 10 )
CONTINUE
WRITE( NT6, 100 )
KM(14) = 0
WRITING FORMAT( 17H SKIP TO NEXT RUN /1H )
CONTINUE
READ( NT5, 901 ) K, K2
CALL SEARCH( K, K2, KK, 4, 32HBEGIN TAPE EOF CORE )
GO TO ( 72, 30, 40, 80, 20 ), KK
CONTINUE
NT5 = NTAP(1)
GO TO 50
CONTINUE
NT5 = NTAP(2)
GO TO 20
CONTINUE
ASSIGN 10 TO K
NT5 = NTAP(1)

**************
READ + WRITE CONTROL
CONTINUE
READ( NT5, 900 ) K, K2, (KB2CD(I), I=5, 20)
READ PAGE ON BEGIN CARD
IF( K NE KBEGIN ) GO TO 75
WRITE( NT6, 909 )
CONTINUE
CALL IMAGE( KB2CD(1), KB2CD(2), 1, 8H(1X, A4, A) )
CALL IMAGE( KB2CD(3), KB2CD(4), 1, 8H, 66H )
CALL IMAGE( KB2CD(21), KB2CD(22), 1, 8H )
WRITE( NT6, KBCC ) K, K2
KM(14) = 0
CONTINUE
CALL SEARCH( K, K2, KK, 28, 224HALTERA ALTERB BASIS BASIS* BEGIN
1 END EOF ERRORS FREQUE GO INVERT MATRIX OINP 0610
2 OUTPUT PARAME L(0) LIMIT PRMODE PUNCH R RHS ROWS SETINP 0620
BINV SOLVE TAPE TIME TOLERA L(0) EXIT
GO TO 90, 110, 120, 130, 140, 150, 160, INP 0650
170, 180, 190, 210, 220, 230, 250, 260, 270, 280, 290, INP 0660
2 310, 320, 350, 360, 370, 390, 410, 420, 270, 440, 430, INP 0670
KK

--------------
ALTERA ... CHANGE MATRIX DATA

90 CONTINUE
KK = 4
GO TO 340

ALTERR ... CHANGE RHS DATA

110 CONTINUE
KK = 5
GO TO 330

BASIS ... READ BASIS HEADINGS

120 CONTINUE
KK = 2
GO TO 340

BASIS* ... READ CARDS THAT WERE PUNCHED BY THE CODE

130 CONTINUE
KK = 3
GO TO 340

BEGIN ... CLEAR DATA REGION FOR NEW RUN

140 CONTINUE
KIN = 0
GO TO 240

BLANK ... BLANK CARD IS ERROR

150 CONTINUE
WRITE( NT6, 200 )
KM(14) = 0
200 FORMAT( 35H0CONTROL CARD HAS BLANK FIRST FIELD )
GO TO 10

END

160 GO TO 70

EDF ... RESUME READING OF MONITOR TAPE

170 CONTINUE
NTAP(1) = NTAP(2)
NT5 = NTAP(1)
GO TO 70

ERRORS ... CHECK SOLUTION
180 CONTINUE
   CALL ERR( A,B,JH,KB,KL,KN,NR,P,X,Y,KN2,NR2,IA )
   GO TO 70
190 CONTINUE

FREQUENCIES	READ INTEGRAL CONSTANTS KM

READ( NT5,902 ) K,K2,KBCD(5),KBCDZ(5),KFIX
IF( K,NE,KM(48),OR,K2,NE,KM(48) ) GO TO 20
   CALL SEARCH( KBCD(5),KBCDZ(5),KK,3,24HOUTPUT INVERT CUTOFF )
   IF( KK,GT,3 ) GO TO 190
   KM(45) = KBCD(5)
   KM2(45) = KBCDZ(5)
   WRITE( NT6,300 ) KM(45),KM2(45),KFIX
   KM(14) = 0
300 FORMAT( 7X,A4,A2,18 )
   IF( KK,EQ,2 ) KM(8) = KFIX
   IF( KK,LT,2 ) KM(5) = KFIX
   IF( KK,GT,2 ) KM(9) = KFIX
   GO TO 190

GO		--- COMPUTE

210 CONTINUE
   CALL BOS( A,B,E,IE,IP,JH,KB,KL,KN,LE,NR,P,X,Y,KLT,KN2,NR2,IA )
   GO TO 70

INVERT		--- FORM BASIS INVERSE

220 CONTINUE
   ASSIGN 70 TO KVER
   GO TO 380

MATRIX		--- RETURN TO MAIN ROUTINE TO INPUT MATRIX

230 CONTINUE
   KIN = 1
240 RETURN

OUTPUT		--- SPECIAL OUTPUT

250 CONTINUE
   CALL UOUT( 6 )
   GO TO 70

PARAM		--- START PARAMETRIC RUN

260 CONTINUE
   KMQ(11) = -1
   GO TO 210
   PLAMDA = L(0)

270 CONTINUE
   READ( NT5,905 ) PLAMDA
   KMQ(13) = 0
   GO TO 70
C PLIMIT ... = LIMIT

280 CONTINUE
READ( NT5,905 ) PLIMIT
GO TO 70

C

C PRMODE ... OUTPUT CONTROL

290 CONTINUE
READ( NT5,904 ) (KP(I),I=1,10)
GO TO 70

310 CONTINUE

C

C PUNCH ... PUNCH BASIS HEADINGS

KM(16) = 6
CALL OUP( JH,KN,NR,KN2,NR2 )
GO TO 70

320 CONTINUE

C

C RHS ... READ RIGHT HAND SIDE

KK = 1
330 KMQ(13) = 0
340 CONTINUE
CALL CAP( KK,K,K2,A,B,JH,KB,KL,KN,NR,KLT,KN2,NR2,IA )
GO TO 80

350 CONTINUE

C

C ROWS ... READ LIST OF ROW NAMES WITH IDENTIFICATION FOR TYPE OF EQUATION

KK = 6
GO TO 340

C

C SETINV ... SET INVERSION FREQUENCIES

360 CONTINUE
KM(8) = M/10 + 6
GO TO 70

370 CONTINUE

C

C SOLVE ... CRASH, THEN START PHASE ONE

ASSIGN 210 TO KVER
KM(43) = 1
KM(23) = 1
380 CONTINUE
CALL VER( A,B,E,IE,IP,JH,KB,KL,KN,LE,NR,X,Y,KLT,KN2,NR2,IA )
GO TO KVER , (70,210)

C

C TAPE ... INPUT FROM RESERVED TAPE

390 CONTINUE
NTAP(1) = NTAP(4)
NT5 = NTAP(1)
GO TO 70

C

C TIME ... TREAT AS DUMMY CONTROL

INP 1800
INP 1810
INP 1820
INP 1830
INP 1840
INP 1850
INP 1860
INP 1870
INP 1880
INP 1890
INP 1900
INP 1910
INP 1920
INP 1930
INP 1940
INP 1950
INP 1960
INP 1970
INP 1980
INP 1990
INP 2000
INP 1990
INP 2010
INP 2020
INP 2010
INP 2030
INP 2040
INP 2050
INP 2060
INP 2070
INP 2080
INP 2090
INP 2100
INP 2110
INP 2120
INP 2130
INP 2140
INP 2150
INP 2160
INP 2170
INP 2180
INP 2190
INP 2200
INP 2210
INP 2220
INP 2230
INP 2240
INP 2250
INP 2260
INP 2270
INP 2280
INP 2290
INP 2300
INP 2310
INP 2320
INP 2330
INP 2340
INP 2350
INP 2360
INP 2370
INP 2380
410 GO TO 70

TOLERANCES ... READ FLOATING CONSTANTS Z

420 CONTINUE
READ ( NT5,903 ) K,K2,KBCD(5),KBCD2(5),T(1)
IF( K.NE.KM(48).OR.K2.NE.KM(48) ) GO TO 80
CALL SEARCH( KBCD(5),KBCD2(5),K,K,7,56HPIVOT
1 TMULT ENTRY REJECT)
IF( K.K .GT. .7 ) GO TO 420

******
ABS CALL

Z(KK) = DABS( T(1) )
IF( K.K .EQ. 3 ) Z(3) = -Z(3)
KM(45) = KBCD(5)
KM2(45) = KBCD2(5)
WRITE( NT6,400 ) KM(45),KM2(45),Z(KK)
KM(14) = 0
400 FORMAT( 7X,A4,A2,E12.4 )
GO TO 420

430 CONTINUE

COMMENT CARD

KM(41) = K
KM2(41) = K2
GO TO 70

900 FORMAT( A4,A2,16A4 )
901 FORMAT( A4,A2 )
902 FORMAT( 2(A4,A2),I8 )
903 FORMAT( 2(A4,A2),E8-1 )
904 FORMAT(10I1)
905 FORMAT(18X,F12.6)
908 FORMAT(20I2)
909 FORMAT(1H1)
440 CONTINUE

EXIT ... EJECT PAGE AND EXIT - FINISHED

WRITE( NT6,909 )
CALL EXIT
END

SUBROUTINE CAP( KK,K,K2,A,B,JH,KB,KL,NR,KLT,KN2,NR2,IA )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19),M,MA,MB,MC,MD,ME,ME,F,N,KM(100),KP(12),
1 NTAP(4),KBCD(40),MP,NQ,KM(15),T(20),T(10),PLAMDA,PLIMIT
CALLED BY INP FOR RHS, BASIS, BASIS*, ALTERA, ALTERB, ROWS

DIMENSION A(1),B(1),JH(1),KB(1),KL(1),KN(1),NR(1),KLT(1)
CAP 0010
CAP 0020
CAP 0030
CAP 0040
CAP 0050
CAP 0060
CAP 0070
CAP 0080
CAP 0090
DIMENSION IA(1)
DIMENSION KN2(1), NR2(1)
DIMENSION KM(50), KB(50), KBCD(2, 20)
EQUIVALENCE (KM(51), KM2(1)), (KBCD(21), KBCD2(1))
INTEGER*2 JH, KB, KL, KLT, IA

KERR = 0
NB = NTAP(1)
NOT = NTAP(3)
NT6 = NTAP(3)
IIA = KMQ(2) + 4
IIB = IIA + KMQ(3)

SET LOCATIONS FOR FIRST AND LAST TRANSPOSED CONSTRAINT

JF = KMQ(7) + KMQ(1)
JL = JF + KMQ(2) + KMQ(3) + KMQ(4) - 1
GO TO (10, 170, 160, 90, 30, 540), KK

K=1 RH5S RIGHT HAND SIDES
12X, A6(ROW), F12.6(QUAN)

10 IF (M.EQ.0) GO TO 730
   DO 20 I = 1, M
20 B(I) = 0.0

K=5 ALTER8 RIGHT-HAND SIDES WITHOUT ZEROING

30 CONTINUE
   KM(3) = KM(41)
   KM2(3) = KM2(41)
40 CONTINUE
   READ (NB, 100) K, K2, KBCD(5), KBCD2(5), JJ, JJ2, AA
   IF (K.NE.KM(48) .OR. K2.NE.KM(48)) GO TO 720
   DO 50 J = 1, M
   IF (NR(J) .EQ. JJ .AND. NR2(J) .EQ. JJ2) GO TO 60
50 CONTINUE
55 CONTINUE
   WRITE (NT6, 200) JJ, JJ2
   KM(14) = 0
200 FORMAT (19H ILLEGAL ROW(COL), A4, A2, 10H, IN ENTRY)
   KERR = 1
   GO TO 40
60 CONTINUE
   IF (K.€. 4) GO TO 110

CHECK KM(50) TO SEE IF MATRIX HAS BEEN READ. IF YES TALLY B(I)
ENTRY FOR COLUMN THAT HAS NR(I) NAME.

IF (KM(50) .EQ. 0) GO TO 80
IF (B(J) .EQ. 0.0 .AND. AA .EQ. 0.0) GO TO 40
IF (B(J) .NE. 0.0 .AND. AA .NE. 0.0) GO TO 80
IT = 1
IF (AA .NE. 0.0) IT = -IT
DO 70 IT = JF, JL
IF (KN(I) .NE. JJ .OR. KN2(I) .NE. JJ2) GO TO 70

GO TO 15
KLT(I) = KLT(I)+IT
GO TO 80
70 CONTINUE
GO TO 55
80 CONTINUE

....... REVERSE SIGN OF DATA IN MINIMUM CONSTRAINT
B(J) = AA
IF( J.GE.IJA.AND.J.LT.IIB ) B(J) = -AA
GO TO 40

....... KK=4 ALTERA CHANGE MATRIX ENTRY
90 CONTINUE
0 KMQ(13) = 0
0 KM(1) = KM(41)
0 KM2(1) = KM2(41)
0 GO TO 40
0 J CONTINUE
0 DO 120 II=1,N
0 IF( KN(II).EQ.KBCD(5).AND.KN2(II).EQ.KBCD2(5) ) GO TO 130
3 J CONTINUE
) II = N
JJ = KBCD(5)
JJ2 = KBCD2(5)
GO TO 55
130 CONTINUE
NOQ = KL(II)
NOT = KL(II+1)-1
DO 140 IJ=NOQ,NOT
140 CONTINUE

******* JA CALL
 IF( IA(IJ).EQ.J ) GO TO 150
140 CONTINUE
GO TO 55
150 CONTINUE
A(IJ) = AA
IF( J.GE.IJA.AND.(J.LT.IIB.OR.J.GT.MP) ) A(IJ) = -AA
IF( A(IJ).EQ.-AA.AND.JJ.EQ.KBCD(5).AND.JJ2.EQ.KBCD2(5) ) A(IJ) = 1
2.0*A(IJ)
GO TO 40

KK=3 BASIS* LIST Punched by Code

160 CONTINUE
KMQ(10) = 1
KM(23) = 0
KM(4) = 1
GO TO 180

KK=2 BASIS LIST OF BASIS HEADINGS

170 CONTINUE
KMQ(10) = 0

CAP 0690
CAP 0700
CAP 0710
CAP 0720
CAP 0730
CAP 0740
CAP 0750
CAP 0760
CAP 0770
CAP 0780
CAP 0790
CAP 0800
CAP 0810
CAP 0820
CAP 0830
CAP 0840
CAP 0850
CAP 0860
CAP 0870
CAP 0880
CAP 0890
CAP 0900
CAP 0910
CAP 0920
CAP 0930
CAP 0940
CAP 0950
CAP 0960
CAP 0970
CAP 0980
CAP 0990
CAP 1000
CAP 1010
CAP 1020
CAP 1030
CAP 1040
CAP 1050
CAP 1060
CAP 1070
CAP 1080
CAP 1090
CAP 1100
CAP 1110
CAP 1120
CAP 1130
CAP 1140
CAP 1150
CAP 1160
CAP 1170
CAP 1180
CAP 1190
CAP 1200
CAP 1210
CAP 1220
CAP 1230
CAP 1240
CAP 1250
CAP 1260
CAP 1270
\begin{verbatim}
180 CONTINUE
   NL = NQ+1
   II A = KMQ(7)-1
   IIB = NL
   IF( KMQ(8) .NE. 0 ) IIB = KMQ(8)
   DO 190 I=1, NL
190   KB(I) = 0
   KM(45) = 0

   : SET JH(I) LIST TO 12345

   DO 210 I=1,F,M
210   JH(I) = 12345

   220 CONTINUE
   READ( NB,12 ) K,K2,(KBCD(I),KBCD2(I),I=5,15)
   IF( K.EQ.KM(48) .AND. K2.EQ.KM2(48) ) GO TO 240

   NEW CONTROL CARD, TERMINATE BASIS

   NXXXXX = 0
   KB(NXXXXX) = 0
   IF( KMQ(10) .EQ. 0 ) GO TO 725

   CLEAR DUAL BASIS HEADING

   J = MP+1
   GO 230 I=J,M
230   JH(I) = 0
   GO TO 725

   240 CONTINUE

   TEST COLUMN 67-72 FOR TEST WORD

   CALL SEARCH( KBCD(15),KBCD2(15),II,4,32H)

   IFRAM...
   GO TO ( 250,360,420,410,530 ),II
250   CONTINUE
   KM(46) = 0
   DO 350 I=5,15,2
   KM(45) = KBCD(I)
   KM2(45) = KBCD2(I)
350   CONTINUE
   IF( KM(45) .EQ. KM(48) .AND. KM2(45) .EQ. KM2(48) ) GO TO 290
   DO 270 J=1,IIA
   IF( KM(45) .EQ. KN(J) .AND. KM2(45) .EQ. KN2(J) ) GO TO 310
270   CONTINUE

   CHECK FOR FREE VARIABLES OR PARAMETRIC

   J = II A
   DO 280 JJ=IIB,NL
   IF( KM(45) .EQ. KN(JJ) .AND. KM2(45) .EQ. KN2(JJ) ) GO TO 340
280   CONTINUE
   WRITE( NOT,96 ) KM(45),KM2(45)
290   JJ = 0
   GO TO 320
310   JJ = J
\end{verbatim}
320 IF(KM(46).EQ.0) GO TO 330
    KM(46) = 0
    GO TO 340
330 CONTINUE
    IF(KBCD(I+1).EQ.KM(20).AND.KBCD2(I+1).EQ.KM(20)) GO TO 340
    IF(KBCD(I+1).EQ.KM(21).AND.KBCD2(I+1).EQ.KM(21)) GO TO 335

    FIND COLUMN NAME
    KM(46) = 1
    KB(JJ) = 1
    KM(45) = KBCD(I+1)
    KM2(45) = KBCD2(I+1)
    GO TO 260
335 JJ = JJ+KMQ(9)
340 KB(JJ) = 1
350 CONTINUE
    GO TO 220
360 CONTINUE

    CARD CONTAINS ROW NAMES FOR ARTIFICIALS
    DO 390 I=5,14
        IF(KBCD(I).EQ.KM(48).AND.KBCD2(I).EQ.KM(48)) GO TO 390
    DO 370 II=MF,MP
        IF(KBCD(I).EQ.NR(II).AND.KBCD2(I).EQ.NR2(II)) GO TO 380
    370 CONTINUE
        WRITE(*,NOT,92) KBCD(I),KBCD2(I)
        I = MP
        GO TO 390
380 JH(II) = 0
390 CONTINUE
    GO TO 220
410 CONTINUE

    PARAMETRIC CONTROL CARD
    KMQ(11) = 1
    N = NQ+1

    DUAL CONTROL WORD
420 CONTINUE
    KMQ(10) = 0
    KM(4) = 0

    READ NAMES OF Y/S AND/OR LAST COLUMN THAT LEFT THE BASIS
    L = 0
430 CONTINUE
    IF(KBCD(L+5).EQ.KM(48).AND.KBCD2(L+5).EQ.KM(48)) GO TO 490
    CALL SEARCH(KBCD(L+6),KBCD2(L+6),II,Z,16H
    GO TO (450,440,480),II
440 CONTINUE
    I = KMQ(7)
    J = NL
    GO TO 460
450 I = 1
    J = KMQ(7)-1
460 CONTINUE
   DO 470 I=I,J
   IF( KBCD(I+5).EQ.KN(I) .AND. KBCD2(I+5).EQ.KN2(I) ) GO TO 510
470 CONTINUE
480 CONTINUE
   WRITE( NOT,901 ) K,K2,(KBCD(I),KBCD2(I),I=5,15)
490 II = 0
510 CONTINUE
   IF( L.EQ.0 ) GO TO 520
   KM(15) = II
   GO TO 220
520 L = 2
   KMQ(14) = II
   GO TO 430
530 CONTINUE

C CONTROL FIELD HAS AN ILLEGAL WORD, TREAT AS A BLANK
C
   WRITE( NOT,902 ) K,K2,(KBCD(I),KBCD2(I),I=5,15)
   GO TO 250

C

C    KK=6 ROWS, COLUMN 12 = $ FOR LINEAR FORM PX

540 CONTINUE
   M = 3
   MF = 4
   NR(1) = KM(48)
   NR2(1) = KM(48)
   NR(2) = KM(48)
   NR2(2) = KM(48)
   NR(3) = KM(48)
   NR2(3) = KM(48)
   LA = 4
   LB = 4

   LB IS LOCATION OF FIRST MINIMUM CONSTRAINT
   LA IS LOCATION OF FIRST EQUALITY
   CODE ORDERS ROW NUMBERS ON +,-,0. THE FIRST ENTRY OF A
   GROUP IS MOVED TO LAST WHEN AN INSERTION HAS TO BE MADE.

550 CONTINUE
   ASSIGN 620 TO KSW
   READ( NB,12 ) K,K2,(KBCD(I),KBCD2(I),I=5,15)
   IF( K.EQ.KM(48) .AND. K2.EQ.KM(48) ) GO TO 560
   MP = M
   GO TO 720
560 CONTINUE
   CALL SEARCH( KBCD(5),KBCD2(5),LC,5,40H + 1 0 $ )
   IF( LC.GT.5 ) GO TO 570
IF ( LC .LT. 5 ) GO TO 580

LC = 5, STORE NAME OF LINEAR FORM

NR(3) = KBCD(6)
NR2(3) = KBCD2(6)
GO TO 550

570 KERR = 1
WRITE( NOT, 903 ) K, K2, KBCD(5), KBCD2(5), KBCD(6), KBCD2(6)
GO TO 550

580 CONTINUE
M = M + 1
IF ( LC .LT. 3 ) GO TO 670, 610, 590

590 CONTINUE
LC = 3
610 ASSIGN 680 TO KSW
620 CONTINUE

FILE ENTRY READ INTO NEXT POSITION, NO SORT

CHANGE TO SORT FOR * IF NEW ENTRY IS --

IF ( LC .EQ. 2 ) ASSIGN 710 TO KSW

640 NR(M) = KBCD(6)
NR2(M) = KBCD2(6)
645 CONTINUE
KM(KC + 1) = KM(KC + 1) + 1
IF ( LC = 2 ) GO TO 650, 660, 550

650 LB = LB + 1
660 LA = LA + 1
GO TO 550
670 CONTINUE

TEST FOR SORT

GO TO KSW+ ( 620, 680, 710 )

680 CONTINUE
NR(M) = NR(LA)
NR2(M) = NR2(LA)
IF ( LC .EQ. 1 ) GO TO 690
NR(LA) = KBCD(6)
NR2(LA) = KBCD2(6)
GO TO 645

690 CONTINUE
NR(LA) = NR(LB)
NR2(LA) = NR2(LB)
NR(LB) = KBCD(6)
NR2(LB) = KBCD2(6)
GO TO 645

710 CONTINUE
IF ( LC .EQ. 1 ) GO TO 680
GO TO 640

RETURN FOR 1, 4, 5

720 IF ( KERR .NE. 0 ) CALL ENDTIE
725 CONTINUE
RETURN
730 CONTINUE
    WRITE( NT6, 300 )
    KM(14) = 0
300 FORMAT(38HOROW ID LIST MISSING, MUST PRECEDE RHS )
    CALL ENDTIE
    RETURN
12 FORMAT(12(A4,A2))
100 FORMAT( 3(A4,A2),F12.6 )
92 FORMAT(3HO  ,A4,A2,24H IS AN ILLEGAL ROW NAME. )
96 FORMAT(3HO  ,A4,A2,28H IS AN ILLEGAL COLUMN NAME. )
901 FORMAT(31HO BASIS# HAS A BAD CONTROL CARD /6X, 12(A4,A2) )
902 FORMAT(45HO ILLEGAL WORD IN COL. 67-72. TREAT AS BLANK /6X,12( 1 A4,A2) )
903 FORMAT(25H  ---ILLEGAL ROW ID --- ,3(A4,A2) )
END

SUBROUTINE VER( A,B,E,IE,IP,JH,KB,KL,KN,LE,NR,X,Y,KLT,KN2,NR2,IA )VER 0010
  IMPLICIT REAL*8 ( A-H,O-Z), INTEGER ( I-N)
  COMMON / XBLANK / NN(19),M,MA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
1 NTAP(4),KBCC(40),MP,NQ,KMQ(15),Z(20),T(10),PLAMDA,PLIMIT
  CALL BYOS,INP --- INVERSION AND CRASH
    KM(23)=0 INVERT CURRENT BASIS
    KM(23)=1 CRASH, INSERT AS MANY VARIABLES AS POSSIBLE
             INTO SOLUTION
    KM(23)=-1 EXTEND BASIS LIST FOR DUAL PROBLEM
             ENTER COMPLEMENT OF NON-BASIC PRIMAL VAR.
             ENTER FREE VARIABLES, THEN INVERT
    KM(23)=-2 RESET BASIS FOR PRIMAL VARIABLES

DIMENSION A(1),B(1),E(1),IE(1),IP(1),JH(1),KB(1),KL(1),KN(1),LE(1),
1 NR(1),X(1),Y(1),KLT(1)
DIMENSION IA(1)
DIMENSION KN2(1),NR2(1)
DIMENSION KM(50),KBCC(20)
INTEGER*2 IE,IP,JH,KB,KL,LE,KLT,IA
EQUIVALENCE (KM(51),KM2(1)) , (KBCC(21),KBCC2(1))


INITIALIZE PARAMETERS

NT6 = NTAP(3)
KM(32) = KM(2)
WRITE( NT6,100 ) KM(2),ME,KM(19),KM(35)
KM(14) = 0
100 FORMAT(29HO REINVERTING AFTER ITERATION ,I5,2H, ,I5,21H TRANSFORMATION
1 TIONS WITH ,I5,18H ENTRIES, TYPE ,I2 )
    MTD = MP+1
IF( KM(23) .EQ. -2 ) GO TO 380
5 CONTINUE
    KBAD = 0
    KM(24) = 0
KM(28) = 0
KMQ(12) = 0

CLEAR HEADINGS IN OUTPUT

KM(10) = KM(48)
KM2(10) = KM(48)
KM(11) = KM(48)
KM2(11) = KM(48)
KM(12) = KM(48)
KM2(12) = KM(48)
KM(13) = KM(48)
KM2(13) = KM(48)
MA = 0
MB = 0
MC = 0
MF = KMQ(1)+1

10 CONTINUE
   IF( KMQ(10) ) 20,40,30
20 CONTINUE
   KMQ(10) = 1
   KM(7) = 0
   SET RANGES FOR SUB PROBLEM
30 CONTINUE
   MM = MP
   NL = KMQ(7)-1
   NS = NL
   KM(4) = 1
   GO TO 80
40 CONTINUE
   SET RANGES FOR FULL PROBLEM
   NL = N
   NS = KMQ(7)+KMQ(1)-2
   MM = M
   IF( KMQ(13).NE.0 ) GO TO 80
   EXTEND B(I) FOR DUAL
   DO 50 I=MTO,M
50 B(I) = 0.0
   MTC = KMQ(1)-1
   FIND ENTRIES FOR LINEAR FORM
   DO 70 J=1,MTC
   MTA = KL(J)
   MTB = KL(J+1)-1
   DO 60 JJ=MTA,MTB
60       I = IA(JJ)
   IF( I.NE.3 ) GO TO 60
   I = J+KMQ(9)
   ****
   JA CALL

I = IA(JJ)
   IF( I.NE.3 ) GO TO 60
   I = J+KMQ(9)
I = KL(I)
B(I) = A(JJ)*PLAMDA
GO TO 70
60 CONTINUE
70 CONTINUE
KMQ(13) = 1

TEST KM(23) SWITCH
80 CONTINUE
IF( KM(23) ) 90,230,170
90 CONTINUE

EXTEND BASIS LIST FOR DUAL
INSERT COMPLEMENT OF NON-BASIS VARIABLE

MTC = KMQ(7)-1
DO 120 J=1,MTC
JJ = J+KMQ(9)
IF( KB(JJ).GT.0 ) GO TO 110
KB(J) = 0
KB(JJ) = -12345
GO TO 120
110 CONTINUE
KB(J) = -12345
KB(JJ) = 0
120 CONTINUE

INSERT FREE VARIABLES

MTA = KMQ(8)
IF( MTA.EQ.0 ) GO TO 140
DO 130 J=MTA,N
130 KB(J) = -12345
140 CONTINUE

EXTEND JH(I) LIST, CHECK FOR PRIMAL ARTIFICIALS
DELETE FREE VARIABLE FOR ARTIFICIAL ROW

DO 160 I=MF,M
IF( JH(I).GT.0 ) GO TO 150
JH(I) = 0
IF( I.GT.MP ) GO TO 150
J = KMQ(8)+1-4-KMQ(2)-KMQ(3)
KB(J) = 0
GO TO 160
150 JH(I) = 12345
160 CONTINUE
GO TO 230
170 CONTINUE

INITIALIZE BASIS FOR CRASH

DO 180 J=1,NL
IF( KB(J).LE.0 ) KB(J) = -12345
180 CONTINUE
IF( LE(I).EQ.0 ) GO TO 210

SET JH LIST FOR CRASH AFTER INITIAL INVERSION
DO 190 I=MF,MP
IF( JH(I).LE.0 ) JH(I) = 12345
190 CONTINUE
GO TO 270
210 CONTINUE
DO 220 I=MF,MP
220 JH(I) = 12345

INVERSION ENTRY

230 CONTINUE
ME = 0
KM(17) = MM-MF+1
KM(19) = 0
KM(31) = 0
KM(44) = 0
Z(9) = 1.0
LE(1) = 1
DO 240 I=1,MM
240 X(I) = B(I)
IF( KM(23).NE.0 ) GO TO 270

SET KB AND JH LISTS FOR INVERSION

DO 250 J=1,NL
II = 0
IF( KB(J).GT.0 ) II = -12345
KB(J) = II
250 CONTINUE
DO 260 I=MF,MM
II = 0
IF( JH(I).GT.0 ) II = 12345
JH(I) = II
260 CONTINUE
270 CONTINUE

CLEAR DUMMY VARIABLES

J = KMQ(1)
JJ = J+KMQ(9)
KB(J) = 0
KB(JJ) = 0

INSTALL BASIC SLACKS FIRST

IF( KMQ(10).NE.0.AND.KMQ(2).EQ.-KMQ(3) ) GO TO 290
DO 280 J=NF,NS
II = KL(J)
IF( KB(J).GE.0.OR.JH(II).NE.12345 ) GO TO 280
I = II
KB(J) = I
JH(I) = J
KM(17) = KM(17)-1
KM(44) = KM(44)+1
280 CONTINUE
290 CONTINUE

FORM INVERSE ----- LOOK FOR SHORTEST COLUMN
C
IF( KM(17).EQ.0 ) GO TO 330
KM(45) = 12345
DO 310 JJ=1,NL
IF( KB(JJ).GE.0.OR.KM(45).LE.KLT(JJ) ) GO TO 310
KM(45) = KLT(JJ)
J = JJ
310 CONTINUE
IF( KM(45).EQ.12345 ) GO TO 330
KB(J) = 0
CALL EXP( J,A,B,KL,NR,Y,NR2,IA )
CALL JMY( E,IP,LE,Y,IE )

CHOOSE PIVOT
IR = 0
T(2) = Z(I)
DO 320 I=MF,MM

****** ABS CALL
IF( JH(I).NE.12345.OR.DABS(Y(I)).LE.T(2) ) GO TO 320

****** ABS CALL
T(2) = DABS(Y(I))
IR = I
320 CONTINUE
IF( IR.EQ.0 ) GO TO 420

PIVOT
JH(IR) = J
KB(J) = IR
KM(17) = KM(17)-1
CALL PIV( IR,E,IE,IP,LE,X,Y )
IF( KM(24).EQ.0 ) GO TO 290

RESET ARTIFICIALS
330 CONTINUE
MTA = 0
DO 340 I=MF,MM
IF( JH(I).NE.12345.AND.JH(I).NE.0 ) GO TO 340
JH(I) = 0
MTA = 1
340 CONTINUE
RESET BASIS HEADINGS

DO 350 J=1,N
IF( KB(J).LT.0 ) KB(J) = 0
350 CONTINUE

IF( KM(24).EQ.0.AND.KM(2).EQ.0 ) GO TO 370
355 CONTINUE
WRITE( NT6,200 ) KM(44),KBAD,ME,KM(19),KM(14) = 0
200 FORMAT( 21H *INVERSION COMPLETED, I5, 8H SLACKS, I5, 9H POOR COLS, VER 2750)
   I I5, 22H TRANSFORMATIONS WITH, I5, 9H ENTRIES /1x )
   VER 2760
   VER 2770
   VER 2780
   VER 2790
   VER 2800
   VER 2810
   VER 2820
   VER 2830
   VER 2840
   VER 2850
   VER 2860
   VER 2870
   VER 2880
   VER 2990
   VER 2900
   VER 2910
   VER 2920
   VER 2930
   VER 2940
   VER 2950
   VER 2960
   VER 2970
   VER 2980
   VER 2990
   VER 3000
   VER 3010
   VER 3020
   VER 3030
   VER 3040
   VER 3050
   VER 3060
   VER 3070
   VER 3080
   VER 3090
   VER 3100
   VER 3110
   VER 3120
   VER 3130
   VER 3140
   VER 3150
   VER 3160
   VER 3170
   VER 3180
   VER 3190
   VER 3200
   VER 3210
   VER 3220
   VER 3230
   VER 3240
   VER 3250
   VER 3260
   VER 3270
   VER 3280
   VER 3290
   VER 3300
   VER 3310
   VER 3320
   VER 3330

WRITE( NT6, 300 )

300 FORMAT( 29H 1-TCO MANY ENTRIES IN INVERSE )
   KM(14) = 1
   KM(19) = LE(ME)-1
   ME = ME-1
   CALL UOUT( 7 )
   CALL ENDTIE
   360 CONTINUE
   RETURN

370 CONTINUE
   IF( KMQ(TO).EQ.0 OR MTA.EQ.0 ) GO TO 355
   IF( KM(TO).EQ.1 ) GO TO 355

   AT ITERATION ZER0 IF SUBPROBLEM HAS ARTIFICIALS IN THE BASIS,
   TRY TO CRASH IN A FULL BASIS

   KM(TO) = 1
   GO TO 170

   ( STATEMENTS DELETED FOR CARDS )

   SOLUTION FOR FULL PROBLEM IS INFEASIBLE

380 CONTINUE
   WRITE( NT6, 400 )
   KMQ(14) = 0

400 FORMAT( 78H 1-SOLUTION FOR FULL PROBLEM IS INFEASIBLE. DELETE DUAL
   1-VARIABLES AND REINVERT. )

   RESET BASIS AND PARAMETERS FOR PRIMAL PROBLEM

   KMQ(TO) = 1
   KMQ(N) = 0
   KM(TO) = 0
   DO 390 I=MTD,M
      390 JH(I) = 0
      I = KMQ(7)
   DO 410 J=1,N
      410 KB(J) = 0
      N = NQ
   GO TO 5

420 CONTINUE

TALLY NUMBER OF POOR PIVOTS
KBAD = KBAD+1

PRINT NAME OF REJECTED COLUMN IF NOT CRASHING

IF( KM(TO).EQ.1 ) GO TO 290
K = 0
IF( J .GE. KMQ(7) ) K = 1
WRITE( NT6,500 ) KN(J),KN2(J),KM(K+20),KM2(K+20)
NM14 = 0
500 FORMAT(28HO INVERSION REJECTS COLUMN ,2(A4,A2) )
GO TO 290
END

SUBROUTINE ERR( A,B+JH,KB,KL,KN,NR,P,X,Y,KN2,NR2,IA )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19),M,MA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
1 NTAP(4),KBCC(40),MP,NQ,KMQ(15),Z(20),T(10),PLAMDA,PLIMIT
CALLED BY BOS, AND INP, CALCULATES ROW ERRORS, STORES IN P
DIMENSION AI(1),BI(1),JH(1),KB(1),KL(1),KN(1),NR(1),P(1),X(1),Y(1)
DIMENSION IA(1)
DIMENSION KN2(1),NR2(1)
DIMENSION KM(50),KBCC(20)
EQUIVALENCE (KM(51),KMQ(1)), (KBCC(21),KBCC(21))
INTEGER*2 JH,KB,KL,IA

NT6 = NTAP(3)
CALL IMAGE( KM(12),KM2(12),1,BERRORS )
MC = 0

SET RANGES FOR COLUMN TYPES

NF = KMQ(1)+1
ND = KMQ(7)+KMQ(1)
NL = ND-2
NP = KMQ(1)-1
NQ = NQ+1
NXXX = 0
XINXXX = 0.0

STORE -B(I) IN P(I) AND COMPUTE VALUES FOR ARTIFICIALS

DO 10 I=1,M
P(I) = -B(I)
IF( JH(I),EQ,0 ) P(I) = P(I)+X(I)
10 CONTINUE

COMPUE VALUES FOR SLACKS

DO 20 J=NF,NL
IF( KB(J),LE,0 ) GO TO 20
I = KB(J)
IB = KL(J)
P(IB) = P(IB)+X(I)
20 CONTINUE

COMPUTE ROW 2 FOR DUAL CONSTRAINTS

IF( KMQ(10),NE,0 ) GO TO 40
DO 30 J=ND,NQ
IF( KB(J),LE,0 ) GO TO 30
VER 3340
VER 3350
VER 3360
VER 3370
VER 3380
I = KB(J)
IB = KL(J)
P(2) = P(2) + B(IB) * X(I)
30 CONTINUE
ASSIGN 60 TO KDUAL
COMPUTE ROW 1 FOR PARAMETRIC
IF( KMQ(I1) .LE. 0 ) GO TO 50
I = KB(N)
P(I1) = P(I1) - X(I)
GO TO 50
40 CONTINUE
ASSIGN 130 TO KDUAL
50 CONTINUE
COMPUTE MATRIX TIMES SOLUTION
DO 140 J = 1, NP
I = KB(J)
T(I) = 0.0
IF( I .GT. 0 ) T(I) = X(I)
JD = J + KMQ(9)
JD = KL(JD)
MTA = KL(J)
MTB = KL(J+1) - 1
DO 130 K = MTA, MTB
****** JAN CALL
I = IA(K)
P(I) = P(I) + T(I) * A(K)
GO TO KDUAL + ( 60, 130 )
60 CONTINUE
CHECK FOR QUADRATIC OR DUAL OF ROW
IF( I .LE. MP ) GO TO 90
QUADRATIC ENTRY, CHECK FOR DIAGONAL ELEMENT
IF( I .EQ. JD ) GO TO 130
70 CONTINUE
JD = IA(NP)
IF( NR(I) .EQ. KN(JJ) .AND. NR2(I) .EQ. KN2(JJ) ) GO TO 80
DO 70 JJ = 1, NP
80 CONTINUE
JJ = NP
80 CONTINUE
90 CONTINUE
FIND TRANSPOSED ROW
DO 110 JI = NO, NCP
IF( I .EQ. KL(JI) ) GO TO 120
110 CONTINUE
JI = NCP
120 CONTINUE
T(2) = - A(K)
125 CONTINUE
   IF ( ID.GT.0 ) P(JD) = P(JD) + X(ID)*T(2)
130 CONTINUE
140 CONTINUE
   FIND SUM AND MAXIMUM OF ABSOLUTE VALUES OF ERRORS
   T(1) = 0.0
   T(2) = 0.0
   ID = 1
   MM = M
   IF ( KMQ(10).NE.0 ) MM = MP
   DO 150 I=1,MM

   150 CONTINUE
   ABS CALL
   T(1) = T(1) + CAES(P(I))
   ABS CALL
   IF ( DABS(T(2)).GE.DABS(P(I)) ) GC TO 150
   T(2) = P(I)
   ID = I

150 CONTINUE
   OUTPUT ERROR STATEMENTS
   WRITE( NT6,100 ) NR(IDC),NR2(ID),T(2),T(1)
   KM(14) = 0

100 FORMAT(25H0MAXIMUM ERROR IS ON RCW ,A4,A2,ZH =,1PE10.2/33HOSUM OF
   LABSOLUTE ERRORS ON RCWS = ,1PE10.2 /1H0 )
   RETURN
END

SUBROUTINE CUP(JH,KN,NR,KN2,NR2)
IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
COMMON/XBLANK/NN(19),M,NA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
1 NTAP(4),KBCC(40),MP,NQ,KMQ(15),Z(20),T(10),PLAMDA,PLIMIT

CUP CALLED BY INP AND GQT — PUNCHES CARDS

PUNCH BASIS CARDS, MARK DUAL VARIABLES
PUNCH KMQ(14) AND KMQ(15) FOR NAME OF Y/S COLUMN AND LAST COLUMN
DROPPED FROM BASIS
COLUMNS 67-72 ARE BLANK FOR BASIS HEADINGS
CONTROL WORDS ARE ARIF., DUAL.., PARAM.

DIMENSION JH(1),KN(1),NR(1)
DIMENSION KN2(1),NR2(1)
DIMENSION KM(50),KBCC(20)
EQUIVALENCE (KM(51),KM2(1)),(KBCC(21),KBCC2(1))
INTEGER*2 JH

NT = 7

OUT 0010
OUT 0020
OUT 0030
OUT 0040
OUT 0050
OUT 0060
OUT 0070
OUT 0080
OUT 0090
OUT 0100
OUT 0110
OUT 0120
OUT 0130
OUT 0140
OUT 0150
OUT 0160
OUT 0170
OUT 0180
OUT 0190
OUT 0200
OUT 0210
WRITE( NT, 901 ) KM(I), KM2(I), KM(3), KM2(3), KM(2), KM(16)
NTAL = -1
MM = M
KBCD(I1) = KM(48)
KBCD2(I1) = KM(48)

SET UP CONTROL WORDS IN BCD(17-19)

DO 10 I=1,3
CALL IMAGE( KBCD(I+16), KBCD2(I+16), I, 24HARTIF, DUAL, PARAM... )
10 CONTINUE
IF( KMC(10) .NE. 0 ) MM = MP
DO 20 I=MP, MM
IF( JHI(EQ.0) ) GO TO 20
NTAL = NTAL+2
JHI = JH(I)
KBCD(NTAL) = KN(JHI)
KBCD2(NTAL) = KN2(JHI)

STORE BLANK FOR PRIMAL, ..DUAL FOR DUAL

KBCD(NTAL+1) = KM(21)
KBCD2(NTAL+1) = KN2(21)
IF( JHI .LT. KMQ(7) ) KBCD(NTAL+1) = KM(20)
IF( JHI .LT. KMQ(7) ) KBCD2(NTAL+1) = KM2(20)
IF( NTAL .LT. 9 ) GO TO 20
WRITE( NT, 902 ) ( KBCD(J), KBCD2(J), J=1, 11 )
NTAL = 1
20 CONTINUE
IF( NTAL .LE. 9 ) GO TO 25
NTAL = NTAL+1
WRITE( NT, 902 ) ( KBCD(I), KBCD2(I), I=1, NTAL )

CHECK FOR ARTIFICIALS

25 CONTINUE
NTAL = 0
KBCD(I1) = KBCD(I7)
KBCD2(I1) = KBCD2(I7)
DO 30 I=MP, MM
IF( JHI .NE. 0 ) GO TO 30
NTAL = NTAL+1
K3CD(NTAL) = NN(I)
K3CD2(NTAL) = NN2(I)
IF( NTAL .LT. 9 ) GO TO 30
WRITE( NT, 902 ) ( KBCD(J), KBCD2(J), J=1, 11 )
NTAL = 0
30 CONTINUE
IF( NTAL .EQ. 0 ) GO TO 50

CLEAR EXTRA FIELDS FOR LAST ARTIF. CARD

NTAL = NTAL+1
DO 40 I=NTAL, 10
KBCD2(I) = KM(48)
40 KBCD(I) = KM(48)
WRITE( NT, 902 ) ( KBCD(J), KBCD2(J), J=1, 11 )
50 CONTINUE
CLEAR CARD IMAGE

DO 60 J=1,11
   KBOD2(J) = KM2(48)
60  KBOD(J) = KM(48)

CHECK FOR PARAMETRIC

IF( KMQ(11) .LT. 0. AND. KMQ(10) .NE. 0 ) GO TO 70
   KBOD(11) = KBOD(19)
   KBOD2(11) = KBOD2(19)
   IF( KMQ(11) .LT. 0 ) KBOD(11) = KBOD(18)
   IF( KMQ(11) .LT. 0 ) KBOD2(11) = KBOD2(18)
   JHI = KMQ(14)
   KBOD(1) = KN(JHI)
   KBOD2(1) = KN2(JHI)
   KBOD(2) = KM(20)
   KBOD2(2) = KM(20)
   IF( JHI .GE. KMQ(7) ) KBOD(2) = KM(21)
   IF( JHI .GE. KMQ(7) ) KBOD2(2) = KM(21)
   JHI = KM(15)
   KBOD(3) = KN(JHI)
   KBOD2(3) = KN2(JHI)
   KBOD(4) = KM(20)
   KBOD2(4) = KM(20)
   IF( JHI .GE. KMQ(7) ) KBOD(4) = KM(21)
   IF( JHI .GE. KMQ(7) ) KBOD2(4) = KM(21)
   WRITE( NT, 902 ) ( KBOD(I), KBOD2(I), I=1,11 )
70  CONTINUE
   WRITE( NT, 904 )
   RETURN

901  FORMAT( 6HBASIS*, 3Z(3X,A4,A2), 3X, 9HITERATION ,I6,4H (,12,1H) )
902  FORMAT( 6X,11(A4,A2) )
904  FORMAT( 3HEND )

END

SUBROUTINE SOT( JH,KB,NR,X,NR2 )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON /XBLANK/ NN(19), N, MA, MB, MC, MD, ME, MF, N, KM(100), KP(12),
1 NTAP(4), KBOD(40), MP, NQ, KMQ(15), Z(20), T(10), PLAMDA, PLIMIT

CALLED BY QOT     SHORT OUTPUT
     PRINTS TOTAL OBJECTIVE, LINEAR, QUADRATIC TERMS, AND LAMBDAS

DIMENSION JH(1), X(1), NR(1), KB(1)
DIMENSION NR2(1)
DIMENSION KM2(50), KBOD2(20)
EQUIVALENCE (KM(51), KM2(1)) , (KBOD(21), KBOD2(1))
INTEGER*2 JH, KB
LOGICAL FLAG

KM(29) = 0

SOT 0010
SOT 0020
SOT 0030
SOT 0040
SOT 0050
SOT 0060
SOT 0070
SOT 0080
SOT 0090
SOT 0100
SOT 0110
SOT 0120
SOT 0130
SOT 0140
SOT 0150
SOT 0160
SOT 0170
SOT 0180
T(2) = 0.0
NT6 = NTAP(3)
MM = M

IF (KMQ(10).NE.0) MM = MP

COMPUTE INFEASIBILITIES

DO 10 I = MF, MM
  TEMP = X(I)
  IF (TEMP.EQ.0.) GO TO 10
  FLAG = .TRUE.
  IF (TEMP.LT.0.0) FLAG = .FALSE.
  IF (FLAG .AND. JH(I).NE.0.) GO TO 10
  IF (.NOT. FLAG .AND. KMQ(8).NE.0 .AND. JH(I).GE.KMQ(8)) GO TO 10
  IF (.NOT. FLAG) T(2) = T(2) + X(I)
  IF (.NOT. FLAG) T(2) = T(2) - X(I)
  KM(29) = KM(29) + 1
10 CONTINUE

PRINT ALL PARTS OF THE OBJECTIVE
COMPUTE LAMBDA, LINEAR AND QUADRATIC TERMS OF OBJECTIVE.

T(1) = LAMBDA = PLAMDA + X(1)
T(6) = LINEAR (PX) = -X(3)
T(5) = -PX*LAMBDA = X(3)T(I)
T(3) = QUADRATIC (X*QX) = 0.5*(X(2)*T(5) + X(2)*Y(S))
T(4) = TOTAL OBJECTIVE = T(3) - T(5)

X(3) = -PX
X(2) = -RHS TRANSPOSED * TRANSPOSED PRIMAL ROW VARIABLES

NXXXX = 0
X(NXXXX) = 0.0
I = KMQ(14)
J = KMQ(9)
I = KB(I)
J = KB(J)
T(6) = -X(3)
T(1) = PLAMDA + X(1)
T(5) = X(3)T(I)
T(3) = 0.5*(X(2)*T(5) + X(2)*X(J))
T(4) = T(3) - T(5)

WRITE( NT6,901 ) KM(16),KM(29),KM(1),KM(2),KM(3),KM(2),
1 KM(2),KM(26),T(2),Z(9),KM(31),KM(13),KM(213),KM(MA+20),
2 KM(MA+20),KM(10),KM(210),KM(MB+20),KM(MB+20),KM(11),KM(211)
WRITE( NT6,902 ) T(4),T(1),T(6),T(3)
RETURN

901 FORMAT(2HQ(12,30H) MATRIX R.H.S. ITER PIVS, I7,61H INFEAS
1 DETERMINANT NEW COLUMN OLD COLUMN PIVOT ROW / 6X, A4,
2 A2,2X,A4,A2,17,17,F13.3,F11.5,1HE,13,3X,A4,A2,A4,A2,2X,A4,
3 A2,A4,A2,4X,A4,A2 // )
902 FORMAT(13X,19HC NCBJECTIVE TOTAL = 8X, 6H L7X, 1H+, 5X, 37HLINEAR
1 (PX) + QUADRATIC (X*QX) / 9X,F19.6,4H =, F19.6,
2 3H *,F17.6,4X,1H+,F19.6 / 1X )
END
SUBROUTINE PCOT( JH,KB,KN,X,KN2 ) 
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N) 
COMMON /XBLANK/ NN,(19),M,MAM,MB,MC,MD,ME,ME,MF,N,KM(100),KP(12), 
1 NTAP(4),KBCC(40),MP,NQ,KMQ(15),Z(20),T(10),PLAMDA,PLIMIT 
C C C 
PDOT CALLED BY QOT. PRINTS SOLUTION FOR PRIMAL AND DUAL VARIABLES 
C C C 
DIMENSION JH(1),KB(1),KN(1),X(1) 
DIMENSION KN2(1) 
DIMENSION KMZ(50),KBCC2(20) 
EQUIVALENCE (KM(51),KM2(1)) , (KBCC(21),KBCC2(1)) 
INTEGER*2 JH,KB 
C C C 
NT6 = NTAP(3) 
WRITE( NT6,900 ) 
C C C 
INITIALIZE CONSTANTS 
C C C 
I = 0 
IL = KMQ(7)-1 
J = IL 
JL = NQ+1. 
ASSIGN 50 TO K 
5 IF( I.LT.0 ) GO TO 25 
10 CONTINUE 
I = I+1 
IF( I.LE.IL ) GO TO 20 
IF( J.LT.0 ) RETURN 
I = -1 
ASSIGN 60 TO K 
GO TO 25 
20 CONTINUE 
C C C C C C 
FIND NEXT PRIMAL BASIC VARIABLE 
IF( KB(I).LE.0 ) GO TO 10 
II = KB(I) 
C C C C C C 
FIND NEXT DUAL VARIABLE 
25 IF( J.LT.0 ) GO TO 45 
30 CONTINUE 
J = J + 1 
IF( J.LE.JL ) GO TO 40 
IF( I.LT.0 ) RETURN 
J = -1 
ASSIGN 70 TO K 
GO TO 45 
40 CONTINUE 
IF( KB(J).LE.0 ) GO TO 30 
JJ = KB(J) 
45 GO TO K, (50,60,70) 
50 CONTINUE
PRINT LINE - IN ONE OF THREE FORMATS

WRITE( NT6,901 ) KN(I),KN2(I),X(I),KN(J),KN2(J),X(J)
GO TO 5
60 CONTINUE
WRITE( NT6,902 ) KN(J),KN2(J),X(J)
GO TO 25
70 CONTINUE
WRITE( NT6,903 ) KN(I),KN2(I),X(I)
GO TO 5
900 FORMAT( 28HO PRIMAL VARIABLES SOLUTION,20X,24HDUAL VARIABLES SOL)
INATION / 4X,4HNAME,13X,5HVALUE,24X,4HNAME,14X,5HVALUE / 1X )
901 FORMAT( 3X,A4,A2,F20.6,20X,A4,A2,F20.6 )
902 FORMAT( 49X,A4,A2,F20.6 )
903 FORMAT( 3X,A4,A2,F20.6 )
END

SUBROUTINE QOT( K,A,B,JH,KB,KL,KN,NR,P,X,Y,KN2,NR2,IA )
IMPLICIT REAL*8 (A-H,O-Z),INTEGER (I-N)
COMMON / XBLANK / NN(19),M,MA,MB,MC,MD,ME,MF,N,NM(100),KP(12),
1 NTAP(4),KBCC(40),MP,NQ,KNM(15),Z(20),T(10),PLAM2A,PLIMIT
DIMENSION A(1),B(1),JH(1),KB(1),KL(1),KN(1),KR(1),P(1),X(1),Y(1)
DIMENSION NR(1)
DIMENSION KN2(1),NR2(1)
DIMENSION KM2(50),KBCC2(20)
EQUIVALENCE (KM(51),KM2(1)),(KBCC(21),KBCC2(1))
INTEGER*2 JH,KB,KL,IA

CALLED BY UGUT     OUTPUT CONTROL

SET CONDITION NUMBER

KM(16) = K
N6 = NTAP(3)

RESET QOT(6) COUNTER

IF( KM(16).EQ.6 ) KM(25) = 0

TEST IF PUNCHING IS WANTED

KD = KP(K)
IF( KD.LT.5 ) GO TO 5
KD = KD-5
CALL OUP( JH,KN,NR,KN2,NR2 )

EJECT PAGE BEFORE LONGER OUTPUT

5 IF( KD+1 ) .110,30,10
10 IF( KM(14).NE.0 ) GO TO 40
ASSIGN 40 TO KEJECT
20 CONTINUE
WRITE( N6,92 )
GO TO KEJECT, ( 40,90,60 )

30 CONTINUE

CONTROL 1. PAGE WILL NOT BE RESTORED

KM(14) = 0
40 CALL SOT( JH,KB,NR,X,NR2)
  IFI MOD(KD,2) EQ 0 ) CALL LOT( B,JH,KN,NR,P,X,Y,KN2,NR2)
  IFI KD=2 ) 110,80,50
50 IFI KD.EQ.3 ) GO TO 70
  ASSIGN 60 TO KEJECT
  GO TO 20
60 CONTINUE
  CALL SOT( JH,KB,NR,X,NR2)
70 CONTINUE
  CALL PACT( JH,KB,KN,X,KN2)

EJECT PAGE

80 ASSIGN 90 TO KEJECT
  GO TO 20
90 CONTINUE

NOTE PAGE EJECTED

KM(14) = 1
110 RETURN
92 FORMAT(1H1)
END

SUBROUTINE DEL( JD,A,KL,P,Y,IA )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19), MA, MB, MC, MD, ME, MF, N, KM(100), KP(12),
1 NTAP(4), KBCD(40), MP, NQ, KMQ(15), Z(20), T(10), PLAMDA, PLIMIT

DEL FOR QP
DEL IS CALLED ONLY IF WORKING ON PRIMAL SUBPROBLEM

DIMENSION A(1), KL(1), P(1), Y(1)
DIMENSION KMZ(50), KBCD(20)
EQUIVALENCE (KM(51), KMZ(1) ), (KBCD(21), KBCDZ(21))
DIMENSION IA(1)
INTEGER*2 KL, IA

DT = 0.0

TEST FOR DUMMY PRIMAL VARIABLE

IFI( JD.EQ.KMQ(1).OR.(J.GT.KMQ(1).AND.J.GE.KMQ(7)) ) GO TO 20
IFI( J.GT.KMQ(1) ) GO TO 30
MTA = KL(J)
MTB = KL(J+1) - 1
DO 10 K=MTA, MTB

***** JA CALL

DEL 0010
DEL 0020
DEL 0030
DEL 0040
DEL 0050
DEL 0060
DEL 0070
DEL 0080
DEL 0090
DEL 0100
DEL 0110
DEL 0120
DEL 0130
DEL 0140
DEL 0150
DEL 0160
DEL 0170
DEL 0180
DEL 0190
DEL 0200
DEL 0210
DEL 0220
DEL 0230
DEL 0240
DEL 0250
DEL 0260
IN = IA(K)  
IF( P(IN),NE.0.0 ) DT = DT+P(IN)*A(K)  
10 CONTINUE  
20 CONTINUE  
D = DT  
RETURN  

COLUMN J A SLACK VARIABLE  
30 CONTINUE  
':'R = KL(J)  
DT = P(R)  
GO TO 20  
ENO  

SUBROUTINE EXP( J, A, B, KL, NR, Y, NR2, IA )  
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)  
COMMON / XBLANK / NN(19), M, MA, MB, MC, MD, ME, MF, N, KM(10), KP(12),  
1 NTAP(4), KBGD(40), MP, MQ, KMQ(15), Z(20), T(10), PLAMDA, PLIMIT  
CALLED BY BOS, VER  
EXPAND MATRIX COLUMN IN Y FOR USE BY JMY  
DIMENSION A(1), B(1), KL(1), NR(I), Y(I)  
DIMENSION NR2(1)  
DIMENSION KM2(50), KBGD2(20)  
EQUIVALENCE (KM(51), KM2(I)), (KBGD(21), KBGD2(I))  
DIMENSION IA(I)  
INTEGER*2 KL, IA  

JX = KMQ(I)  
MM = MP+1  
DO 10 I=1, M  
10 Y(I) = 0.0  
IF( J.GT.KMQ(I) ) GO TO 120  
MTA = KL(J)  
MTB = KL(J+1)-1  
DO 20 K=MTA, MTB  

****** JA CALL  

I = IA(K)  
Y(I) = A(K)  
20 CONTINUE  
IF( KMQ(10) .EQ. 0 ) GO TO 50  
FOR SUB PROBLEM CLEAR QUADRATIC TERMS  
DO 30 I=MM,M  
30 Y(I) = 0.0  
40 CONTINUE  
RETURN  
50 CONTINUE  

FIND QUADRATIC SYMETRIC TERMS, FIND ROW NO. FOR COL.
IJ = J+KMQ(9)
KM(45) = KL(IJ)
60 CONTINUE
DO 90 JJ=1,JX
IF( J.EQ.JJ ) GO TO 90
MTA = KL(JJ)
MTB = KL(JJ+1)-1
DO 70 K=MTA,MTB
70 CONTINUE

****** JA CALL

I = IA(K)
IF( I.EQ.KM(45) ) GO TO 80
70 CONTINUE
GO TO 90
80 CONTINUE
IJ = JJ+KMQ(9)
JI = KL(IJ)
Y(JI) = A(K)
90 CONTINUE
IF( J.LE.KMQ(1) ) GO TO 40

REVERSE SIGN ON TRANPOSED ROW AND PICK UP B(I)

DO 110 I=M,M
110 Y(I) = -Y(I)
1 = KM(45)
Y(2) = B(I)
IF( J.EQ.NQ+1 ) Y(I) = -1.0
GO TO 40
120 CONTINUE

J IS NOT PART OF THE INPUT MATRIX

IF( J.GE.KMQ(1)+KMQ(7) ) GO TO 130
1 = KL(J)
Y(I) = 1.0
GO TO 40
130 CONTINUE

FIND TRANPOSED ROW

KM(45) = KL(J)
GO TO 60
END

SUBROUTINE BEGIN( IC,K )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19),M,MA,MH,MC,MD,ME,MF,N,KM(100),VP(12),
1 NTAP(4),KBCC(40),MP,NQ,KMQ(15),Z(20),T(10),PLAMCA,PLIMIT
2 ,A(2500)
4 ,KLAST
2 ,A(6000),IA(6004),KL(1104),KN(1102),KN2(1102),NR(512),NR2(512),
3 B(511),KLT(1104),KNR(1104)

EXP 0410
EXP 0420
EXP 0430
EXP 0440
EXP 0450
EXP 0460
EXP 0470
EXP 0480
EXP 0490
EXP 0500
EXP 0510
EXP 0520
EXP 0530
EXP 0540
EXP 0550
EXP 0560
EXP 0570
EXP 0580
EXP 0590
EXP 0600
EXP 0610
EXP 0620
EXP 0630
EXP 0640
EXP 0650
EXP 0660
EXP 0670
EXP 0680
EXP 0690
EXP 0700
EXP 0710
EXP 0720
EXP 0730
EXP 0740
EXP 0750
EXP 0760
EXP 0770
EXP 0780
EXP 0790
EXP 0800
EXP 0810
EXP 0820
EXP 0830
EXP 0840
EXP 0850
EXP 0860
BGN 0010
BGN 0020
BGN 0030
BGN 0040
BGN 0050
BGN 0060
BGN 0070
BGN 0080
DIMENSION KM2(50), KBCD2(20)
EQUIVALENCE (KM(51), KM2(1)), (KBCD(21), KBCD2(1))
INTEGER*2 IA, KL, KLT, KNR

BEGIN COMPUTES THE NUMBER OF STORAGE WORDS FROM A GIVEN VARIABLE
LOCATION TO THE END OF MEMORY.
IGETAD OBTAINS THE ADDRESS OF A VARIABLE. KLAST IS THE END OF
COMMON.

K = IGETAD(KLAST) - IC
RETURN
END

SUBROUTINE IMGAGE( KA, KB, L, KC )

DIMENSION KC(1)
KA = KC(2*L-1)
KB = KC(2*L)
RETURN
END

SUBROUTINE GET( IR, E, IP, JH, LE, P, X, IE )
IMPLICIT REAL*8 (A-H, O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19), M, MA, MB, MC, MD, ME, MF, N, KM(100), KP(12)
1 NTAP(4), KBCD(40), MP, NO, KMQ(15), Z(20), T(10), PLAMDA, PLIMIT

CALLED BY B05, ERR
GET PRICES FOR CP
GET IS CALLED ONLY IF WORKING ON PRIMAL SUBPROBLEM.
KMQ(10) NEGATIVE, COMPUTE ROW OF INVERSE FOR FIRST ARTIFICIAL ROW

DIMENSION E(1), JH(1), LE(1), P(1), X(1), IP(1)
DIMENSION KM2(50), KBCD2(20)
EQUIVALENCE (KM(51), KM2(1)), (KBCD(21), KBCD2(1))
DIMENSION IE(1)
INTEGER*2 IP, JH, LE, IE

IR = 0
DD 10 I=1, M
10 P(I) = 0.0
DD 30 I=MF*MP
20 CONTINUE
10 CONTINUE
GO TO 30
20 CONTINUE
10 CONTINUE
GO TO 50
40 IR = 1
50 CONTINUE
   IF( ME.EQ.0 ) RETURN
   CALL KRT( E,IP,LE,P,IE )
   RETURN
END

SUBROUTINE TAP( L,IA,KL,KN,NR,B,KLT,KNR,KDIM,KN2,NR2 )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19), M, MA, MB, MC, MD, ME, MF, N, KM(100), KP(12),
   NTAP(4), KB(40), MP, NQ, KMQ(15), Z(20), T(10), PLAMDA, PLIMIT
   A(1)
EQUIVALENCE ( NT6, NTAP(3) )

CALLED BY FIN
INPUT MATRIX
EXTEND KM FOR QP DATA

DIMENSION KDIM(3), L(1), IA(1), KL(1), KN(1), NR(1), B(1), KLT(1), KNR(1)
DIMENSION KN2(1), NR2(1)
DIMENSION KM2(50), KB(20)
EQUIVALENCE ( K(51), KM2(11) )

INTEGER*2 IA, KL, KLT, KNR

: SET = KDIM(3) - 3

IF( KM(50).EQ.0 ) GO TO 10
WRITE( NT6, 100 )
KM(14) = 0
10 FORMAT( 30HOMATRIX READ TWICE IN SAME RUN )
GO TO 290

10 CONTINUE
   NXXXXX = 0
   KN(NXXXXX) = KM(48)
   KN2(NXXXXX) = KM(48)
   NR(NXXXXX) = KM(48)
   NR2(NXXXXX) = KM(48)
   N = 0
   N1 = NTAP(1)
   KM(18) = 0
   K22 = 1
   KM(1) = KM(41)
   KM2(1) = KM2(41)

SET UP ROW NUMBER RANGE FOR MINIMUM TYPE CONSTRAINTS.

IIA = KMQ(2)+4
IIB = IIA+KMQ(3)
MM = MP+1

IF( MM.EQ.0 ) GO TO 360

20 CONTINUE

READ( N1, 900 ) K, K2, J, J2, I, I2, AA
IF( K NE KM(48) OR K2 NE KM2(48) ) GO TO 190

BUMP MATRIX ENTRY COUNT

GET 0300
GET 0310
GET 0320
GET 0330
GET 0340
GET 0350
KM(18) = KM(18)+1
IF( KM(18).GT.KDIM(1) ) GO TO 320
KM18 = KM(18)

LOOK FOR COLUMN NAME

IF( N.GT.0.AND.KN(N).EQ.J.AND.KN2(N).EQ.J2 ) GO TO 60
M = N+1
KL(N) = KM18
KN(N) = J
KN2(N) = J2
IF( N.GT.KDIM(2) ) GO TO 330

IN PRIMAL MATRIX A - CHECK THAT COLUMN NAME DIFFERS FROM ROW NAME

DO 40 JJ=1,M
IF( J.EQ.NR(JJ).AND.J2.EQ.NR2(JJ) ) GO TO 280
40 CONTINUE

IN DUAL MATRIX Q FIND ROW NUMBER FOR COLUMN NAME

DO 50 IR=M,M
IF( J.EQ.NR(IR).AND.J2.EQ.NR2(IR) ) GO TO 60
50 CONTINUE
M = M+1
IF( M.GT.KDIM(3) ) GO TO 340
NR(M) = J
NR2(M) = J2
IR = M
60 CONTINUE
KNR(IR) = KNR(IR)+1

LOOK FOR ROW NAME

DO 70 II=1,M
IF( I.EQ.NR(II).AND.I2.EQ.NR2(II) ) GO TO 80
70 CONTINUE
M = M+1
IF( M.GT.KDIM(3) ) GO TO 340
II = M
NR(M) = I
NR2(M) = I2

80 CONTINUE
IF( II.LT.IIA ) GO TO 170
IF( II.LT.IIB ) GO TO 160
IF( II.LE.MP ) GO TO 170

TEST FOR DIAGONAL ELEMENT IN Q

IF( J.EQ.NR(II).AND.J2.EQ.NR2(II) ) GO TO 120

FIND COLUMN NO. FOR ROW NAME

DO 110 JJ=1,N
IF( I.EQ.KN(JJ).AND.I2.EQ.KN2(JJ) ) GO TO 130
110 CONTINUE
GO TO 160
120 CONTINUE
C DOUBLE DIAGONAL ELEMENT
C AA = -AA*2.0
GO TO 190
130 CONTINUE
C FIND Q TRANSPOSED ELEMENT
C MTA = KL(JJ)
MTB = KL(JJ+1)-1
DO 140 K=MTA,MTB
C********* JA CALL
C
KK = IA(K)
IF ( KK.EQ.IR ) GO TO 150
140 CONTINUE
GO TO 160
150 CONTINUE
C SUBTRACT NEW ENTRY FROM OLD AND REPLACE OLD ONE
C
A(K) = A(K)-AA
KM(18) = KM(18)+1
KNR(IR) = KNR(IR)+1
GO TO 20
160 CONTINUE
C REVERSE SIGN OF DATA IN MINIMUM CONSTRAINT
C
AA = -AA
170 KNR(I) = KNR(I)+1
180 CONTINUE
C FILE AN ENTRY
C
C********* INSERT CALLED
C
IA(KM18) = II
A(KM18) = AA
GO TO 20
190 CONTINUE
C TERMINATE MATRIX DATA
C
KL(N+1) = KM18+1
ASSIGN 220 TO KWOT
210 CONTINUE
WRITE( NT6,200 ) M,N,KM(18)
KM(14) = 0
200 FORMAT( 17H0, PROBLEM HAS 14,6H ROWS, 14,14H COLUMNS, AND 1 15,16H MATRIX ENTRIES. )
KM(17) = M-MF+1
GO TO KWOT, ( 220,290 )
EXTEND KN LIST, SET TALLIES AND KL LIST. ONE DUMMY VARIABLE AT END OF X1 STORAGE. INCLUDE DUMMY VARIABLE IN COUNT.

220 CONTINUE
N = N+1
KM(1) = N
KM(9) = KM(1) + KM(2) + KM(3)
IIB = KM(1) + MP - 4
IF( IIB .LE. ISET ) GO TO 230
WRITE( NT6, 300 ) IIB
KM(14) = 0
300 FORMAT( 51HOPROBLEM TOO BIG. CONSTANTS PLUS VARIABLES EQUAL 1 I5 )
GO TO 290
230 CONTINUE

....... INITIALIZE CONSTANTS

IIA = N * KM(9)
MTA = 4
MTB = MTA + KM(2) + KM(3) - 1
KM(5) = 0
IF( KM(2) .NE. 0 ) KM(5) = N + 1
KM(6) = 0
IF( KM(3) .NE. 0 ) KM(6) = N + KM(2) + 1
KM(7) = KM(9) + 1

IF( KM(2) .EQ. -KM(3) ) GO TO 250

SET SLACKS FOR PRIMAL AND RECORD ROW NO.
SET COLUMN NAME AND COUNTS FOR TRANSPPOSED INEQUALITIES

DO 240 I = MTA, MTB
N = N + 1
IIA = IIA + 1
KN(N) = NR(I)
KNz(N) = NRZ(I)
KL(N) = I
KLT(N) = 1
KN(IIA) = NR(I)
KNz(IIA) = NRZ(I)
KL(IIA) = I
KLT(IIA) = KNR(I)
240 CONTINUE

250 CONTINUE

....... SET SLACKS FOR DUAL AND RECORD ROW NO.
RECORD COUNTS FOR PRIMAL VARIABLE

IIA = KM(1) - 1
DO 258 I = 1, IIA
N = N + 1
KN(N) = KN(I)
KNz(N) = KNZ(I)
DO 254 II = MM, M
IF( KN(I) .EQ. NR(II) .AND. KN2(I) .EQ. NRZ(II) ) GO TO 256
254 CONTINUE
GO TO 258
256 CONTINUE
   KL(N) = II
   KLT(N) = 1
   KLT(I) = KNR(I)
258 CONTINUE
      ADJUST N FOR TRANSPOSE OF INEQUALITY CONSTRAINTS AND DUMMY
      DUAL SLACK NAME

      N = N+KMQ(2)+KMQ(3)+1

      STORE TRANSPOSE OF A3 EQUALITY CONSTRAINTS

      KMQ(8) = N+1
      IF( KMQ(4).EQ.C ) GO TO 270
      MTA = KMQ(2)+KMQ(3)+4
      MTB = MTA+KMQ(4)-1
      DO 260 I=MTA,MTB
         N = N+1
         KN(N) = NR(I)
         KN2(N) = NR2(I)
         KL(N) = I
         KLT(N) = KNR(I)
260 CONTINUE
270 CONTINUE
      STORE PARAMETRIC COLUMN NAME

      CALL IMAGE( KN(N+1),KN2(N+1),I,8HL-L(0) )
      KL(N+1) = 3
      KLT(N+1) = KNR(3) + 1
      NQ = N
      N = N+1
      GO TO 310
280 CONTINUE
   WRITE( 6,400 ) J,J2
   KM(14) = 0
400 FORMAT( 2H0 , A4,A2,47H HAS BEEN USED FOR A COLUMN NAME AND AS ROW
          1ID. )
290 CALL ENDTIE
310 CONTINUE
   KBCD(5) = J
   KBCD2(5) = J2
   RETURN
320 K22 = K22+1
330 K22 = K22+1
340 CONTINUE
      CALL IMAGE( K1,K3,K22,24H ROW COLUMN ENTRIE )
350 CONTINUE
   WRITE( 6,500 ) K1,K3
   KM(14) = 0
500 FORMAT(27HOPROBLEM TOO BIG, TOO MANY ,A4,A2,2HS. )
      ASSIGN 290 TO KWOT
      GO TO 210
360 CONTINUE
   WRITE( 6,600 )
   KM(14) = 0
SUBROUTINE JMY( E, IP, LE, Y, IE )
IMPLICIT REAL*8 (A-H, O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19), M, MA, MB, MC, MD, ME, MF, N, KM(100), KP(12),
1 NTAP(4), KBCCD(40), MP, NC, KM(15), Z(20), T(10), PLAMDA, PLIMIT -
CALLED BY BOS, VER — COMPUTES ONE TRANSFORMED COLUMN

DIMENSION E(1), IP(1), LE(1), Y(1)
DIMENSION KM(50), KBCCD(2C)
EQUIVALENCE (KM(51), KM(21)), (KBCCD(21), KBCCD(21))
DIMENSION IE(1)
INTEGER*2 IP, LE, IE

IF ( ME * LE = 0 ) RETURN
DO 20 K2=1, ME
K3 = IP(K2)
D = Y(K3)
Y(K3) = 0.0

***** ABS CALL

IF ( DABS(D) * LE, Z(5) ) GC TO 20
MTA = LE(K2)
MTB = LE(K2+1) - 1
DO 10 K4=MTA, MTB

***** JE CALL

K5 = IE(K4)
10 Y(K5) = Y(K5) + E(K4) * D
20 GC CONTINUE
RETURN
END

SUBROUTINE KRT( E, IP, LE, P, IE )
IMPLICIT REAL*8 (A-H, O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19), M, MA, MB, MC, MD, ME, MF, N, KM(100), KP(12),
1 NTAP(4), KBCCD(40), MP, NC, KM(15), Z(20), T(10), PLAMDA, PLIMIT -
CALLED BY GET — APPLY REVERSE TRANSFORMATIONS TO P

DIMENSION E(1), IP(1), LE(1), P(1)
DIMENSION KM(50), KBCCD(2D)
INTEGER*2 IP, LE, IE
DIMENSION IE(1)
EQUIVALENCE (KM(51), KM(21)), (KBCCD(21), KBCCD(21))
DO 20 K=1, ME
   K1 = ME - K + 1
   MTA = LE(K1)
   MTB = LE(K1+1) - 1
   K2 = IP(K1)

   APPLY ONE TRANSFORMATION

   T(2) = 0.0
   DO 10 K3=MTA, MTB

   10 K4 = IE(K3)
   T(2) = T(2) + P(K4)*E(K3)
   P(K2) = T(2)
   20 CONTINUE

   RETURN
END

SUBROUTINE LOT(B,JH,KN,NR,P,X,Y,KN2,NR2)
IMPLICIT REAL*8(A-H,O-Z), INTEGER (I-N)
COMMON/XBLANK/NN(19),M,MA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
       NTAP(4),KBDC(40),MP,NQ,KMQ(15),Z(20),T(10),PAMCA,PLIMIT
CALLED BY QOT  OUTPUT FOR ROW DEPENDENT QUANTITIES

DIMENSION B(1),JH(1),KN(1),NR(1),P(1),X(1),Y(1)
DIMENSION KN2(1),NR2(1)
DIMENSION KM2(50),KBDC2(20)
EQUIVALENCE(KM(51),KM2(1)),(KBDC(21),KBDC2(1))
INTEGER*2 JH

NT6 = NTAP(3)
MM = MP

IF(KMQ(10).EQ.0) MM = M
K = MC + 1
WRITE(NT6,901)KM(12),KM2(12),KM(K+19),KM2(K+19)
DO 10 I=3,MM
   L = JH(I)
   K = 1
   IF(L.GE.KMQ(7)) K = 2
   WRITE(NT6,902)KN(L),KN2(L),KM(K+19),KM2(K+19),X(I),NR(I),
       1 NR2(I),B(I),P(I)
10 CONTINUE
RETURN

901 FORMAT(7HO NAME,15X,SHVALUE,13X,3HROW,10X,3HRHS,14X,2(A4,A2) /
       1 1X)
902 FORMAT(1X,2(A4,A2),F18.6,6X,A4,A2,3F18.6)
END
SUBROUTINE MIN( JM,A,KB,KL,P,Y,IA )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19),M,MA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
1 INTP(4),KB(40),MP,NQ,KM(15),Z(20),T(10),PLAMDA,PLIMIT
MIN CALLED BY BCS WHEN WORKING ON PRIMAL SUBPROBLEM
FIND COLUMNS WITH MINIMUM REDUCED COST
DIMENSION A(1),KL(1),P(1),KB(1),Y(1)
DIMENSION KM2(50),KB(2)(20)
EQUIVALENCE (KM(51),KM2(11)), (KB(21),KB(2)(1))
DIMENSION IA(1)
INTEGER*2 KB,KL,IA
NL = KM(7) - 1
IF( KM(47).EQ.KM(37) ) GO TO 20
CD 10 I=1,NL
IF( KB(I).LT.0 ) KB(I) = 0
10 CONTINUE
KM(47) = 0
20 JM = 0
Z(13) = 1.E+20
DO 30 J=1,NL
IF( KB(IJ).NE.0 ) GO TO 30
CALL DEL( J,0,A,KL,P,Y,IA )
****** ABS CALL
IF( KMG(10).LT.0 ) D = -CABS(C)
IF( D.GE.Z(13) ) GO TO 30
Z(13) = D
JM = J
30 CONTINUE
RETURN
END

SUBROUTINE PIV( IR,E,IE,IP,LE,X,Y )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)
COMMON / XBLANK / NN(19),M,MA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
1 INTP(4),KB(40),MP,NQ,KM(15),Z(20),T(10),PLAMDA,PLIMIT
PIV MODIFIED FOR QP
CALLED BY BCS,NEW,VER PILOT,E,- ADD A TRANSFORMATION (RO M2*)
DIMENSION KM2(50),KB(2)(20)
EQUIVALENCE (KM(51),KM2(11)), (KB(21),KB(2)(1))
DIMENSION E(1),IE(1),IP(1),LE(1),X(1),Y(1)
INTEGER*2 IE,IP,LE
KM(26) = KM(26)+1
T(1) = Y(IR)
T(3) = X(IR)/T(1)
C***** ABS CALL
C
        T(4) = DABS(Z(6)*T(1))
        X(IR) = 0.0
        Y(IR) = -1.0
        K = LE(ME+1)
        MM = MP
        IF( KMQ(10) .EQ. 0 ) MM = M
        DO 30 I=1,MM
          30          
            
C***** ABS CALL
C
        IF( DABS(Y(I)) .LE. T(4) ) GO TO 30
        X(I) = X(I)-T(3)*Y(I)
        IF( K .LE. KM(40) ) GO TO 10
        KM(24) = 1
        GO TO 20
        10          
          E(K) = -Y(I)/T(1)
C***** INSERT CALL
C
        IE(K) = I
        20          
          CONTINUE
          K = K+1
        30          
          CONTINUE
          Y(IR) = T(1)
          KM(19) = K-1
          ME = ME+1
          Z(9) = Z(9)*T(1)
        40          
          CONTINUE

C***** ABS CALL
C
        IF( DABS(Z(9)) .LT. 1.0 ) 50,70,60
        50          
          Z(9) = Z(9)*10.0
          KM(31) = KM(31)-1
          GO TO 40

C***** ABS CALL
C
        60          
          IF( DABS(Z(9)) .LT. 10.0 ) GO TO 70
          Z(9) = Z(9)/10.0
          KM(31) = KM(31)+1
          GO TO 60
        70          
          CONTINUE
          IF( ME .GT. KM(36) ) GO TO 80
          P(ME) = IR
          LE(ME+1) = K
          RETURN
        80          
          KM(24) = 1
          RETURN
          END

SUBROUTINE R(O( IS,JH,X,Y )
IMPLICIT REAL*8 (A-H,O-Z), INTEGER (I-N)

ROW 0010  ROW 0020
COMMON / XBLANK / NN(19), M, MA, MB, MC, MD, ME, MF, N, KM(100), KP(12),
NTAP(4), KBCD(40), MP, MQ, KMQ(15), Z(20), T(10), PLAMDA, PLIMIT

DIMENSION JH(1), X(1), Y(1)
DIMENSION KM2(50), KBCD2(20)
EQUIVALENCE (KM(51), KM2(1)), (KBCD(21), KBCD2(1))
INTEGER*2 JH

PIVOT SELECTION FOR QUADRATIC ALGORITHM — CALLED BOS
PHASE ONE SUB-PROBLEM, USE COMPOSITE — PREFER AN ARTIFICIAL
ROW IF X ZERO OR IN TIES, NO NEGATIVE ARTIFICIAL MAY BECOME
POSITIVE, DROP IT AT ZERO.
FOR QUADRATIC PROBLEM CALCULATE THETA FOR MOST NEGATIVE Y
TYPE VARIABLE, IF RATIO IS POSITIVE, STORE AS FIRST CHOICE.
MAINTAIN ALL X VARIABLES POSITIVE, SKIP Y AND PI TYPES.
FOR PARAMETRIC QUADRATIC, MAINTAIN ALL X AND Y VARIABLES
POSITIVE, SKIP PI.

KMQ(10) = 1
KMQ(10) = 0, AND KMQ(11) = 1
KMQ(10) = 0, AND KMQ(11) = 0

IR = 0
IA = 0
IB = 0
T(3) = 1.0E+10
T(4) = 0.0
WM = M
KM(45) = KMQ(7)
IF(KMQ(10).EQ.0) GO TO 10
WM = MP
GO TO 70
10 IF(KMQ(11).EQ.0) GO TO 20
KM(45) = N
IF(KMQ(8).NE.0) KM(45) = KMQ(8)
GO TO 70
20 CONTINUE
IF(KMQ(12).NE.0) GO TO 50

FIND YS INDEX IN JH LIST
DO 30 I=MF,MM
IF(JH(I).EQ.KMQ(14)) GO TO 40
30 CONTINUE
I = MM
40 KMQ(12) = I
50 IA = KMQ(12)
IF(-Y(IA).LT.Z(1)) GO TO 60
T(3) = X(IA)/Y(IA)
GO TO 70
60 IA = 0
70 CONTINUE
DO 130 I=MF,MM

******* ABS CALL
IF(DABS(Y(I)).LT.Z(1)) GO TO 130

ROW 0030
ROW 0040
ROW 0050
ROW 0060
ROW 0070
ROW 0080
ROW 0090
ROW 0100
ROW 0110
ROW 0120
ROW 0130
ROW 0140
ROW 0150
ROW 0160
ROW 0170
ROW 0180
ROW 0190
ROW 0200
ROW 0210
ROW 0220
ROW 0230
ROW 0240
ROW 0250
ROW 0260
ROW 0270
ROW 0280
ROW 0290
ROW 0300
ROW 0310
ROW 0320
ROW 0330
ROW 0340
ROW 0350
ROW 0360
ROW 0370
ROW 0380
ROW 0390
ROW 0400
ROW 0410
ROW 0420
ROW 0430
ROW 0440
ROW 0450
ROW 0460
ROW 0470
ROW 0480
ROW 0490
ROW 0500
ROW 0510
ROW 0520
ROW 0530
ROW 0540
ROW 0550
ROW 0560
ROW 0570
ROW 0580
ROW 0590
ROW 0600
ROW 0610
IF( JH(I).GE.KM(45) ) GO TO 130
IF( X(I) ) 90,80,110
80 CONTINUE

C
C
X(I) ZERO
C
IF( JH(I).EQ.0.OR.Y(I).GT.0.0 ) GO TO 150
GO TO 130
90 CONTINUE

C
C
X(I) NEGATIVE
C
IF( Y(I).GE.0.0 ) GO TO 130
IF( JH(I).EQ.0 ) GO TO 120
T(2) = X(I)/Y(I)
IF( T(2).LE.T(4) ) GO TO 130
IB = I
T(4) = T(2)
GO TO 130
110 CONTINUE

C
C
X(I) POSITIVE
C
IF( Y(I).EQ.0.0 ) GO TO 130
120 CONTINUE
T(2) = X(I)/Y(I)
IF( T(2).GT.T(3).OR.(T(2).EQ.T(3).AND.JH(I).NE.0) ) GO TO 130
IA = I
T(3) = T(2)
130 CONTINUE

C
C
CHOOSE MIN (T(3),T(4))
C
IF( IB.EQ.0 ) GO TO 140
IF( T(3).LT.T(4).OR.(T(3).EQ.T(4).AND.JH(IA).EQ.0) ) GO TO 140
IR = IB
Z(I) = T(4)
GO TO 180
140 CONTINUE
IR = IA
Z(I) = T(3)
GO TO 180
150 CONTINUE

C
C
MAXIMUM ABSOLUTE Y(I) FOR X(I)=0 IF ROW IS ARTIFICIAL
C
IT = 1
T(5) = 0.0
Z(I) = 0.0
IR = 0
GO 160 I=IT,MM

C
C
****** ABS CALL
C
IF( JH(I).NE.0.OR.X(I).NE.0.0.OR.DABS(Y(I)).LE.T(5) ) GO TO 160

C
C
****** ABS CALL
C
T(5) = DABS(Y(I))
IR = I

160 CONTINUE
IF( T(5) .GE. Z(1) ) GO TO 180
T(5) = 0.0
Z(8) = 0.0
IR = 0
CD = 170 = IT, MM
IF( X(I) .NE. 0.0 .OR. JH(I) .GE. KM(45) .OR. Y(I) .LE. T(5) ) GO TO 170
T(5) = Y(I)
IR = I
170 CONTINUE
IF( T(5) .LT. Z(1) ) IR = 0
180 CONTINUE
IS = IR
RETURN
END

SUBROUTINE XCH( K, JH, X )
IMPLICIT REAL*8 ( A-H, O-Z ), INTEGER ( I-N )
COMMON / XBLANK / NN(19), M, MA, MB, MC, MD, ME, MF, N, KM(100), KP(12),
1 NTAP(4), KBCD(40), MP, NG, KMQ(15), Z(20), T(10), PLAMDA, PLIMIT

CHECK SOLUTION FOR FEASIBILITY FOR QP CALLED BY BGS

DIMENSION JH(1), X(I)
DIMENSION KMZ(50), KBCDZ(20)
EQUIVALENCE ( KM(51), KMZ(1) ), ( KBCD(21), KBCDZ(1) )
INTEGER K, JH

RESET X AND CHECK FOR INFEASIBILITIES

KM(30) = 0
K = 0
IF( KMQ(10) .EQ. 0 ) GO TO 10
ASSIGN 60 TO KSW
MM = MP
GO TO 15
10 ASSIGN 40 TO KSW
MM = M
IF( KMQ(11) .LT. 0 ) N = NQ+1
KM(45) = KMQ(8)
IF( KMQ(8) .EQ. 0 ) KM(45) = N
IF( KMQ(11) .EQ. 0 ) KM(45) = KMQ(7)
15 CONTINUE
GO 80 I=MF, MM

***** **** ABS CALL

IF( DABS(X(I)) .GT. Z(2) ) GO TO 20
X(I) = 0.0
GO TO 80
20 CONTINUE
IF( X(I) ) 30, 80, 50
30 GO TO KSW, ( 60, 40 )
40 CONTINUE
```
CHECK FOR X OR Y TYPE VARIABLE IN QF

IF( JH(I),LT,KM(45) ) GO TO 60
IF( KMQ(11),EQ,0 ) GO TO 80

CHECK FOR PARAMETRIC VARIABLE

IF( JH(I),EQ,N ) GO TO 60
GO TO 80
50 CONTINUE
IF( JH(I),EQ,0 ) GO TO 70
GO TO 80
60 CONTINUE

DECLARE INFEASIBILITY

KM(30) = 1
70 K = 1
80 CONTINUE
RETURN
END

SUBROUTINE UOUT(K)
IMPLICIT REAL*(A-H,O-Z), INTEGER(I-N)
COMMON / XBLANK / NNN(19),M,MA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
1 NTAP(4),KBCD(40),MP,NQ,KMQ(15),Z(20),T(10),PLAMDA,PLIMIT
2 ,A(1)
DIMENSION L(1)
EQUVALENCE (A(1),L(1)), (NN(1),N1), (NN(2),N2), (NN(3),N3),
1 (NN(4),N4), (NN(5),N5), (NN(6),N6), (NN(7),N7),
2 (NN(8),N8), (NN(9),N9), (NN(10),N10), (NN(11),N11),
3 (NN(12),N12), (NN(13),N13), (NN(14),N14), (NN(15),N15),
4 (NN(16),N16), (NN(17),N17)
EQUVALENCE (NN(18),N18), (NN(19),N19)

OUTPUT CONTROL

CALL QOT(K,A(N1),A(N2),A(N7),A(N8),A(N9),A(N10),A(N12),A(N13),
1 A(N14),A(N15),A(N18),A(N19),A(N4))
RETURN
END

SUBROUTINE TAPENO
IMPLICIT REAL*(A-H,O-Z), INTEGER(I-N)
COMMON / XBLANK / NNN(19),M,MA,MB,MC,MD,ME,MF,N,KM(100),KP(12),
1 NTAP(4),KBCD(40),MP,NQ,KMQ(15),Z(20),T(10),PLAMDA,PLIMIT
NTAP(2) IS FORTRAN MONITOR INPUT TAPE
NTAP(3) IS FORTRAN MONITOR OUTPUT TAPE
NTAP(4) IS RESERVE INPUT TAPE
`````
NTAP(1) IS CURRENT INPUT TAPE (2 OR 4). SET IN INP OR MAIN.

NTAP(2) = 5
NTAP(3) = 6
NTAP(4) = 11
RETURN
END

SUBROUTINE SEARCH( KA, KC, L, M, KB )

SEARCH TRIES TO MATCH KA, KC WITH MATCHING BCD WORDS IN KB.

THERE ARE M WORDS (8 CHAR. LONG) IN KB. PUT THE ANSWER IN L.

DIMENSION KB(1)

DO 10 I=1,M
  IF( KA.EQ.KB(2*I-1).AND.KC.EQ.KB(2*I) ) GO TO 20
10 CONTINUE
  I = M+1
20 CONTINUE
  L = I
RETURN
END

SUBROUTINE NOP( I, J )

NOP SET UP THE ADDRESSES FOR IA AND IE ARRAYS. CALLS TO JA AND JE COULD THEN RETRIEVE THE RESPECTIVE INDECES. THESE CALLS ARE MARKED IN THE CODE.

SUBROUTINE INSERT WAS USED TO PACK THE INDEXES INTO IA AND IE.

THIS ROUTINE IS NOT USED FOR THE 360 VERSION.

DIMENSION A(2)

K = 1
RETURN
ENTRY ENDTIE

THIS MEANT TO RETURN TO THE LAST CALL TO TIE AT ONE TIME.

WRITE( 6,100 )
100 FORMAT(1X,' ENDTIE CALLED - EXIT TAKEN' )

CALL EXIT
ENTRY TIE
K = 1
RETURN
END

*GO*SYSDUMP DO SYSDUT=A
BEGIN
CP DIET PROBLEM
QPTTEST PROB 1
ROWS
$P
-004
-005
-006
-007
-008
-009
-010
-011
-012
END
QPTTEST
MATRIX
X0130 X0130 4.15
X0130 P 1.
X0130 004 0000.447
X0130 005 0001.411
X0130 006 0000.020
X0130 007 0003.650
X0130 009 0000.554
X0130 010 0000.333
X0130 011 0004.410
X0140 X0140 3.45
X0140 P 1.
X0140 004 0000.360
X0140 005 0000.897
X0140 006 0000.017
X0140 007 0000.990
X0140 008 0000.309
X0140 009 0000.174
X0140 010 0000.079
X0140 011 0001.060
X0150 X0150 .5
X0150 P 1.
X0150 004 0000.084
X0150 005 0000.422
X0150 006 0000.151
X0150 007 0000.090
X0150 008 0000.260
X0150 009 0000.030
X0150 010 0000.235
X0150 011 0000.110
X0150 012 0000.060
X0160 X0160 2.5
X0160 P 1.
X0160 004 0000.206
X0160 005 0000.017
X0160 006 0000.006
X0160 007 0000.060
X0160 008 0000.558
X0160 009 0000.002
X0170 X0170 .6
X0170 P 1.
X0170 004 0000.074
X0170 005 0000.448
QPT1000
QPT10001
QPT10002
QPT10003
QPT10004
QPT10005
QPT10006
QPT10007
QPT10008
QPT10009
QPT10010
QPT10011
QPT10012
QPT10013
QPT10014
QPT10015
QPT10016
QPT10017
QPT10018
QPT10019
QPT10020
QPT10021
QPT10022
QPT10023
QPT10024
QPT10025
QPT10026
QPT10027
QPT10028
QPT10029
QPT10030
QPT10031
QPT10032
QPT10033
QPT10034
QPT10035
QPT10036
QPT10037
QPT10038
QPT10039
QPT10040
QPT10041
QPT10042
QPT10043
QPT10044
QPT10045
QPT10046
QPT10047
QPT10048
QPT10049
QPT10050
QPT10051
QPT10052
QPT10053
QPT10054
QPT10055
QPT10056
| X0230 010 | 0000.058 |
| X0230 011 | 0000.370 |
| X0230 012 | 0000.862 |
| X0240 X0240 | 3.65 |
| X0240 P | 1. |
| X0240 004 | 0000.026 |
| X0240 005 | 0000.125 |
| X0240 006 | 0000.040 |
| X0240 007 | 0000.360 |
| X0240 008 | 0000.072 |
| X0240 009 | 0000.090 |
| X0240 010 | 0000.045 |
| X0240 011 | 0000.260 |
| X0240 012 | 0000.369 |
| X0250 X0250 | 6.7 |
| X0250 P | 1. |
| X0250 004 | 0000.058 |
| X0250 005 | 0000.166 |
| X0250 006 | 0000.038 |
| X0250 007 | 0000.590 |
| X0250 008 | 0000.166 |
| X0250 009 | 0000.047 |
| X0250 010 | 0000.059 |
| X0250 011 | 0000.210 |
| X0250 012 | 0001.184 |
| X0260 X0260 | 2.4 |
| X0260 P | 1. |
| X0260 004 | 0000.143 |
| X0260 005 | 0000.336 |
| X0260 006 | 0000.018 |
| X0260 007 | 0001.180 |
| X0260 008 | 0000.067 |
| X0260 009 | 0000.294 |
| X0260 010 | 0000.071 |
| X0260 011 | 0001.980 |
| X0260 012 | 0002.522 |
| X0270 X0270 | 2.05 |
| X0270 P | 1. |
| X0270 004 | 0000.011 |
| X0270 005 | 0000.106 |
| X0270 007 | 0001.380 |
| X0270 008 | 0009.184 |
| X0270 009 | 0000.057 |
| X0270 010 | 0000.138 |
| X0270 011 | 0000.330 |
| X0270 012 | 0002.755 |
| X0280 X0280 | 1.65 |
| X0280 P | 1. |
| X0280 004 | 0000.096 |
| X0280 005 | 0000.138 |
| X0280 006 | 0000.027 |
| X0280 007 | 0000.540 |
| X0280 008 | 0002.907 |
| X0280 009 | 0000.084 |
| X280 010 | 0.000.054 |
| X280 011 | 0000.830 |
| X280 012 | 0001.912 |
| X0290 X0290 | 1.05 |
| X0290 P | 1. |
| X0290 004 | 0000.085 | QPT10175 |
| X0290 005 | 0000.087 | QPT10176 |
| X0290 006 | 0000.017 | QPT10177 |
| X0290 007 | 001.730  | QPT10178 |
| X0290 008 | 0000.868 | QPT10179 |
| X0290 009 | 0000.012 | QPT10180 |
| X0290 010 | 0000.043 | QPT10181 |
| X0290 011 | 0000.550 | QPT10182 |
| X0290 012 | 0000.057 | QPT10183 |
| X0300 X0300 | 1.35   | QPT10184 |
| X0300 P     | 1.     | QPT10185 |
| X0300 004 | 0000.128 | QPT10186 |
| X0300 005 | 0000.099 | QPT10187 |
| X0300 006 | 0000.025 | QPT10188 |
| X0300 007 | 001.540  | QPT10189 |
| X0300 008 | 0000.357 | QPT10190 |
| X0300 009 | 0000.039 | QPT10191 |
| X0300 010 | 0000.043 | QPT10192 |
| X0300 011 | 0000.650 | QPT10193 |
| X0300 012 | 0000.257 | QPT10194 |
| X0310 X0310 | 1.6    | QPT10195 |
| X0310 P     | 1.     | QPT10196 |
| X0310 004 | 0000.174 | QPT10197 |
| X0310 005 | 001.055  | QPT10198 |
| X0310 006 | 0000.037 | QPT10199 |
| X0310 007 | 0004.590 | QPT10200 |
| X0310 008 | 0000.051 | QPT10201 |
| X0310 009 | 0000.269 | QPT10202 |
| X0310 010 | 0000.382 | QPT10203 |
| X0310 011 | 0000.930 | QPT10204 |
| X0320 X0320 | 2.65   | QPT10205 |
| X0320 P     | 1.     | QPT10206 |
| X0320 004 | 0000.269 | QPT10207 |
| X0320 005 | 001.691  | QPT10208 |
| X0320 006 | 0000.114 | QPT10209 |
| X0320 007 | 0007.920 | QPT10210 |
| X0320 009 | 0000.384 | QPT10211 |
| X0320 010 | 0000.246 | QPT10212 |
| X0320 011 | 0002.170 | QPT10213 |

**ROB 1**

| ND | 004 | 0000.030 |
|    | 005 | 0000.070 |
|    | 006 | 0000.008 |
|    | 007 | 0000.120 |
|    | 008 | 0000.050 |
|    | 009 | 0000.018 |
|    | 010 | 0000.027 |
|    | 011 | 0000.180 |
|    | 012 | 0000.075 |

**VTINV**

**RMODE**

242412934

**SOLVE**

**ARANMETRIC**

**EGIN**

**AMPLE PROBLEM FROM RSQP1**

**QPTTEST PRCB 2**

QPT2000
SOLVE
PARAMETRIC
BEGIN
THE FOLLOWING TEST IS A REPEAT OF PROBLEM 2. ASSUME THAT L(0) = 1.
ORIGINALY THE PROBLEM WAS SOLVED FOR THE SINGLE VALUE. NOW WE WISH
TO DO A PARAMETRIC RUN. WE CAN START WITH THE PUNCHOUTS FROM THE
PREVIOUS RUN, INVERT AND CONTINUE.

PARAMODE
2092222234
SAMPLE PROBLEM FROM RSQP1
QPTTEST PROB 2
<table>
<thead>
<tr>
<th>X1</th>
<th>LINEAR</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>MAXIM</td>
<td>5.0</td>
</tr>
<tr>
<td>X1</td>
<td>MIN</td>
<td>4.0</td>
</tr>
<tr>
<td>X1</td>
<td>EQUAL*</td>
<td>2.0</td>
</tr>
<tr>
<td>X1</td>
<td>X1</td>
<td>20.0</td>
</tr>
<tr>
<td>X2</td>
<td>LINEAR</td>
<td>-3.0</td>
</tr>
<tr>
<td>X2</td>
<td>MIN</td>
<td>2.0</td>
</tr>
<tr>
<td>X2</td>
<td>EQUAL*</td>
<td>1.0</td>
</tr>
<tr>
<td>X2</td>
<td>X2</td>
<td>5.0</td>
</tr>
<tr>
<td>X3</td>
<td>LINEAR</td>
<td>2.0</td>
</tr>
<tr>
<td>X3</td>
<td>MAXIM</td>
<td>-2.0</td>
</tr>
<tr>
<td>X3</td>
<td>MIN</td>
<td>1.0</td>
</tr>
<tr>
<td>X3</td>
<td>EQUAL*</td>
<td>1.0</td>
</tr>
<tr>
<td>X3</td>
<td>X2</td>
<td>1.0</td>
</tr>
<tr>
<td>X3</td>
<td>X3</td>
<td>5.5</td>
</tr>
</tbody>
</table>

**ETINV**

<table>
<thead>
<tr>
<th>X1</th>
<th>MIN</th>
<th>X3</th>
<th>MAXIM</th>
<th>X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQUAL*</td>
<td>DUAL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MINVERT**

**PARAMETRIC BEGIN**

**SECURITIES PORTFOLIO ANALYSIS**

**WSS**

$PROW +A1ROW A3R1 A3R2

**TSG 3**

**TRIX**

<table>
<thead>
<tr>
<th>X01</th>
<th>PROW</th>
<th>1.768</th>
</tr>
</thead>
<tbody>
<tr>
<td>X01</td>
<td>A1ROW</td>
<td>1.0</td>
</tr>
<tr>
<td>X01</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X01</td>
<td>A3R2</td>
<td>1.4174</td>
</tr>
<tr>
<td>X01</td>
<td>X01</td>
<td>1485690</td>
</tr>
<tr>
<td>X02</td>
<td>PROW</td>
<td>1.533</td>
</tr>
<tr>
<td>X02</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X02</td>
<td>A3R2</td>
<td>0.8398</td>
</tr>
<tr>
<td>X02</td>
<td>X02</td>
<td>441835</td>
</tr>
<tr>
<td>X03</td>
<td>PROW</td>
<td>1.641</td>
</tr>
<tr>
<td>X03</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X03</td>
<td>A3R2</td>
<td>0.9844</td>
</tr>
<tr>
<td>X03</td>
<td>X03</td>
<td>0.748415</td>
</tr>
<tr>
<td>X04</td>
<td>PROW</td>
<td>1.775</td>
</tr>
<tr>
<td>X04</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X04</td>
<td>A3R2</td>
<td>1.7250</td>
</tr>
<tr>
<td>X04</td>
<td>X04</td>
<td>1744960</td>
</tr>
<tr>
<td>X05</td>
<td>PROW</td>
<td>1.487</td>
</tr>
<tr>
<td>X05</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X05</td>
<td>A3R2</td>
<td>0.4940</td>
</tr>
<tr>
<td>X05</td>
<td>X05</td>
<td>322070</td>
</tr>
<tr>
<td>X06</td>
<td>PROW</td>
<td>1.630</td>
</tr>
<tr>
<td>X06</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X06</td>
<td>A3R2</td>
<td>0.9990</td>
</tr>
<tr>
<td>X06</td>
<td>X06</td>
<td>-0.0716985</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>------------</td>
</tr>
<tr>
<td>X07</td>
<td>PROW</td>
<td>1.652</td>
</tr>
<tr>
<td>X07</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X07</td>
<td>A3R2</td>
<td>0.7571</td>
</tr>
<tr>
<td>X07</td>
<td>X07</td>
<td>0.0513375</td>
</tr>
<tr>
<td>X08</td>
<td>PROW</td>
<td>1.547</td>
</tr>
<tr>
<td>X08</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X08</td>
<td>A3R2</td>
<td>0.8207</td>
</tr>
<tr>
<td>X08</td>
<td>X08</td>
<td>0.0601455</td>
</tr>
<tr>
<td>X09</td>
<td>PROW</td>
<td>1.792</td>
</tr>
<tr>
<td>X09</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X09</td>
<td>A3R2</td>
<td>1.5085</td>
</tr>
<tr>
<td>X09</td>
<td>X09</td>
<td>0.1008925</td>
</tr>
<tr>
<td>X10</td>
<td>PROW</td>
<td>2.009</td>
</tr>
<tr>
<td>X10</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X10</td>
<td>A3R2</td>
<td>2.0150</td>
</tr>
<tr>
<td>X10</td>
<td>X10</td>
<td>0.2647320</td>
</tr>
<tr>
<td>X11</td>
<td>PROW</td>
<td>1.461</td>
</tr>
<tr>
<td>X11</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X11</td>
<td>A3R2</td>
<td>1.1934</td>
</tr>
<tr>
<td>X11</td>
<td>X11</td>
<td>0.0692795</td>
</tr>
<tr>
<td>X12</td>
<td>PROW</td>
<td>1.725</td>
</tr>
<tr>
<td>X12</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X12</td>
<td>A3R2</td>
<td>1.1843</td>
</tr>
<tr>
<td>X12</td>
<td>X12</td>
<td>0.0921625</td>
</tr>
<tr>
<td>X13</td>
<td>PROW</td>
<td>1.669</td>
</tr>
<tr>
<td>X13</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X13</td>
<td>A3R2</td>
<td>0.5831</td>
</tr>
<tr>
<td>X13</td>
<td>X13</td>
<td>0.0682475</td>
</tr>
<tr>
<td>X14</td>
<td>PROW</td>
<td>1.438</td>
</tr>
<tr>
<td>X14</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X14</td>
<td>A3R2</td>
<td>0.5904</td>
</tr>
<tr>
<td>X14</td>
<td>X14</td>
<td>0.0276570</td>
</tr>
<tr>
<td>X15</td>
<td>PROW</td>
<td>1.676</td>
</tr>
<tr>
<td>X15</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X15</td>
<td>A3R2</td>
<td>1.4622</td>
</tr>
<tr>
<td>X15</td>
<td>X15</td>
<td>0.0832675</td>
</tr>
<tr>
<td>X16</td>
<td>PROW</td>
<td>1.442</td>
</tr>
<tr>
<td>X16</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X16</td>
<td>A3R2</td>
<td>0.5562</td>
</tr>
<tr>
<td>X16</td>
<td>X16</td>
<td>0.0291935</td>
</tr>
<tr>
<td>X17</td>
<td>PROW</td>
<td>1.423</td>
</tr>
<tr>
<td>X17</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X17</td>
<td>A3R2</td>
<td>0.4924</td>
</tr>
<tr>
<td>X17</td>
<td>X17</td>
<td>0.0191135</td>
</tr>
<tr>
<td>X18</td>
<td>PROW</td>
<td>1.517</td>
</tr>
<tr>
<td>X18</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X18</td>
<td>A3R2</td>
<td>0.3989</td>
</tr>
<tr>
<td>X18</td>
<td>X18</td>
<td>0.0335600</td>
</tr>
<tr>
<td>X19</td>
<td>PROW</td>
<td>1.497</td>
</tr>
<tr>
<td>X19</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X19</td>
<td>A3R2</td>
<td>0.3315</td>
</tr>
<tr>
<td>X19</td>
<td>X19</td>
<td>0.0289235</td>
</tr>
<tr>
<td>X20</td>
<td>PROW</td>
<td>1.820</td>
</tr>
<tr>
<td>X20</td>
<td>A3R1</td>
<td>1.0</td>
</tr>
<tr>
<td>X20</td>
<td>A3R2</td>
<td>1.7867</td>
</tr>
<tr>
<td>X20</td>
<td>X20</td>
<td>0.1804645</td>
</tr>
<tr>
<td>X21</td>
<td>PROW</td>
<td>0.503</td>
</tr>
<tr>
<td>X21</td>
<td>A3R2</td>
<td>1.0</td>
</tr>
</tbody>
</table>
X21 X21 .0444020 -111-
END
QPOBTEST
RHS
A1ROW 1.0
A3R1 1.0
A3R2 0.0
END
PRMODE
007212224
L(0)
0.15
LIMIT L(0)= .15 LIMIT IS .75 .75
FREQUENCIES
OUTPUT 5
INVERT 10
END
SOLVE
PARAMETRIC
EXIT
/*
//
// EXEC ASMCL PARM ASM='DECK'
//ASM SYSIN DD *
IGET TITLE 'FUNCTION IGETAD(WORD)'
MACRO
&NAME BEGINZ &A,&B
**************************************************************************
**SUBROUTINE ENTRY AND EXIT MACROS. (BEGINZ/EXITZ)
**************************************************************************
**PURPOSE
BEGINZ SETS UP THE ENTRY SEQUENCE FOR A SUBROUTINE. IT
**CORRESPONDS TO THE FORTRAN LINKAGE CONVENTIONS. EXITZ
**CAUSES A RETURN FROM THE SUBROUTINE.
**
**FORMS
NAME BEGINZ NSA,FN
**NAME IS THE NAME OF THE SUBROUTINE.
**BEGINZ MUST BE CODED AS SHOWN.
**NSA IS OPTIONAL. IF CODED, NO REGISTER SAVE AREA IS
**PROVIDED. THIS SAVES 64 BYTES BUT SHOULD BE USED
**ONLY IF THE SUBROUTINE DOES NOT CALL ANOTHER
**SUBROUTINE.
**FN IS OPTIONAL. IF CODED, IT INDICATES THAT THE
**ROUTINE IS A FUNCTION AND REGISTER ZERO IS NOT
**RESTORED.
**SYMBOL EXITZ NAME
**THIS CAUSES A RETURN FROM THE SUBROUTINE. THE MACRO
**MAY APPEAR ANYWHERE WITHIN THE SUBROUTINE. THERE MAY
**BE MULTIPLE EXITS. THE NAME MUST CORRESPOND TO THE
**NAME USED IN BEGINZ.
*IN ALL GENERAL REGISTERS ARE SAVED.
*GR2 IS SET UP AS THE BASE FOR THE SUBROUTINE.
*GR13 IS LOADED WITH THE ADDRESS OF A SAVE AREA (UNLESS NSA IS SPECIFIED.)
*OUT ALL REGISTERS ARE RESTORED.

**********ENTRY SEQUENCE TO SUBROUTINE**********

&NAME CSECT
BC 15+12(0+15)
DC X'7'
DC CL7'&NAME'
STM 14+12,12(13)
BALR 2,0
USING *+2
ST 13+SAVRESYSNDX+4
B GOONSYSNDX
L 13+SAVRESYSNDX+4
AIF ('&B' NE 'FN') A
LM 14+15,12(13)
ULM 1+12,24(13)
AGO *B
.A
ANOP
LM 14+12,12(13)
.B
ANOP
MVI 12(13),X'FF'
BCR 15+14
SAVRESYSNDX DS 2F
AIF ('&A' EQ 'NSA') C
DS 16F
GOONSYSNDX LR 15+13
LA 13+SAVRESYSNDX
ST 13+8(0+15)
ST 15+4(0+13)
AGO *D
.C
ANOP
GOONSYSNDX DS 0H
.D
ANOP

**********END OF ENTRY SEQUENCE**********

MEND
MACRO

**********RETURN TO CALLER**********

&NAME EXITZ &B

**********RETURN TO CALLER**********

&NAME B &B+26
MEND

IGETAD START 0
IGETAD BEGINZ NRS.FN
L 0+0(1)
EXITZ IGETAD
END
REFERENCES
