AD-A096 140 STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB F/G 5/1

A MODIFIED AGGREGATION PROGRAM FOR THE PILOT PROCESS INTEGRATED--ETC(U)

DEC 80 H HIROSE DE-AC03-76SF00326

UNCLASSIFIED SOL-80-31 NL

1 OF 1

AD-A
096140

END

DATE

FILMED

4-81

DTIC

LEVEL II

Systems
Optimization
Laboratory

AD A096140

Department of Operations Research
Stanford University
Stanford, CA 94305

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

A MODIFIED AGGREGATION PROGRAM FOR THE PILOT PROCESS INTEGRATED MODEL

by

Haruko Hirose

TECHNICAL REPORT SOL-80-31

December 1980

Research and reproduction of this report were supported by the Department of Energy Contract DE-AC03-76SF00326, PA# DE-AT03-79EI10601, DE-AM03-76SF00326, PA# DE-AT03-80EI10682; Electric Power Research Institute Contract RP 652:1; Institute for Energy Studies at Stanford University.

The views expressed in this document are those of the authors and NOT necessarily those of any of the sponsors.

Abstract

The PILOT Process Integrated Model can produce energy/economic scenarios for time periods of up to 100 years by aggregating several 5 year time periods into one. This report presents modification to an existing aggregation method that utilize the special structure of the Consumers Energy Service Model (CESM) and the Industrial Energy Service Model (IESM) and reduce aggregation bias in these portions of the PPIM.

TABLE OF CONTENTS

TABLES

ILLUSTRATIONS

I. INTRODUCTION

The Stanford PILOT Energy/Economic model provides projections of energy production and use and of economic growth in the U.S. over a 40 year span, 1973-2012, divided into eight 5 year periods. A longer time horizon of say, 100 years, would enable the PILOT model to address policy decisions whose effects may not be felt till well past the turn of the century. The decisions surrounding plutonium recycling and the fast breeder reactor fall in such a category, and have been studied using a longer time horizon by Avi-Itzhak and Connolly [1]. A longer time horizon for the PILOT model is also useful for determining terminal capital stocks and other end conditions for the shorter 40 year time horizon.

However, it is not practical computationally to run a scenario of 20 periods of 5 years each. To overcome this difficulty, a computer program has been developed and tested to aggregate the 20 time periods into a smaller number of planning periods of variable length, yielding a LP matrix the size of the 40-year PILOT model [2]. The length of any time period in the aggregated matrix is some multiple of 5 years.

The aggregation scheme consists of two steps.

- Aggregating variables (by adding column coefficients).
- Aggregating equations (by adding row coefficients).

It has been shown that the solution yielded by the reduced problem is consistent with that of the original problem but not necessarily conversely [3].

The aggregation scheme substitutes one planning period for several periods of the original matrix. The activity levels in the aggregated periods are intended to be representative of similar activities in the several unaggregated periods.

Since the date this scheme was first implemented, some modifications to the PILOT model have been made. A Consumers Energy Service Model (CESM) [4] and an Industrial Energy Service Model (IESM) [5] have been added to the PILOT model, forming the PILOT Process Integrated Model. These two submodels utilize energy facility capital stock accounting different from that in the main model. This capital stock modeling leads to LP columns with exponentially declining coefficients in later periods, and suggests that a somewhat different aggregation scheme may help decrease aggregation bias in the CESM and IESM portion of the integrated model. The CESM and IESM together contain approximately 320 rows and 1000 columns of the total 1300 rows and 2700 columns in an eight-period PILOT matrix. An aggregation scheme that can reduce bias in this fraction of the total model should yield improved results for the whole as well.

II. MODIFICATION OF THE VARIABLE TIME MODEL

A. CESM and IESM models

Many CESM and IESM variables refer to the total amount of capacities installed in the current period. Fractions of these capacities survive to be used in latter periods. The capacities depreciate according to an exponential curve, for example, if the coefficient of a column in period t is 1, the coefficient in period $t + k$ is d^k where $0 < d < 1$ is the survival fraction from one 5-year period to the next.

Consider the example of an energy technology T installed in period 1. Suppose periods 2, 3 and 4 are aggregated to a single 15 year period. Since midpoints of the planning periods are used as representative dates, the contribution of technology T in the aggregated period should be given by a survival fraction based on 2 full time periods, or d^2. A scheme of choosing the coefficient according to an arithmetic average would give a coefficient equal to $\frac{d + d^2 + d^3}{3}$. If the geometric mean is used, the new coefficient is $\sqrt[3]{d \cdot d^2 \cdot d^3} = d^2$.

Any aggregation scheme introduces a bias. But a scheme that more accurately approximates the "true" coefficient is desirable. Therefore we will use an aggregation scheme that computes the geometric mean of original coefficients for those columns in the CESM and IESM portion that display the exponentially declining coefficients.

B. Geometric aggregation scheme

An outline of the geometric scheme follows.

1. Take the geometric mean of the column's coefficients across all rows of the periods to be aggregated.

2. Add coefficients across columns of the periods to be aggregated.

Columns in the CESM and IESM other than these capacity columns are aggregated in the standard aggregation scheme.

The following is an example of aggregation of 3 periods with a representative survival rate of 0.6 and an increasing service need.

$$
\begin{array}{ccc}
\text{new period} \longrightarrow & & \overbrace{\begin{array}{ccc} \quad & \quad & \quad \end{array}}^{1} \\
\downarrow & \text{old period} \rightarrow & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\
1 & \left\{ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \right. & \begin{bmatrix} 1 & 0 & 0 \\ 0.6 & 1 & 0 \\ 0.36 & 0.6 & 1 \end{bmatrix}
\end{array}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} y \\ 2y \\ 3y \end{bmatrix}
$$

x_i = values are total capacity installed.
y_i = values are service needed.
assumed $y_1 = y$, $y_2 = 2y$, $y_3 = 3y$.

Without aggregation, the solutions are

$$x_1 = y, \quad x_2 = 1.4y, \quad x_3 = 1.8y$$

A representative value, the mean of x_1, x_2, x_3 is $\overline{x} = 1.4y$ for the new period 1.

Using the geometric aggregation scheme, the single resulting equation and its solution are:

$$4.35x = 6y \ , \quad \bar{x}_g = 1.38$$

Using the arithmetic aggregation scheme, the single resulting equation and its solution are

$$4.56x = 6y \ , \quad \bar{x}_a = 1.32$$

This result illustrates that geometric aggregation gives results closer to the original 5 year period model.

C. Modification of variable time model program

The arithemtic aggregation scheme is implemented in a FORTRAN program that processes the MPS - format LP matrix listing. As a programming convenience in the first implementation, only additions are made and the coefficients in arithemtic aggregation are not divided by the number of periods in the aggregation. Thus two identical rows would appear aggregated as one row, but with all coefficients multiplied by two. The geometric aggregation scheme must therefore multiply the geometric mean by the number of periods to maintain correct linkage and consistency with the rest of the model.

The aggregation takes place in two stages, first across rows then across columns. Modification to the existing program is done in two parts. The first is shown in Figure 1 where row aggregation in the main program is done. The second is shown in Figure 2 where column aggregation is done in the subroutine UPDATE. To distinguish the two aggregation modes, marker cards reading "*ARITH" and "*GEOM" are needed in the input deck. If no marker appear, the program defaults to arithmetic aggregation for the entire matrix.

Figure 1

Row aggregation in the MAIN program

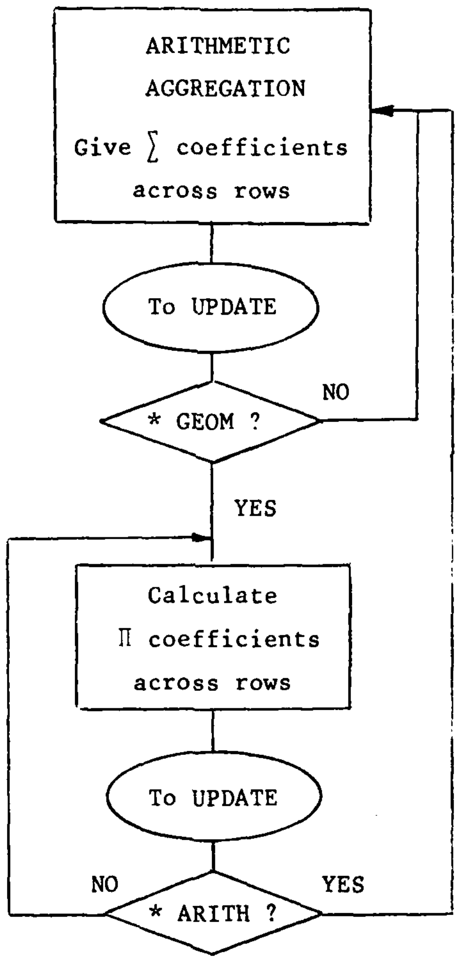
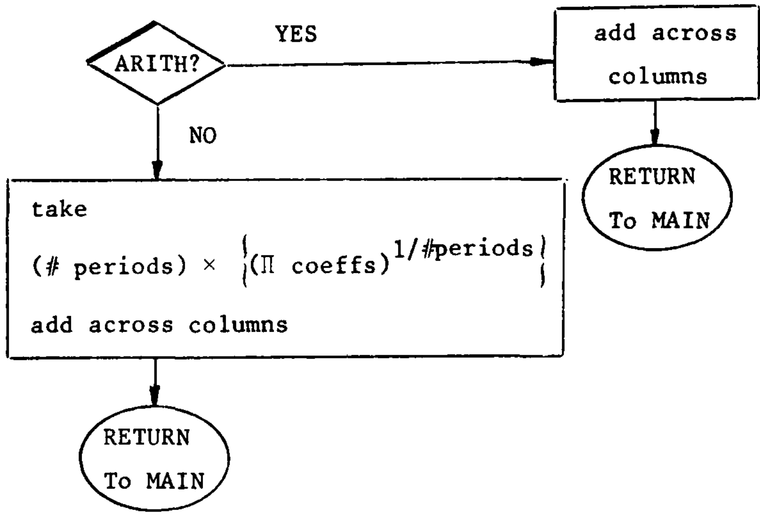


Figure 2

Column aggregation in the subroutine UPDATE



III. Test Run of Modified Variable Time Model

A. Test run

Test run of the geometric aggregation scheme was made and compared with arithmetic aggregation results. To minimize computational costs the development and testing was done on a eight period 40 year model which contained the CESM but not the IESM. The eight periods were aggregated to five planning periods covering the same horizon. The qualitative results presented here will generalize both to a longer time horizons and to a model containing the IESM.

The test runs were made with an unaggregated 8 period "Longdeck" and two aggregated 5 period models: "Short G", derived using the modified geometric aggregation scheme and "Short A", derived using only the arithmetic aggregation. A single aggregation mapping of periods from Longdeck to either aggregated matrix was tested. The mapping 1-2-3-1-1 yields 5 periods in the short deck with lengths 5, 10, 15, 5 and 5. For example periods 2 and 3 from longdeck become period 2 in either short A or short G. No aggregation is done for either the first or last periods in order that the aggregated matrices have several periods with identical coefficients as the original.

B. Comparison of results

The observed objective function values were

Longdeck	Short G	Short A
5745.16663	5701.71558	5654.43644

Note that the objective value of Short G is closer than that of Short A to the value of the unaggregated Longdeck.

The comparison of objective values alone is not sufficient to indicate that Short G yields better results. We will also present comparison of more detailed model activities. Gross National Product and Total Primary Energy Consumption are activities from the main portion of the model. Their aggregation is done using an arithmetic scheme in both the original and modified variable time programs. The results given in Table 1 show that differences are small between aggregation schemes for these two variables. However, we note that the numerical vaules from the modified program are larger than the those from the original program.

The modified aggregation scheme focused on the CESM portion of the integrated PILOT model. The CESM models uses four energy services; space heat, other thermal residential, air conditioning, and automobile drive. A total of 55 energy service technologies provide these four services. Due to the structure of the CESM we cannot expect the LP solution of an aggregated model to agree with the unaggregated solution in all 55 technologies for every period. However, the totals across energy service types demonstrate that a geometric aggregation scheme for CESM capital stock variables yields solution closer to the unaggregated values. Tables 2 and 3 present the solution values for all technologies in two of the four CESM energy services. The survival rates are 0.918 for Space Heat and 0.59 for Other Thermal.

Table 1

GNP and Total Primary Energy Consumption Comparison

<table>
<tr><td colspan="2" rowspan="2">old period / new period / used scheme / activity</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>1</td><td colspan="2">2</td><td colspan="3">3</td><td>4</td><td>5</td></tr>
<tr><td rowspan="4">GNP</td><td rowspan="2">Longdeck</td><td rowspan="2">1906.110</td><td>2346.653</td><td>2744.179</td><td>3184.402</td><td>3635.427</td><td>4118.824</td><td rowspan="2">4623.734</td><td rowspan="2">5201.469</td></tr>
<tr><td colspan="2">2545.416</td><td colspan="3">3646.218</td></tr>
<tr><td>Short G</td><td>1906.109</td><td colspan="2">2561.071</td><td colspan="3">3699.107</td><td>4388.879</td><td>4964.898</td></tr>
<tr><td>Short A</td><td>1906.109</td><td colspan="2">2513.325</td><td colspan="3">3693.579</td><td>4367.926</td><td>4936.766</td></tr>
<tr><td rowspan="4">Total Primary Energy Consumption</td><td rowspan="2">Longdeck</td><td rowspan="2">75.201</td><td>82.395</td><td>92.871</td><td>105.898</td><td>117.441</td><td>128.746</td><td rowspan="2">146.118</td><td rowspan="2">162.654</td></tr>
<tr><td colspan="2">87.633</td><td colspan="3">117.362</td></tr>
<tr><td>Short G</td><td>75.504</td><td colspan="2">90.279</td><td colspan="3">118.303</td><td>140.272</td><td>157.158</td></tr>
<tr><td>Short A</td><td>75.504</td><td colspan="2">87.974</td><td colspan="3">117.987</td><td>140.262</td><td>157.041</td></tr>
</table>

Table 2

Space Heat Comparison

Energy Service	used scheme	old period 1	2	3	4	5	6	7	8
	new period	1	2		3			4	5
ER0	Longdeck	.302	.286	.268	.249	.228	.205	.179	.151
	Short G	.302	.277		.227			.179	.151
	Short A	.302	.277		.227			.179	.151
ER2	L			.032	.060	.055	.051	.046	.301
	S G								
	S A		.081		.065			.055	.051
HP2	L	.019	.119	.362	.738	1.327	1.851	2.659	3.441
	S G	.019	.194		.784			1.621	2.488
	S A	.019	.194		.784			1.621	2.488
SE1	L	.012	.011	.010	.009	.008	.008	.007	.006
	S G	.012	.120		.096			.081	.075
	S A	.012	.011		.009			.007	.007
SE2	L	.019	.046	.295	.676	.666	.612	.755	.693
	S G	.019	.194		.310			.504	.553
	S A	.019	.194		.156			.564	.830
FG0	L	1.493	1.412	1.325	1.229	1.124	1.010	.884	.748
	S G	1.493	1.369		1.121			.884	.748
	S A	1.493	1.369		1.121			.884	.748
SG2	L	.019	.017	.194	.319	.293	.269	.247	.227
	S G	.019	.114		.092			.078	.071
	S A	.019	.113		.091			.077	.070
FO0	L	1.171	1.108	1.039	.964	.882	.792	.694	.586
	S G	1.171	1.074		.879			.694	.586
	S A	1.171	1.074		.879			.694	.586
FO2	L	.411	.377	.346	.318	.292	.268	.246	.226
	S G	.411	.376		.304			.256	.235
	S A	.411	.361		.292			.246	.226
Total	Longdeck	3.446	3.6235		4.834			5.717	6.379
	Short G	3.446	3.718		3.813			4.297	4.907
	Short A	3.446	3.674		3.624			4.327	5.157

Table 3

Other Thermal Comparison

Energy Service	used scheme \ old period	1	2	3	4	5	6	7	8
	new period	1	2		3			4	5
EW0	Longdeck	.350	.132						
	Short G	.350	.081						
	Short A	.350	.081						
EW1	L								
	S G				.609			.212	.125
	S A				.511			.178	.105
EW2	L								.706
	S G								
	S A								
EW3	L			.460	.867	1.074	1.188	1.363	.804
	S G							.653	1.075
	S A							.715	1.142
GW0	L	.741	.279						
	S G	.741	.172						
	S A	.741	.172						
GW3	L	.168	.786	.625	.369	.218	.128	.076	.045
	S G	.168	.833		.222			.077	.046
	S A	.168	.802		.214			.075	.044
SW1	L			.150	.088	.052	.031	.018	.011
	S G		.035		.271			.094	.056
	S A		.030		.235			.082	.048
SW2	L	.013	.063	.187	.321	.406	.469	.567	.668
	S G	.013	.131		.035			.292	.500
	S A	.013	.131		.035			.310	.523
Total	L	1.271	1.342		1.736			2.624	1.528
	S G	1.271	1.252		1.137			1.328	1.800
	S A	1.271	1.216		0.995			1.360	1.862

Note that solution values from the geometric aggregation are greater than the those from the arithmetic aggregation. This is a general result and can be stated as the following proposition.

Proposition

$\bar{x}_g \geq \bar{x}_a$ where x_i has exponentially declining coefficients.

Proof) For $0 < d < 1$, the geometric mean of the powers of d $\leq$ arithmetic mean, i.e.

$$\left(\prod_{i=1}^{n} d^{i-1}\right)^{1/n} \leq \frac{1}{n}\sum_{i=1}^{n} d^{i-1}$$

Therefore, the new coefficients of geometrically aggregated periods are less than or equal to the corresponding coefficients in arithmetically aggregated periods.

The energy service demanded in PILOT is influenced indirectly by the total investments in energy facility capital stocks, but this influence is quite small and not large enough to overcome the difference in coefficient values between the aggregation schemes. Therefore the integrated solutions of the aggregated models will exhibit similar values for GNP and other macroeconomic values and larger values for total CESM capital stocks in the modified aggregation.

C. Conclusion

Any aggregation scheme introduces some aggregation bias, which indicates information is lost. For the PILOT CESM and IESM, this aggregation bias can be reduced by using a scheme based on a geometric mean of coefficients. The numerical results and the proposition above show that CESM and IESM values from a geometric aggregation are larger in absolute values than those from an arithmetic scheme.

The capital stock structure of the CESM embodies information of two types. Stocks are installed that provide energy service demand within a single time period and that replace earlier vintages of capital stock that have depreciated. This inter-temporal depreciation relation is destroyed by aggregation. By using a geometric aggregation scheme, solution values more closely approximate the representative values from the unaggregated periods, thus recapturing some lost information and reducing aggregation bias. Even though numerical results are presented for a short time horizon in a model containing only the CESM, the qualitative results are expected to hold for longer time horizons and for an integrated model containing the IESM as well.

REFERENCES

[1] Avi-Itzhak, B., and T.J. Connolly, "The Plutonium Issue, An Analysis of Policies Deferring the Introduction of Plutonium-Fueled Reactors in the U.S.", Technical Report SOL 78-24, Department of Operations Research, Stanford University, Stanford, California, September 1978.

[2] Buras, N., and G.B. Dantzig, "Analysis Over Longer Planning Horizon in the PILOT Energy/Economic Model", Energy Project Memorandum 77-19, Deaprtment of Operations Research, Stanford University, Stanford, California, October 1977.

[3] Dantzig, G.B., T.J. Connolly, and S.C. Parikh, "Stanford PILOT Energy/Economic Model (appendix H)", Report prepared for Electric Power Research Institute, EPRI EA-626, Project 652-1, Interim Report, Volume 2, May 1978.

[4] Avi-Itzhak, B., and A. Iusem, "A Consumers Energy Services Model", Technical Report SOL 79-16, Department of Operations Research, Stanford University, Stanford, California, September 1979.

[5] Avi-Itzhak, B., and A. Iusem, "The Industrial Energy Service Model", to appear as an appendix B in "PILOT 1980 Energy-Economy Model", Department of Operations Research, Stanford University, Stanford, California, forthcoming.

```
// JOB ,CLASS=E,REGION=256K,TIME=(10,00)
//*
//*   VARIABLE TIME PERIOD PROGRAM
//*   -------- ---- ------ -------
//*
//*MAIN HOLD=OUTPUT
//DELCONDS EXEC PGM=IEFBR14
//DD1       DD DSN=WYL.WJ.***.SHORT,VOL=SER=WORK03,UNIT=DISK,
//             DISP=(OLD,DELETE)
//*
//*   THE PRECEEDING STEP (DELCONDS) SHOULD DELETE THE OUTPUT FILE
//*   FROM ANY PREVIOUS RUN OF THIS PROGRAM
//*
//*   VARIABLE TIME PERIOD PROGRAM
//*   -------- ---- ------ -------
//*
//*   INPUT  - TWO CARDS LOCATED AT END OF THIS DECK
//*             (2ND CARD CONTAINS AGGREGATION SCHEME)
//*            - FT08F001 INPUT MODEL FILE IN MPS FORMAT
//*             (SEE COMMENTS WITHIN PROGRAM FOR ASSUMPTIONS)
//*
//*   OUTPUT - FT06F001 LIST OF WARNING MESSAGES
//*            - FT09F001 OUTPUT MODEL FILE IN MPS FORMAT
//*
//*   DOCUMENTATION - COMMENTS WITHIN THIS PROGRAM
//*
//*     MPS III MATH. PROG. SYSTEM USER MANUAL, SECTION 6,
//*           INPUT DATA FORMATS
//*
//*     ENERGY PROJECT MEMO # 76-34, "THE PROCEDURE OF USING THE
//*            VARIABLE TIME MODEL", KUE-LIN WU, DEC. 1976.
//*
//*     ENERGY PROJECT MEMO # 77-1,"AGGREGATION OF CONSTRAINTS AND
//*            VARIABLES IN LINEAR PROGRAMS",RICHARD WOLLMER,JAN. 1977.
//*
//*     ENERGY PROJECT MEMO # 77-19,"ANALYSIS OVER LONGER PLANNING
//*            HORIZON IN THE PILOT ENERGY/ECONOMIC MODEL",NATHAN
//*            BURAS AND GEORGE B. DANTZIG, OCT. 1977.
//*
//*     SOL WORKING PAPER #76-3,"VARIABLE-TIME PERIODS AND END-
//*            CONDITION EFFECTS OF THE PILOT ENERGY MODEL",
//*            KUE-LIN WU,RICHARD WOLLMER, AND NATHAN BURAS, DEC. 1976.
//*
// EXEC WATFIV,FORTVER=NEW
//FT06F001 DD SYSOUT=A
//FT08F001 DD UNIT=DISK,DSN=WYL.WJ.***.LONGDECK,VOL=SER=WORK03,
//             DISP=SHR
//FT09F001 DD UNIT=DISK,DSN=WYL.WJ.***.SHORT,VOL=SER=WORK03,
//             SPACE=(TRK,(200,20),RLSE),DISP=(NEW,KEEP),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//GO.SYSIN DD *
$WATFIV
```

```
      CHARACTER*1 PRNAME(100,2),TYPIN(4),TYPE(4),NAMEIN(8),NEWNAME(8),
     1            ROWNAME(8),COLNAME(8),CNAME(8),RNAME(2,8)
      CHARACTER*1 DIGIT(10),BLANK,ASTERSK,
     1            COL(4),RHS(4),BOUNDS(4),ENDATA(4),GEM(4),ARI(4),
     2            FR(2),FX(2),UP(2),LO(2),MI(2)
      CHARACTER*80 CARD
      INTEGER PBLANK,LISTIN(20),INOUT(20)
      LOGICAL PASS1,ARITH,GEOM,MINUS1,TRUE,FALSE
C---
C---  (THE DIMENSION OF NAMETAB AND VALUTAB SHOULD EXCEED THE MAXIMUM
C---   NUMBER OF MPS ROW ENTRIES ANTICIPATED IN ANY COLUMN, AFTER
C---   AGGREGATION. ANY REDIMENSIONING MUST BE CARRIED OUT IN SUBROUTINES
C---   UPDATE AND COLOUT AS WELL)
C---
      CHARACTER*1 NAMETAB(100,8)
      DIMENSION   VALUTAB(100),RVALU(2)
      COMMON/BLOCK1/NPRIN,NPROUT,INOUT,PRNAME
      COMMON/BLOCK2/COLNAME,NAMETAB,VALUTAB,MAXENT
C---
C---
C---
C---
C INITIALIZE CONSTANTS AND READ AGGREGATION SCHEME
C---
C---  INITIALIZE ARRAY DIGIT TO CHARACTER EQUIVALENTS OF THE 10 DIGITS
C---
      DATA DIGIT(1),DIGIT(2),DIGIT(3),DIGIT(4)/'0','1','2','3'/
      DATA DIGIT(5),DIGIT(6),DIGIT(7),DIGIT(8)/'4','5','6','7'/
      DATA DIGIT(9),DIGIT(10)/'8','9'/
C---
C---  INITIALIZE COL,RHS,BOUNDS, AND ENDATA (MPS SEGMENT NAMES)
C---
      DATA COL(1),COL(2),COL(3),COL(4)/'C','O','L','U'/
      DATA RHS(1),RHS(2),RHS(3),RHS(4)/'R','H','S',' '/
      DATA BOUNDS(1),BOUNDS(2),BOUNDS(3),BOUNDS(4)/'B','O','U','N'/
      DATA ENDATA(1),ENDATA(2),ENDATA(3),ENDATA(4)/'E','N','D','A'/
      DATA GEM(1),GEM(2),GEM(3),GEM(4)/'*','G','E','O'/
      DATA ARI(1),ARI(2),ARI(3),ARI(4)/'*','A','R','I'/
C---
C---  INITIALIZE FR,FX,UP,LO AND MI (BOUND TYPE NAMES)
C---
      DATA FR(1),FR(2),FX(1),FX(2)/'F','R','F','X'/
      DATA UP(1),UP(2),LO(1),LO(2)/'U','P','L','O'/
      DATA MI(1),MI(2)/'M','I'/
C---
C---  INITIALIZE BLANK AND ASTERSK (SINGLE CHARACTERS)
C---
      DATA BLANK,ASTERSK/' ','*'/
C---
C---  INITIALIZE TRUE AND FALSE (LOGICALS)
C---
      DATA TRUE,FALSE/.TRUE.,.FALSE./
```

```
C---
C---  INITIALIZE 2-DIM. ARRAY, PRNAME, TO TWO CHARACTER EQUIVALENTS
C---  OF EACH POSSIBLE PERIOD NUMBER (I.E. ('0','0') TO ('9','9') )
C---  LOOP OVER I TO SELECT FIRST DIGIT
C---           J TO SELECT SECOND DIGIT
C---
      DO 20 I=1,10
         DO 20 J=1,10
C---
C---     (N-TH LEVEL OF ARRAY PRNAME CORRESPONDS TO N-TH POSITION
C---      IN THE SEQUENCE 00,01,02,03,...,99 )
C---
         N = (I-1)*10 + J
         PRNAME(N,1) = DIGIT(I)
         PRNAME(N,2) = DIGIT(J)
C---
C---     EXAMPLES -
C---     09 IS IN 10-TH POSITION           10 IS IN 11-TH POSITION
C---     N=10 RESULTS FROM I=1,J=10        N=11 RESULTS FROM I=2,J=1
C---     I=1 SELECTS DIGIT '0'             I=2 SELECTS DIGIT '1'
C---     J=10 SELECTS DIGIT '9'            J=1 SELECTS DIGIT '0'
C---
20    CONTINUE
C---
C---  INITIALIZE MAXPR (MAXIMUM NUMBER OF PERIODS IN OUTPUT MODEL)
C---
      MAXPR = 20
C---
C---  INITIALIZE MAXENT (MAXIMUM NUMBER OF ENTRIES IN OUTPUT TABLES)
C---         AND LASTIX (INDEX OF LAST ENTRY IN OUTPUT TABLES)
C---
      MAXENT = 100
      LASTIX = 0
C---
C---  (TWO INPUT CARDS ARE LOCATED AT THE END OF THIS DECK)
C---  READ AND IGNORE DUMMY CARD (USED ONLY FOR IDENTIFYING FIELDS
C---                             OF NEXT CARD WHEN KEYING IN DATA)
C---
      READ (5,900) CARD
900   FORMAT (A80)
C---
C---  READ AGGREGATION SCHEME CARD
C---
      READ (5,902) LISTIN
902   FORMAT(20I3)
C---
C---  (LISTIN - THE I-TH NUMBER IN LISTIN IS THE NUMBER OF PERIODS
C---            FROM THE INPUT MODEL TO BE AGGREGATED WHEN FORMING
C---            THE I-TH PERIOD OF THE OUTPUT MODEL)
C---
C---  COMPUTE NPROUT (NUMBER OF PERIODS IN OUTPUT MODEL
C---                  = NUMBER OF NONZEROS IN LISTIN)
```

```
C---
      DO 40 N=1,MAXPR
         IF (LISTIN(N).EQ.0) GO TO 50
40    CONTINUE
C---
C---  NO ZEROS IN LISTIN.
C---
      N = MAXPR + 1
C---
C---  N EQUALS NUMBER OF NONZEROS IN LISTIN PLUS ONE.
C---
50    NPROUT = N - 1
      IF (NPROUT.EQ.0) STOP
C---
C---  COMPUTE INOUT (I-TH ELEMENT OF ARRAY INOUT IS THE PERIOD
C---   NUMBER OF THE LAST PERIOD OF THE INPUT MODEL TO BE
C---   AGGREGATED INTO THE I-TH PERIOD OF THE OUTPUT MODEL)
C---
      LASTPR = 0
      DO 60 I=1,MAXPR
         LASTPR = LASTPR + LISTIN(I)
         INOUT(I) = LASTPR
60    CONTINUE
C---
C---  SAVE NPRIN (NUMBER OF PERIODS IN THE INPUT MODEL)
C---
      NPRIN = INOUT(NPROUT)
C---
C---  COMMON BLOCK1 IS NOW WELL-DEFINED.
C---
C---
C---  PRINT OUT AGGREGATION SCHEME
C---
      WRITE (6,991)
991   FORMAT (1H ,'INPUT CARDS -')
      WRITE (6,992) CARD
992   FORMAT (1H ,A80)
      WRITE (6,993) LISTIN
993   FORMAT (1H ,20I3,/)
      WRITE (6,994)
994   FORMAT (1H ,'LIST OF LAST INPUT PERIOD NUMBER CORRESPONDING',
     1            ' TO EACH OUTPUT PERIOD NUMBER -',/)
      WRITE (6,995) CARD
995   FORMAT (1H ,'OUTPUT PERIOD NUMBER',A80)
      WRITE (6,996) (INOUT(I),I=1,NPROUT)
996   FORMAT (1H ,'INPUT PERIOD NUMBER ',20I3)
      WRITE (6,997)
997   FORMAT (1H1,'LIST OF WARNING MESSAGES - SEE COMMENTS WITHIN ',
     1            'PROGRAM FOR DEFAULT ACTION',/)
C---
C---
C---
```

```
C SWITCH TO INPUT MODEL FILE
C---
C---  COPY "NAME" CARD AND "ROWS" CARD TO OUTPUT FILE
C---
      DO 80 I=1,2
         READ (8,900) CARD
         WRITE (9,900) CARD
80    CONTINUE
C---
C---
C---
C---
C---
C ROWS SEGMENT BEGINS
C---
C---  BEGIN ROWS CARD CYCLE - ONE PASS FOR EACH CARD READ
C---
C---  SET PASS1 (FIRST PASS INDICATOR) ON OR OFF
C---
      PASS1 = TRUE
      GO TO 105
100   PASS1 = FALSE
C---
C---  READ A CARD USING ROW CAPD FORMAT
C---
105   READ (8,910) TYPIN,NAMEIN
910   FORMAT (4A1,8A1)
C---
C---  (TYPIN - ROW TYPE)
C---  (NAMEIN - ROW NAME)
C---
C---  SKIP COMMENT CARDS
C---
      IF (TYPIN(1).EQ.ASTERSK) GO TO 105
C---
C---  REFORM ROW NAME BY CALLING SUBROUTINE RENAME
C---
      CALL RENAME(NAMEIN,NEWNAME,INEWPR,NBLANK)
C---
C---  (NEWNAME - NAME OF INPUT ROW AS IT IS TO APPEAR ON OUTPUT)
C---  (NBLANK - NUMBER OF BLANKS AT END OF NEWNAME)
C---
C---  ON FIRST PASS, BRANCH TO "NEW ROW NAME"
C---
      IF (PASS1) GO TO 170
C---
C---  (ASSUME MPS INPUT FILE SORTED SO THAT ROW NAMES WITH THE
C---   SAME ROOT ARE GROUPED TOGETHER IN ASCENDING ORDER OF PERIOD
C---   NUMBER. CONSEQUENTLY, AFTER NAMES ARE REFORMED THE ROWS
C---   WHICH ARE TO BE AGGREGATED WILL BE GROUPED TOGETHER UNDER
C---   A COMMON (OUTPUT) ROW NAME)
C---
```

```
C---  COMPARE NEWNAME WITH ROWNAME (THE NEWNAME OF THE PREVIOUS CARD)
C---  (TO SAVE TIME, CONSIDER ONLY NONBLANK CHARACTERS OF NEWNAME)
C---
      NONBLK = 8 - NBLANK
      DO 110 I=1,NONBLK
C---
C---     IF NO MATCH, BRANCH TO "OUTPUT PREVIOUS ROW"
C---
         IF (NEWNAME(I).NE.ROWNAME(I)) GO TO 150
 110  CONTINUE
C---
C---  NAMES MATCH. (ROW ID FOR NEWNAME HAS ALREADY BEEN SET UP)
C---  PRINT WARNING IF TYPE DIFFERS FROM TYPE OF PREVIOUS CARD,THEN
C---  GO READ A NEW ROWS CARD
C---
      DO 120 I=1,2
         IF (TYPIN(I).NE.TYPE(I)) GO TO 130
 120  CONTINUE
      GO TO 100
 130  WRITE (6,950) TYPIN,NAMEIN,TYPE,ROWNAME
 950  FORMAT (1H ,'ROW ID INPUT AS ',4A1,8A1,' WILL BE OUTPUT AS ',
     1        4A1,8A1,' ** TYPE CHANGE')
      GO TO 100
C---
C---  OUTPUT PREVIOUS ROW ID USING ROW CARD FORMAT
C---
 150  WRITE (9,910) TYPE,ROWNAME
C---
C---  IF NAMES DID NOT MATCH BECAUSE NEW CARD WAS "COLUMNS" CARD
C---  BRANCH TO "COLUMN SEGMENT BEGINS"
C---
      DO 160 I=1,4
C---
C---     IF TYPE NOT EQUAL TO 'C','O','L','U', BRANCH TO "NEW ROW"
C---
         IF (TYPIN(I).NE.COL(I)) GO TO 170
 160  CONTINUE
      GO TO 200
C---
C---  NEW ROW NAME ENCOUNTERED.
C---
C---  RESET OUTPUT BUFFERS - SAVE NEWNAME AS ROWNAME
C---                       - SAVE TYPIN AS TYPE
C---
 170  DO 180 I=1,8
         ROWNAME(I) = NEWNAME(I)
 180  CONTINUE
      DO 190 I=1,4
         TYPE(I) = TYPIN(I)
 190  CONTINUE
C---
C---  GO READ A NEW ROWS CARD
```

```
C---
      GO TO 100
C---
C---
C---
C---
C---
C COLUMN SEGMENT BEGINS
C---
C---  OUTPUT "COLUMNS" CARD
C---
200   WRITE (9,912)
912   FORMAT ('COLUMNS')
      COUNT=1
      ARITH=TRUE
      GEOM=FALSE
      MINUS1=FALSE
C---
C---  BEGIN COLUMN CARD CYCLE - ONE PASS FOR EACH CARD READ
C---
C---  SET PASS1 (FIRST PASS INDICATOR) ON OR OFF
C---
      PASS1 = TRUE
      GO TO 210
205   PASS1 = FALSE
      GO TO 210
C---
C---  READ A CARD USING COLUMN CARD FORMAT
C---
207   DO 209 J=2,4
      IF (TYPIN(J).N.E.ARI(J)) GO TO 210
209   CONTINUE
      GEOM=FALSE
210   READ (8,914) TYPIN,CNAME,(RNAME(1,I),I=1,8),RVALU(1),
     1                          (RNAME(2,J),J=1,8),RVALU(2)
914   FORMAT (4A1,8A1,2X,2(8A1,2X,F12.6,3X))
C---
C---  (TYPIN - BLANK)
C---  (CNAME - COLUMN NAME)
C---  (RNAME(1) AND (2) - ROW NAMES OF MATRIX ENTRIES)
C---  (RVALU(1) AND (2) - MATRIX ENTRIES)
C---
C---  SKIP COMMENT CARDS
C---
      IF (TYPIN(1).NE.ASTERSK) GO TO 215
      DO 229 J=2,4
      IF (TYPIN(J).NE.GEM(J)) GO TO 207
229   CONTINUE
      GEOM=TRUE
      GO TO 210
C---
C---  REFORM COLUMN NAME BY CALLING SUBROUTINE RENAME
```

```
      C---
  215 CALL RENAME(CNAME,NEWNAME,INEWPR,NBLANK)
      C---
      C--- (NEWNAME - NAME OF INPUT COLUMN AS IT IS TO APPEAR ON OUTPUT)
      C--- (NBLANK - NUMBER OF BLANKS ON END OF NEWNAME)
      C---
      C--- ON FIRST PASS, BRANCH TO "NEW COLUMN"
      C---
           IF (PASS1) GO TO 250
      C---
      C--- (ASSUME MPS INPUT FILE IS SORTED SO THAT COLUMN NAMES WITH
      C---  THE SAME ROOT ARE GROUPED TOGETHER IN ASCENDING ORDER OF
      C---  PERIOD NUMBER. CONSEQUENTLY, AFTER NAMES HAVE BEEN REFORMED
      C---  THE COLUMNS WHICH ARE TO BE AGGREGATED WILL BE GROUPED
      C---  TOGETHER UNDER A COMMON (OUTPUT) COLUMN NAME)
      C---
      C--- IF NEWNAME IS ALL BLANKS BRANCH TO "UPDATE OUTPUT TABLES"
      C--- (ASSUME NEW CARD WAS "RHS" CARD)
      C---
           IF (NBLANK.EQ.8) GO TO 230
      C---
      C--- COMPARE NEWNAME WITH COLNAME (THE NEWNAME OF THE PREVIOUS CARD)
      C--- (CONSIDER ONLY NONBLANK CHARACTERS OF NEWNAME)
      C---
           NONBLK = 8 - NBLANK
           DO 220 I=1,NONBLK
      C---
      C---    IF NO MATCH, BRANCH TO "UPDATE OUTPUT TABLES"
      C---
              IF (NEWNAME(I).NE.COLNAME(I)) GO TO 230
  220 CONTINUE
      C---
      C--- NAMES MATCH. OUTPUT TABLES FOR THIS COLUMN HAVE ALREADY BEEN
      C--- SET UP.
      C--- BRANCH TO "PROCESS ROW NAMES AND VALUES FROM CURRENT CARD"
      C---
           GO TO 275
      C---
      C--- UPDATE OUTPUT TABLES FOR PREVIOUS COLUMN WITH
      C--- ROWNAME (LAST ROWNAME ENCOUNTERED) AND ROWVALU (ASSOCIATED
      C--- MPS MATRIX ENTRY) BY CALLING SUBROUTINE UPDATE
      C---
  230 CALL UPDATE(LASTIX,ROWNAME,ROWVALU,PBLANK,ARITH,GEOM,COUNT,
     1            MINUS1)
      C---
      C--- (LASTIX - INDEX OF LAST ENTRY IN OUTPUT TABLES)
      C--- (PBLANK - NUMBER OF BLANKS ON END OF ROWNAME)
      C---
      C---
      C--- OUTPUT PREVIOUS COLUMN BY CALLING SUBROUTINE COLOUT
      C--- (COMMON BLOCK2 SHOULD BE WELL-DEFINED AT THIS POINT)
      C--- (COLNAME - OUTPUT NAME OF AGGREGATED COLUMN)
```

```
C---  (NAMETAB - LIST OF AGGREGATED ROW NAMES FOR THIS COLUMN)
C---  (VALUTAB - CORRESPONDING LIST OF AGGREGATED MPS MATRIX ENTRIES)
C---  (LASTIX - INDEX OF LAST ENTRY IN NAMETAB AND VALUTAB)
C---
      CALL COLOUT(LASTIX)
C---
C---  IF NAMES DID NOT MATCH BECAUSE NEW CARD WAS "RHS" CARD,
C---  BRANCH TO "RHS SEGMENT BEGINS"
C---
      DO 240 I=1,4
C---
C---     IF TYPE NOT EQUAL TO 'R','H','S',' ', BRANCH TO "NEW COLUMN"
C---
         IF (TYPIN(I).NE.RHS(I)) GO TO 250
240   CONTINUE
      GO TO 400
C---
C---  NEW COLUMN ENCOUNTERED.
C---
C---  ERASE OUTPUT TABLES, NAMETAB AND VALUTAB (A SAFETY MEASURE)
C---
250   NERASE = LASTIX + 1
      NERASE = MIN0(MAXENT,NERASE)
      DO 260 I=1,NERASE
         VALUTAB(I) = 0.
         DO 260 J=1,8
            NAMETAB(I,J) = BLANK
260   CONTINUE
C---
C---  RESET LASTIX (INDEX OF LAST ENTRY IN OUTPUT TABLES) TO ZERO
C---
      LASTIX = 0
C---
C---  SAVE NEW COLUMN NAME AS COLNAME AND ERASE ROWNAME (PREVIOUS
C---   ROWNAME PROCESSED)
C---
      DO 270 I=1,8
         COLNAME(I) = NEWNAME(I)
         ROWNAME(I) = BLANK
270   CONTINUE
C---
C---  RESET PBLANK (NUMBER OF BLANKS AT END OF PREVIOUS ROWNAME)
C---        TO EIGHT AND
C---        ROWVALU (AGGREGATE MPS ENTRY FOR PREVIOUS ROW NAME) TO ZERO
C---
      PBLANK = 8
      ROWVALU = 0.
C---
C---  PROCESS ROW NAMES AND VALUES FROM CURRENT CARD
C---
C---  LOOP ONCE FOR EACH (OF TWO) ROW NAMES (LOOP OVER I)
C---
```

```
      275   DO 370 I=1,2
      C---
      C---     (ASSUME MPS INPUT FILE IS SORTED SO THAT,FOR EACH INPUT
      C---      COLUMN, ROW NAMES WITH THE SAME ROOT ARE GROUPED TOGETHER
      C---      IN ASCENDING ORDER OF PERIOD NUMBER. CONSEQUENTLY, AFTER
      C---      ROW NAMES ARE REFORMED, THE ROWS WHICH ARE TO BE AGGREGATED
      C---      FOR THAT INPUT COLUMN WILL BE GROUPED TOGETHER UNDER A
      C---      COMMON (OUTPUT) ROW NAME. HOWEVER, SINCE SEVERAL INPUT
      C---      COLUMNS MAY NEED TO BE AGGREGATED UNDER ONE COLUMN NAME
      C---      IT IS NECESSARY TO MAINTAIN NAMETAB (TABLE OF (OUTPUT) ROW
      C---      NAMES ENCOUNTERED FOR THAT (OUTPUT) COLUMN), AND VALUTAB
      C---      (TABLE OF CORRESPONDING (AGGREGATE) MPS MATRIX ENTRIES)
      C---
      C---     MOVE RNAME(I) (INPUT ROW NAME BEING PROCESSED) INTO NAMEIN
      C---     (CALL STATEMENT WILL NOT ACCEPT AN IMPLIED DO LOOP)
      C---
               DO 280 J=1,8
                  NAMEIN(J) = RNAME(I,J)
      280      CONTINUE
      C---
      C---     REFORM INPUT ROW NAME BY CALLING SUBROUTINE RENAME
      C---
               CALL RENAME(NAMEIN,NEWNAME,INEWPR,NBLANK)
      C---
      C---     (NEWNAME - NAME OF INPUT ROW AS IT IS TO APPEAR ON OUTPUT)
      C---     (NBLANK - NUMBER OF BLANKS ON END OF NEWNAME)
      C---
      C---     IF INPUT ROW NAME WAS ALL BLANKS, SKIP TO "END OF LOOP"
      C---
               IF (NBLANK.EQ.8) GO TO 370
      C---
      C---     IF PREVIOUS ROW NAME ALL BLANKS, BRANCH TO "NEW ROW NAME"
      C---
               IF (PBLANK.EQ.8) GO TO 350
      C---
      C---     COMPARE NEWNAME WITH ROWNAME (PREVIOUS ROWNAME PROCESSED)
      C---     (CONSIDER ONLY NONBLANK CHARACTERS)
      C---
               NONBLK = 8 - NBLANK
               DO 285 J=1,NONBLK
      C---
      C---        IF NO MATCH, BRANCH TO "UPDATE OUTPUT TABLES"
      C---
                  IF (NEWNAME(J).NE.ROWNAME(J)) GO TO 290
      285      CONTINUE
               IF (ARITH) GO TO 288
               IF (RVALU(I).LT.0.0) GO TO 287
               ROWVALU=ROWVALU*RVALU(I)
      286      COUNT=COUNT+1
               GO TO 370
      287      ROWVALU=ROWVALU*ABS(RVALU(I))
               GO TO 286
```

```
C---
C---     NAMES MATCH.
C---     ADD RVALU (CURRENT MATRIX ENTRY) TO ROWVALU (PREVIOUS TOTAL)
C---     AND BRANCH TO "END OF LOOP"
C---
288      ROWVALU = ROWVALU + RVALU(I)
         GO TO 370
C---
C---     UPDATE OUTPUT TABLES WITH ROWNAME (PREVIOUS ROW NAME)
C---     AND ROWVALU (CORRESPONDING MATRIX ENTRY) BY CALLING
C---     SUBROUTINE UPDATE
C---
290      CALL UPDATE(LASTIX,ROWNAME,ROWVALU,PBLANK,ARITH,GEOM,COUNT,
     1               MINUS1)
C---
C---     (LASTIX - INDEX OF LAST ENTRY IN OUTPUT TABLES)
C---     (PBLANK - NUMBER OF BLANKS ON END OF ROWNAME)
C---
C---    NEW ROWNAME ENCOUNTERED.
C---    SAVE NEWNAME (CURRENT ROW NAME) AS ROWNAME (PREVIOUS ROW NAME)
C---    SAVE NBLANK (NUMBER OF BLANKS IN NEWNAME) AS PBLANK
C---    SAVE RVALU (CORRESPONDING MATRIX ENTRY) AS ROWVALU
C---
350     DO 360 L=1,8
           ROWNAME(L) = NEWNAME(L)
360     CONTINUE
        PBLANK = NBLANK
        IF (ARITH) GO TO 364
        IF (RVALU(I).LT.0.0) GO TO 365
364     ROWVALU=RVALU(I)
        MINUS1=FALSE
        GO TO 370
365     ROWVALU=ABS(RVALU(I))
        MINUS1=TRUE
C---
C---    END OF LOOP
C---
370  CONTINUE
C---
C--- BOTH ROW NAMES FROM INPUT CARD HAVE NOW BEEN PROCESSED.
C--- GO READ ANOTHER COLUMN CARD
C---
     GO TO 205
C---
C---
C---
C---
C---
C RHS SEGMENT BEGINS
C---
C--- OUTPUT "RHS" CARD
C---
```

```
 400   WRITE (9,916)
 916   FORMAT ('RHS')
 C---
 C---  RESET ROWNAME (PREVIOUS ROW NAME) TO BLANKS AND
 C---        ROWVALU (ASSOCIATED MPS MATRIX ENTRY) TO ZERO AND
 C---        PBLANK (NUMBER OF BLANKS ON END OF ROWNAME) TO EIGHT
 C---
       DO 405 I=1,8
          ROWNAME(I) = BLANK
 405   CONTINUE
       ROWVALU = 0.
       PBLANK = 8
 C---
 C---  BEGIN RHS CARD CYCLE - ONE PASS FOR EACH CARD READ
 C---
 C---  SET PASS1 (FIRST PASS INDICATOR) ON OR OFF
 C---
       PASS1 = TRUE
       GO TO 420
 410   PASS1 = FALSE
 C---
 C---  READ A CARD USING COLUMN CARD FORMAT
 C---
 420   READ (8,914) TYPIN,CNAME,(RNAME(1,I),I=1,8),RVALU(1),
      1                           (RNAME(2,J),J=1,8),RVALU(2)
 C---
 C---  (TYPIN - BLANK)
 C---  (CNAME - RHS NAME)
 C---  (RNAME(1) AND (2) - ROW NAMES OF RHS ENTRIES)
 C---  (RVALU(1) AND (2) - RHS VALUES)
 C---
 C---  ON FIRST PASS ONLY, SAVE CNAME AS COLNAME (NAME OF RHS)
 C---                           TYPIN AS TYPE (BLANKS)
 C---
       IF (.NOT.PASS1) GO TO 440
       DO 430 I=1,8
          COLNAME(I) = CNAME(I)
 430   CONTINUE
       DO 435 I=1,4
          TYPE(I) = TYPIN(I)
 435   CONTINUE
 C---
 C---  PROCESS ROW NAME AND VALUES FROM CURRENT CARD
 C---
 C---  LOOP ONCE FOR EACH (OF TWO) ROW NAMES (LOOP OVER I)
 C---
 440   DO 530 I=1,2
 C---
 C---     (ASSUME MPS INPUT FILE SORTED SO THAT ROW NAMES WITH SAME
 C---      ROOT ARE GROUPED TOGETHER IN ASCENDING ORDER OF PERIOD
 C---      NUMBER. CONSEQUENTLY, AFTER ROW NAMES ARE REFORMED, THE
 C---      ROWS WHICH ARE TO BE AGGREGATED ON THE RHS WILL BE
```

```
C---      GROUPED TOGETHER UNDER A COMMON (OUTPUT) ROW NAME)
C---
C---     MOVE RNAME(I) (INPUT ROW NAME BEING PROCESSED) INTO NAMEIN
C---     (CALL STATEMENT WILL NOT ACCEPT AN IMPLIED DO LOOP)
C---
         DO 445 J=1,8
            NAMEIN(J) = RNAME(I,J)
445      CONTINUE
C---
C---     REFORM INPUT ROW NAME BY CALLING SUBROUTINE RENAME
C---
         CALL RENAME(NAMEIN,NEWNAME,INEWPR,NBLANK)
C---
C---     (NEWNAME - NAME OF INPUT ROW AS IT IS TO APPEAR ON OUTPUT)
C---     (NBLANK - NUMBER OF BLANKS AT END OF NEWNAME)
C---
C---     IF PREVIOUS ROW NAME ALL BLANKS (I.E. FIRST NAME ON FIRST CARD)
C---     BRANCH TO "NEW ROW NAME"
C---
         IF (PBLANK.EQ.8) GO TO 475
C---
C---     IF INPUT ROW NAME NOT ALL BLANKS
C---     BRANCH TO "COMPARE NEWNAME WITH ROWNAME"
C---
         IF (NBLANK.LT.8) GO TO 450
C---
C---     INPUT ROW NAME ALL BLANKS.
C---     IF THIS IS FIRST NAME ON CARD ASSUME IT IS "BOUNDS" CARD AND
C---     BRANCH TO "OUTPUT PREVIOUS ROW ENTRY"
C---     OTHERWISE BRANCH TO "END OF LOOP"
C---
         IF (I.EQ.1) GO TO 470
         GO TO 530
C---
C---     COMPARE NEWNAME WITH ROWNAME (PREVIOUS ROW NAME)
C---     (CONSIDER ONLY NONBLANK CHARACTERS OF NEWNAME)
C---
450      NONBLK = 8 - NBLANK
         DO 460 L=1,NONBLK
C---
C---        IF NO MATCH, BRANCH TO "OUTPUT PREVIOUS ROW ENTRY"
C---
            IF (NEWNAME(L).NE.ROWNAME(L)) GO TO 470
460      CONTINUE
C---
C---     NAMES MATCH.
C---     ADD CURRENT RHS ENTRY TO ROWVALU (PREVIOUS TOTAL) AND
C---     BRANCH TO "END OF LOOP"
C---
         ROWVALU = ROWVALU + RVALU(I)
         GO TO 530
C---
```

```
C---     OUTPUT PREVIOUS ROW ENTRY BY CALLING SUBROUTINE CARDOUT
C---
470      CALL CARDOUT(TYPE,COLNAME,1,ROWNAME,ROWVALU,ROWNAME,ROWVALU)
C---
C---     (3-RD ARGUMENT IN CALL IS NUMBER OF ROW ENTRIES SUBMITTED
C---      FOR OUTPUT - IN THIS CASE ONLY ONE SO THE 6-TH AND 7-TH
C---      ARGUMENTS WILL BE IGNORED)
C---
C---     NEW ROW NAME ENCOUNTERED.
C---     IF THIS IS FIRST NAME ON CARD CHECK IF "BOUNDS" CARD
C---     OTHERWISE BRANCH TO "RESET OUTPUT BUFFERS"
C---
475      IF (I.NE.1) GO TO 500
         DO 480 L=1,4
C---
C---        IF TYPE NOT EQUAL TO 'B','O','U','N',
C---        BRANCH TO "RESET OUTPUT BUFFERS"
C---
            IF (TYPIN(L).NE.BOUNDS(L)) GO TO 500
480      CONTINUE
         GO TO 600
C---
C---     RESET OUTPUT BUFFERS - SAVE NEWNAME AS ROWNAME
C---                          - SAVE NBLANK AS PBLANK
C---                          - SAVE RVALU AS ROWVALU
C---
500      DO 510 L=1,8
            ROWNAME(L) = NEWNAME(L)
510      CONTINUE
         ROWVALU = RVALU(I)
         PBLANK = NBLANK
C---
C---     END OF LOOP
C---
530   CONTINUE
C---
C---  BOTH ROW NAMES FROM INPUT CARD HAVE NOW BEEN PROCESSED.
C---  GO READ ANOTHER RHS CARD
C---
      GO TO 410
C---
C---
C---
C---
C---
C BOUNDS SEGMENT BEGINS
C---
C---  OUTPUT "BOUNDS" CARD
C---
600   WRITE (9,918)
918   FORMAT ('BOUNDS')
C---
```

```
C---  BEGIN BOUNDS CARD CYCLE - ONE PASS FOR EACH CARD READ
C---
C---
C---  SET PASS1 (FIRST PASS INDICATOR) ON OR OFF
C---
      PASS1 = TRUE
      GOTO 625
620   PASS1 = FALSE
C---
C---  READ A CARD USING BOUND CARD FORMAT
C---
625   READ (8,920) TYPIN,(RNAME(1,I),I=1,8),CNAME,VALUE
920   FORMAT (4A1,8A1,2X,8A1,2X,F12.6)
C---
C---  (TYPIN - BOUND TYPE)
C---  (RNAME(1) - BOUND NAME)
C---  (CNAME - COLUMN NAME)
C---  (VALUE - BOUND VALUE)  (ASSUME ONLY ONE VALUE INPUT PER CARD)
C---
C---  SKIP COMMENT CARDS
C---
      IF (TYPIN(1).EQ.ASTERSK) GO TO 625
C---
C---  ON FIRST PASS ONLY, SAVE RNAME(1) AS ROWNAME (NAME OF BOUNDS "ROW")
C---
      IF (.NOT.PASS1) GO TO 640
      DO 635 I=1,8
         ROWNAME(I) = RNAME(1,I)
635   CONTINUE
C---
C---  REFORM INPUT COLUMN NAME BY CALLING SUBROUTINE RENAME
C---
640   CALL RENAME(CNAME,NEWNAME,INEWPR,NBLANK)
C---
C---  (NEWNAME - NAME OF INPUT COLUMN NAME AS IT IS TO APPEAR ON OUTPUT)
C---  (NBLANK - NUMBER OF BLANKS AT END OF NEWNAME)
C---  (INEWPR - INDEX OF OUTPUT PERIOD NUMBER
C---          - EQUALS ZERO IF CNAME DID NOT END WITH VALID INPUT PERIOD
C---            NUMBER)
C---
C---  IF FIRST PASS, BRANCH TO "NEW NAME/TYPE"
C---
      IF (PASS1) GO TO 700
C---
C---  (ASSUME MPS INPUT FILE SORTED SO THAT FOR EACH BOUND TYPE
C---   ENCOUNTERED, COLUMN NAMES WITH THE SAME ROOT ARE GROUPED TOGETHER
C---   IN ASCENDING ORDER OF PERIOD NUMBER. CONSEQUENTLY, AFTER COLUMN
C---   NAMES ARE REFORMED, BOUNDS OF THE SAME TYPE WHICH ARE TO BE
C---   AGGREGATED WILL BE GROUPED TOGETHER UNDER A COMMON (OUTPUT)
C---   COLUMN NAME)
C---
C---  IF NEWNAME ALL BLANKS BRANCH TO "CHANGE IN BOUND NAME/TYPE"
```

```
C---  (ASSUME "ENDATA" CARD ENCOUNTERED)
C---
      IF (NBLANK.EQ.8) GO TO 670
C---
C---  COMPARE NEWNAME WITH COLNAME (THE NEWNAME OF THE PREVIOUS CARD)
C---  (CONSIDER ONLY THE NON-BLANK CHARACTERS OF NEWNAME)
C---
      NONBLK = 8 - NBLANK
      DO 650 I=1,NONBLK
C---
C---     IF NO MATCH, BRANCH TO "CHANGE IN BOUND NAME/TYPE"
C---
         IF (NEWNAME(I).NE.COLNAME(I)) GO TO 670
 650  CONTINUE
C---
C---  NAMES MATCH. NOW COMPARE BOUND TYPES
C---
      DO 660 I=1,4
C---
C---     IF NO MATCH, BRANCH TO "CHANGE IN BOUND NAME/TYPE"
C---
         IF (TYPIN(I).NE.TYPE(I)) GO TO 670
 660  CONTINUE
C---
C---  NAMES AND BOUND TYPE MATCH.
C---  INCREMENT NBOUNDS (NUMBER OF BOUNDS ENCOUNTERED)
C---  ADD VALUE TO BDVALU (AGGREGATE BOUND VALUE)
C---  GO READ A NEW CARD
C---
      NBOUNDS = NBOUNDS + 1
      BDVALU = BDVALU + VALUE
      GO TO 620
C---
C---  CHANGE IN BOUND NAME/TYPE.
C---
C---  IDENTIFY BOUND TYPE OF PREVIOUS NAME/TYPE
C---
 670  IF (TYPE(3).EQ.FR(2).OR.TYPE(3).EQ.MI(2)) GO TO 680
      IF (TYPE(3).EQ.LO(2)) GO TO 685
      IF (TYPE(3).EQ.FX(2).OR.TYPE(3).EQ.UP(2)) GO TO 690
C---
C---  TYPE WAS NOT ONE OF (MI,FR,FX,UP, OR LO).
C---  PRINT WARNING AND TREAT SAME AS "FREE" OR "MINUS INFINITY"
C---
      WRITE (6,956) TYPE,ROWNAME,COLNAME,BDVALU
 956  FORMAT (1H ,4A1,8A1,2X,8A1,2X,F12.6,' ** UNRECOGNIZED BOUND TYPE')
C---
C---  BOUND TYPE WAS EITHER "FREE" OR "MINUS INFINITY".
C---  OUTPUT BOUND CARD BY CALLING SUBROUTINE CARDOUT AND
C---  BRANCH TO "NEW NAME/TYPE"
C---
 680  CALL CARDOUT(TYPE,ROWNAME,1,COLNAME,BDVALU,COLNAME,BDVALU)
```

```
      GO TO 700
C---
C---  (3-RD ARGUMENT IN CALL IS NUMBER OF ENTRIES SUBMITTED FOR
C---   OUTPUT - IN THIS CASE ONLY ONE SO THE 6-TH AND 7-TH
C---   ARGUMENTS WILL BE IGNORED)
C---
C---
C---  BOUND TYPE WAS "LOWER".
C---  AVERAGE BDVALU (AGGREGATE BOUND VALUE) OVER NPR (NUMBER OF PERIODS
C---  IN AGGREGATION)
C---  OUTPUT BOUND CARD BY CALLING SUBROUTINE CARDOUT AND
C---  BRANCH TO "NEW NAME/TYPE"
C---
685   BDVALU = BDVALU/NPR
      CALL CARDOUT(TYPE,ROWNAME,1,COLNAME,BDVALU,COLNAME,BDVALU)
      GO TO 700
C---
C---  BOUND TYPE WAS EITHER "FIXED" OR "UPPER".
C---  COMPARE NBOUNDS (NUMBER OF BOUNDS ENCOUNTERED) WITH NPR (NUMBER
C---  OF PERIODS IN AGGREGATION)
C---  IF EQUAL- TREAT SAME AS LOWER BOUND - BRANCH TO "TYPE WAS LOWER"
C---
690   IF (NBOUNDS.EQ.NPR) GO TO 685
C---
C---  INCORRECT NUMBER OF BOUNDS ENCOUNTERED. (ASSUME TOO FEW)
C---  (SINCE DEFAULT UPPER BOUND IS INFINITY THE AVERAGE UPPER BOUND
C---   MUST BE INFINITY)
C---  PRINT WARNING
C---  IF UPPER BOUND DO NOT OUTPUT A BOUND CARD BUT
C---                 BRANCH TO "NEW NAME/TYPE"
C---  IF FIXED BOUND CHANGE TYPE TO "LOWER" AND
C---                 BRANCH TO "TYPE WAS LOWER"
C---
      WRITE (6,958) TYPE,ROWNAME,COLNAME,BDVALU,NPR,NBOUNDS
958   FORMAT (1H ,4A1,8A1,2X,8A1,2X,F12.6,' BOUNDS EXPECTED ',I3,
     1         ' BOUNDS ENCOUNTERED ',I3,' ** TOO FEW BOUNDS')
      IF (TYPE(3).EQ.UP(2)) GO TO 700
      TYPE(2) = LO(1)
      TYPE(3) = LO(2)
      GO TO 685
C---
C---  NEW NAME/TYPE ENCOUNTERED.
C---  IF BREAK CAUSED BY "ENDATA" CARD BRANCH TO "END SEGMENT"
C---
700   DO 710 I=1,4
C---
C---     IF TYPE ON CURRENT CARD NOT EQUAL TO 'E','N','D','A'
C---     BRANCH TO "RESET OUTPUT BUFFERS"
C---
         IF (TYPIN(I).NE.ENDATA(I)) GO TO 720
710   CONTINUE
      GO TO 800
```

```
C---
C---  RESET OUTPUT BUFFERS - SAVE NEWNAME AS COLNAME
C---                       - SAVE TYPIN AS TYPE
C---                       - SAVE VALUE AS BDVALU
C---                       - RESET NBOUNDS TO ONE
C---
720   DO 730 I=1,8
         COLNAME(I) = NEWNAME(I)
730   CONTINUE
      DO 735 I=1,4
         TYPE(I) = TYPIN(I)
735   CONTINUE
      BDVALU = VALUE
      NBOUNDS = 1
C---
C---  IF INEWPR (INDEX OF OUTPUT PERIOD NUMBER) IS INVALID SET NPR TO
C---  ONE AND GO READ A NEW CARD
C---
      IF (INEWPR.GT.0) GO TO 740
      NPR = 1
      GO TO 620
C---
C---  VALID INEWPR.
C---  LOOKUP NPR (NUMBER OF PERIODS IN AGGREGATION) IN TABLE LISTIN
C---  (I-TH NUMBER IN LISTIN IS NUMBER OF PERIODS FROM INPUT MODEL
C---   TO BE AGGREGATED WHEN FORMING I-TH PERIOD OF OUTPUT MODEL)
C---
740   NPR = LISTIN(INEWPR)
C---
C---  GO READ A NEW BOUNDS CARD
C---
      GO TO 620
C---
C---
C---
C---
C END SEGMENT
C---
C---  OUTPUT "ENDATA" CARD AND STOP
C---
800   WRITE (9,922)
922   FORMAT('ENDATA')
      STOP
      END
C---
C---
C---
C---  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C---
C---
C---
C---  SUBROUTINE RENAME
```

```
C---
C---           - INPUT OLDNAME (ANY 8 CHARACTER NAME)
C---           - SELECTS LAST TWO NON-BLANK CHARACTERS OF NAME
C---           - CONVERTS THESE TO A (TWO-DIGIT) INTEGER NUMBER
C---           - RETURNS IF NOT A VALID INPUT PERIOD NUMBER
C---           - CONVERTS TO (TWO-CHARACTER) NEW PERIOD NUMBER
C---             ACCORDING TO AGGREGATION SCHEME
C---           - SUBSTITUTES NEW PERIOD NUMBER FOR OLD AT END OF
C---             NAME
C---           - RETURNS WITH NEWNAME (8 CHARACTER NAME WITH NEW
C---                                  PERIOD NUMBER - IF VALID)
C---                          INEWPR (INTEGER EQUIVALENT OF NEW
C---                                  PERIOD NUMBER)
C---                          NBLANK (NUMBER OF BLANKS AT END OF
C---                                  NAME)
C---
      SUBROUTINE RENAME(OLDNAME,NEWNAME,INEWPR,NBLANK)
C---
      CHARACTER*1 OLDNAME(8),OLDPR(2),NEWNAME(8),NEWPR(2),BLANK
      CHARACTER*1 PRNAME(100,2)
      INTEGER INOUT(20)
      COMMON /BLOCK1/NPRIN,NPROUT,INOUT,PRNAME
C---
C---  COMMON BLOCK1 VARIABLES -
C---  (NPRIN - NUMBER OF PERIODS IN INPUT MODEL)
C---  (NPROUT - NUMBER OF PERIODS IN OUTPUT MODEL)
C---  (INOUT - LAST INPUT PERIOD NUMBER FOR EACH CORRESPONDING
C---           OUTPUT PERIOD NUMBER - I.E. AGGREGATION SCHEME)
C---  (PRNAME - TWO CHARACTER EQUIVALENTS FOR EACH POSSIBLE PERIOD
C---            NUMBER - I.E. ('0','0') TO ('9','9') )
C---
C---  INITIALIZE BLANK TO BLANK
C---             NBLANK TO ZERO
C---             INEWPR TO ZERO AND
C---             OLDPR TO ('0','0')
C---
      DATA BLANK/' '/
      NBLANK = 0
      INEWPR = 0
      OLDPR(1) = PRNAME(1,1)
      OLDPR(2) = PRNAME(1,2)
C---
C---  PROCESS OLDNAME CHARACTER BY CHARACTER BEGINNING WITH
C---  LAST CHARACTER AND INITIALLY SET NEWNAME EQUAL TO OLDNAME
C---
      DO 10 I=1,8
         L = 9 - I
         NEWNAME(L) = OLDNAME(L)
C---
C---     COMPUTE NBLANK (NUMBER OF BLANKS ON END OF OLDNAME)
C---     (ASSUME NO BLANKS OCCUR WITHIN THE BODY OF THE NAME)
C---
```

```
          IF (OLDNAME(L).EQ.BLANK) NBLANK = NBLANK + 1
C---
C---      COMPUTE NONBLK (NUMBER OF NONBLANK CHARACTERS PROCESSED
C---                      SO FAR)
C---
          NONBLK = I - NBLANK
C---
C---      PLACE LAST TWO NONBLANK CHARACTERS IN OLDPR
C---
          IF (NONBLK.GT.2.OR.NONBLK.EQ.0) GO TO 10
          INDEX = 3 - NONBLK
          OLDPR(INDEX) = OLDNAME(L)
10    CONTINUE
C---
C---  IF OLDNAME IS ALL BLANKS, RETURN
C---
      IF (NBLANK.EQ.8) RETURN
C---
C---  CONVERT OLDPR TO INTEGER EQUIVALENT BY CALLING SUBROUTINE CONVERT
C---
      CALL CONVERT(OLDPR,IOLDPR)
C---
C---  (IOLDPR - INTEGER EQUIVALENT OF OLDPR
C---           - EQUALS ZERO IF NOT A VALID PERIOD NUMBER)
C---  IF OLDPR WAS NOT A VALID PERIOD NUMBER, RETURN
C---
      IF (IOLDPR.EQ.0) RETURN
C---
C---  COMPARE IOLDPR WITH INOUT (LIST OF ENDING PERIOD NUMBERS)
C---
30    DO 40 I=1,NPROUT
          IF (IOLDPR.GT.INOUT(I)) GO TO 40
C---
C---      I IS NOW OUTPUT PERIOD NUMBER CORRESPONDING TO OLDPR.
C---      SAVE I AS INEWPR
C---      GET CHARACTER EQUIVALENT OF I FROM PRNAME AND
C---      SAVE IN NEWPR
C---      BRANCH TO "CHANGE PERIOD NUMBER"
C---
          INEWPR = I
          NEWPR(1) = PRNAME(I+1,1)
          NEWPR(2) = PRNAME(I+1,2)
          GO TO 50
40    CONTINUE
C---
C---  NO MATCH FOUND. (OLDPR MUST BE INVALID PERIOD NUMBER)
C---  RETURN (INEWPR WILL EQUAL ZERO)
C---
      RETURN
C---
C---  CHANGE PERIOD NUMBER WITHIN NEWNAME TO NEW PERIOD NUMBER
C---  AND RETURN
```

```
C---
50    INDEX = 7 - NBLANK
      NEWNAME(INDEX) = NEWPR(1)
      NEWNAME(INDEX+1) = NEWPR(2)
      RETURN
      END
C---
C---
C---
C---  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C---
C---
C---
C---  SUBROUTINE CONVERT -
C---
C---             INPUT AB (ANY TWO CHARACTER COMBINATION)
C---           - IF BOTH CHARACTERS ARE VALID DIGITS,
C---             CONVERTS AB TO INTEGER EQUIVALENT, CALLED NUMBER
C---           - OTHERWISE, SETS NUMBER = 0
C---           - RETURNS WITH NUMBER
C---
      SUBROUTINE CONVERT(AB,NUMBER)
C---
      CHARACTER*1 AB(2),DIGIT(10)
      INTEGER N(2)
C---
C---  INITIALIZE ARRAY DIGIT TO CHARACTER EQUIVALENTS OF THE 10 DIGITS
C---
      DATA DIGIT(1),DIGIT(2),DIGIT(3),DIGIT(4)/'0','1','2','3'/
      DATA DIGIT(5),DIGIT(6),DIGIT(7),DIGIT(8)/'4','5','6','7'/
      DATA DIGIT(9),DIGIT(10)/'8','9'/
C---
C---  INITIALIZE NUMBER TO ZERO
C---
      NUMBER = 0
C---
C---  PROCESS INPUT CHARACTERS IN TURN (LOOP OVER I)
C---
      DO 20 I=1,2
         N(I) = 0
C---
C---     COMPARE CHARACTER WITH EACH OF TEN DIGITS (LOOP OVER J)
C---
         DO 10 J=1,10
            IF (AB(I).NE.DIGIT(J)) GO TO 10
C---
C---        I-TH CHARACTER EQUALS J-TH DIGIT.
C---        SAVE AS N(I) AND GO ON TO NEXT CHARACTER
C---
            N(I) = J-1
            GO TO 20
10       CONTINUE
```

```
C---
C---     I-TH CHARACTER IS NOT A VALID DIGIT.
C---     RETURN (WITH NUMBER EQUAL TO ZERO)
C---
         RETURN
20    CONTINUE
C---
C---  N HOLDS DIGIT EQUIVALENTS OF AB CHARACTERS.
C---  COMPUTE NUMBER (INTEGER EQUIVALENT OF AB) AND RETURN
C---
      NUMBER = 10*N(1) + N(2)
      RETURN
      END
C---
C---
C---
C---  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C---
C---
C---
C---  SUBROUTINE UPDATE
C---
C---           - INPUT LASTIX (INDEX OF LAST ENTRY IN OUTPUT TABLES)
C---                  ROWNAME (ROW NAME OF ENTRY TO BE ADDED TO TABLES)
C---                  ROWVALU (ASSOCIATED AGGREGATE MPS MATRIX ENTRY)
C---                  PBLANK (NUMBER OF BLANKS AT END OF ROWNAME)
C---           - IF MATCHING ROW NAME IS FOUND IN TABLE NAMETAB THEN
C---             ROWVALU IS ADDED TO CORRESPONDING ENTRY IN VALUTAB
C---           - IF NO MATCH IS FOUND NEW ENTRIES ARE SET UP IN NAMETAB
C---             AND VALUTAB AND LASTIX IS INCREMENTED BY ONE
C---           - RETURNS WITH NEW VALUE OF LASTIX
C---
      SUBROUTINE UPDATE(LASTIX,ROWNAM,ROWVAL,PBLANK,ARITH,GEOM,COUNT,
     1                   MINUS1)
C---
      CHARACTER*1 COLNAME(8),ROWNAME(8),NAMETAB(100,8)
      DIMENSION VALUTAB(100)
      LOGICAL  ARITH,GEOM,MINUS1,TRUE,FALSE
      INTEGER PBLANK
      COMMON/BLOCK2/COLNAME,NAMETAB,VALUTAB,MAXENT
      DATA TRUE,FALSE/.TRUE.,.FALSE./
C---
C---  COMMON BLOCK2 VARIABLES -
C---  (COLNAME - OUTPUT NAME OF AGGREGATED COLUMN)
C---  (NAMETAB - LIST OF AGGREGATED ROW NAMES ENCOUNTERED FOR
C---             THIS COLUMN)
C---  (VALUTAB - CORRESPONDING LIST OF AGGREGATED MPS MATRIX ENTRIES)
C---  (MAXENT - MAXIMUM NUMBER OF ENTRIES IN NAMETAB/VALUTAB
C---          - SHOULD EQUAL  DIMENSION)
C---
C---   IF NO PREVIOUS ENTRIES IN OUTPUT TABLES
C---   BRANCH TO "NEW OUTPUT TABLE ENTRY"
```

```
C---
10     IF (LASTIX.EQ.0) GO TO 50
C---
C---   COMPARE ROWNAME TO EACH ENTRY IN NAMETAB
C---   (CONSIDER ONLY NONBLANK CHARACTERS IN ROWNAME)
C---
       NONBLK = 8 - PBLANK
       DO 30  IX=1,LASTIX
          DO 20  L=1,NONBLK
C---
C---         IF NO MATCH, GO ON TO NEXT NAMETAB ENTRY
C---
             IF (ROWNAME(L).NE.NAMETAB(IX,L)) GO TO 30
20        CONTINUE
C---
C---      MATCHING NAME FOUND IN NAMETAB.
C---      ADD ROWVALU TO CORRESPONDING ENTRY IN VALUTAB
C---      RESET ROWVALU TO ZERO AND RETURN
C---
         IF (ARITH) GO TO 22
         IF (MINUS1) GO TO 24
         VALUTAB(IX)=VALUTAB(IX)+COUNT*(ROWVALU**(1/COUNT))
23       COUNT=1
         GO TO 25
24       VALUTAB(IX)=VALUTAB(IX)-(COUNT*(ROWVALU**(1/COUNT)))
         GO TO 23
22        VALUTAB(IX)=VALUTAB(IX)+ROWVALU
25        ROWVALU = 0.
          RETURN
30    CONTINUE
C---
C---  NO MATCHING ROW NAME IN NAMETAB. (I.E. ROWNAME HAS NOT
C---   BEEN ENCOUNTERED BEFORE FOR THIS (OUTPUT) COLUMN NAME)
C---
C---  IF TABLES ARE NOT FULL,BRANCH TO "NEW OUTPUT TABLE ENTRY"
C---
40    IF (LASTIX.LT.MAXENT) GO TO 50
C---
C---  OUTPUT TABLES ARE FULL.
C---  PRINT WARNING AND RETURN
C---
      WRITE (6,952) MAXENT,COLNAME
952   FORMAT(1H ,' ** NAMETAB/VALUTAB DIMENSION,',I3,
     1            ', EXCEEDED FOR COLUMN ',8A1)
C---
C---  (THE NUMBER OF SUCH WARNING MESSAGES WILL INDICATE THE
C---   EXTENT OF REDIMENSIONING REQUIRED - DON'T FORGET TO
C---   REDIMENSION IN SUBROUTINE COLOUT)
C---
      RETURN
C---
C---  NEW OUTPUT TABLE ENTRY.
```

```
C---  INCREMENT LASTIX (INDEX OF LAST ENTRY)
C---  INSERT ROWNAME IN NAMETAB
C---  INSERT ROWVALU IN VALUTAB
C---  RESET ROWVALU (FOR SAFETY)
C---
50    LASTIX = LASTIX + 1
      DO 60  L=1,8
         NAMETAB(LASTIX,L) = ROWNAME(L)
60    CONTINUE
         IF (ARITH) GO TO 70
         IF (MINUS1) GO TO 65
         VALUTAB(LASTIX)=COUNT*(ROWVALU**(1/COUNT))
66       COUNT=1
         GO TO 80
65       VALUTAB(LASTIX)=(-1)*(COUNT*(ROWVALU**(1/COUNT)))
         GO TO 66
70    VALUTAB(LASTIX) = ROWVALU
80    ROWVALU = 0.
         IF (GEOM) GO TO 81
      ARITH=TRUE
      GO TO 82
81    ARITH=FALSE
82    RETURN
      END
C---
C---
C---
C---  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C---
C---
C---
C---  SUBROUTINE COLOUT
C---
C---           - INPUT LASTIX (INDEX OF LAST ENTRY IN OUTPUT TABLES)
C---                  (ASSUME VALID INDEX)
C---           - PROCESSES OUTPUT TABLES IN COMMON BLOCK2 TWO ENTRIES
C---             AT A TIME, SUBMITTING THEM TO SUBROUTINE CARDOUT
C---             FOR OUTPUT
C---
      SUBROUTINE COLOUT(LASTIX)
C---
      CHARACTER*1 TYPE(4),COLNAME(8),NAMETAB(100,8),NAME1(8),NAME2(8)
      DIMENSION VALUTAB(100)
      COMMON/BLOCK2/COLNAME,NAMETAB,VALUTAB,MAXENT
C---
C---  COMMON BLOCK2 VARIABLES -
C---  (COLNAME - OUTPUT NAME OF AGGREGATED COLUMN)
C---  (NAMETAB - LIST OF AGGREGATED ROW NAMES ENCOUNTERED FOR
C---             THIS COLUMN)
C---  (VALUTAB - CORRESPONDING LIST OF AGGREGATED MPS MATRIX ENTRIES)
C---  (MAXENT - MAXIMUM NUMBER ENTRIES IN NAMETAB/VALUTAB )
C---          - SHOULD EQUAL DIMENSION )
```

```
C---
C---  INITIALIZE TYPE TO BLANKS
C---
      DATA TYPE(1),TYPE(2),TYPE(3),TYPE(4)/' ',' ',' ',' '/
C---
C---  COMPUTE LEVEN (LARGEST EVEN NUMBER LESS OR EQUAL TO LASTIX) AND
C---          NEVEN (NUMBER OF EVEN NUMBERS LESS OR EQUAL TO LASTIX)
C---
      NEVEN = LASTIX/2
      LEVEN = NEVEN*2
C---
C---  LOOP ONCE FOR EVERY TWO ENTRIES IN OUTPUT TABLES (LOOP OVER I)
C---
      IF (NEVEN.EQ.0) GO TO 25
      DO 20 I=1,NEVEN
         INDEX = 2*I - 1
C---
C---     MOVE NAMES FROM NAMETAB INTO NAME1 AND NAME2
C---     (CALL STATEMENT WILL NOT ACCEPT AN IMPLIED DO LOOP)
C---
         DO 10 J=1,8
            NAME1(J) = NAMETAB(INDEX,J)
            NAME2(J) = NAMETAB(INDEX+1,J)
10       CONTINUE
C---
C---     OUTPUT ONE COLUMN CARD BY CALLING SUBROUTINE CARDOUT
C---
         CALL CARDOUT(TYPE,COLNAME,2,NAME1,
     1     VALUTAB(INDEX),NAME2,VALUTAB(INDEX+1))
20    CONTINUE
C---
C---  IF LASTIX ODD, OUTPUT ONE MORE COLUMN CARD AND RETURN
C---
25    IF (LASTIX.EQ.LEVEN) RETURN
      DO 30 J=1,8
         NAME1(J) = NAMETAB(LASTIX,J)
         NAME2(J) = NAME1(J)
30    CONTINUE
      CALL CARDOUT(TYPE,COLNAME,1,NAME2,
     1 VALUTAB(LASTIX),NAME2,VALUTAB(LASTIX))
C---
C---  (3-RD ARGUMENT IN CALL IS NUMBER OF ROW ENTRIES SUBMITTED FOR
C---   OUTPUT - IN THIS CASE ONLY ONE SO 6-TH AND 7-TH ARGUMENTS
C---   WILL BE IGNORED)
C---
      RETURN
      END
C---
C---
C---
C---  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C---
```

```
C---  SUBROUTINE CARDOUT
C---
C---           - INPUT TYPE (4 CHARACTER (BOUND) TYPE)
C---                   COLNAME (8 CHAR. COLUMN/RHS/BOUND NAME)
C---                   NENTRY (NUMBER OF ENTRIES SUBMITTED
C---                           FOR OUTPUT - ASSUME 1 OR 2 )
C---                   NAME1 (8 CHAR. NAME OF FIRST ENTRY)
C---                   VALUE1 (FIRST ENTRY VALUE)
C---                   NAME2 (8 CHAR. NAME OF SECOND ENTRY)
C---                   VALUE2 (SECOND ENTRY VALUE)
C---           - PRINTS WARNING IF ANY VALUE ENTRY IS ZERO
C---             (UNLESS THIS IS A BOUND CARD)
C---           - COUNTS NUMBER OF DIGITS TO LEFT OF DECIMAL
C---             FOR EACH VALUE ENTRY
C---           - SELECTS APPROPRIATE FORMAT STATEMENT
C---           - WRITES OUT ONE CARD AND RETURNS
C---
      SUBROUTINE CARDOUT(TYPE,COLNAME,NENTRY,NAME1,VALUE1,NAME2,
     1                     VALUE2)
C---
      CHARACTER*1 TYPE(4),COLNAME(8),NAME1(8),NAME2(8),BLANK
C---
C---  INITIALIZE BLANK
C---  THEN, IF INPUT TYPE FIELD NOT BLANK (IE. A BOUND CARD),
C---       BYPASS TEST FOR ZERO ENTRY
C---
      DATA BLANK/' '/
      IF (TYPE(2).NE.BLANK) GO TO 5
C---
C---  INITIALIZE EPSILON (TOLERANCE FOR ZERO)
C---
      EPSILON = 0.000001
C---
C---  IF A SUBMITTED ENTRY IS WITHIN EPSILON OF ZERO, PRINT WARNING
C---
      IF (ABS(VALUE1).LT.EPSILON)
     1   WRITE (6,954) COLNAME,NAME1,VALUE1
      IF (ABS(VALUE2).LT.EPSILON.AND.NENTRY.NE.1)
     1   WRITE (6,954) COLNAME,NAME2,VALUE2
954   FORMAT (1H ,'COLUMN NAME ',8A1,' ROW NAME ',8A1,' VALUE ',
     1        F12.6,' ** ZERO ENTRY')
C---
C---  BRANCH ON ABSOLUTE VALUE OF FIRST ENTRY
C---  (LARGEST NUMBER ANTICIPATED IS 9,999,999.999)
C---
5     IF (ABS(VALUE1).LT.10000.) GO TO 10
      IF (ABS(VALUE1).LT.100000.) GO TO 70
      IF (ABS(VALUE1).LT.1000000.) GO TO 130
      GO TO 190
C---
C---
C---
```

```
C---
C FIRST ENTRY IS WITHIN 10,000. OF ZERO.
C---
C---  IF ONLY ENTRY, OUTPUT AND RETURN
C---
10    IF (NENTRY.NE.1) GO TO 20
      WRITE (9,931) TYPE,COLNAME,NAME1,VALUE1
      RETURN
C---
C---  BRANCH ON ABSOLUTE VALUE OF SECOND ENTRY
C---
20    IF (ABS(VALUE2).LT.10000.) GO TO 30
      IF (ABS(VALUE2).LT.100000.) GO TO 40
      IF (ABS(VALUE2).LT.1000000.) GO TO 50
      GO TO 60
C---
C---  SECOND ENTRY IS WITHIN 10,000 OF ZERO.
C---  OUTPUT AND RETURN
C---
30    WRITE (9,931) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS BETWEEN 10,000. AND 100,000.
C---  OUTPUT AND RETURN
C---
40    WRITE (9,932) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS BETWEEN 100,000 AND 1,000,000.
C---  OUTPUT AND RETURN
C---
50    WRITE (9,933) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS GREATER OR EQUAL 1,000,000.
C---  OUTPUT AND RETURN
C---
60    WRITE (9,934) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---
C---
C---
C FIRST ENTRY IS BETWEEN 10,000 AND 100,000.
C---
C---  IF ONLY ENTRY, OUTPUT AND RETURN
C---
70    IF (NENTRY.NE.1) GO TO 80
      WRITE (9,935) TYPE,COLNAME,NAME1,VALUE1
      RETURN
C---
C---  BRANCH ON ABSOLUTE VALUE OF SECOND ENTRY
```

```
C---
80    IF (ABS(VALUE2).LT.10000.) GO TO 90
      IF (ABS(VALUE2).LT.100000.) GO TO 100
      IF (ABS(VALUE2).LT.1000000.) GO TO 110
      GO TO 120
C---
C---  SECOND ENTRY IS WITHIN 10,000 OF ZERO.
C---  OUTPUT AND RETURN
C---
90    WRITE (9,935) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS BETWEEN 10,000. AND 100,000.
C---  OUTPUT AND RETURN
C---
100   WRITE (9,936) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS BETWEEN 100,000 AND 1,000,000.
C---  OUTPUT AND RETURN
C---
110   WRITE (9,937) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS GREATER OR EQUAL 1,000,000.
C---  OUTPUT AND RETURN
C---
120   WRITE (9,938) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---
C---
C---
C FIRST ENTRY IS BETWEEN 100,000 AND 1,000,000.
C---
C---  IF ONLY ENTRY, OUTPUT AND RETURN
C---
130   IF (NENTRY.NE.1) GO TO 140
      WRITE (9,939) TYPE,COLNAME,NAME1,VALUE1
      RETURN
C---
C---  BRANCH ON ABSOLUTE VALUE OF SECOND ENTRY
C---
140   IF (ABS(VALUE2).LT.10000.) GO TO 150
      IF (ABS(VALUE2).LT.100000.) GO TO 160
      IF (ABS(VALUE2).LT.1000000.) GO TO 170
      GO TO 180
C---
C---  SECOND ENTRY IS WITHIN 10,000 OF ZERO.
C---  OUTPUT AND RETURN
C---
150   WRITE (9,939) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
```

```
      RETURN
C---
C---  SECOND ENTRY IS BETWEEN 10,000. AND 100,000.
C---  OUTPUT AND RETURN
C---
160   WRITE (9,940) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS BETWEEN 100,000 AND 1,000,000.
C---  OUTPUT AND RETURN
C---
170   WRITE (9,941) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS GREATER OR EQUAL 1,000,000.
C---  OUTPUT AND RETURN
C---
180   WRITE (9,942) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---
C---
C---
C FIRST ENTRY IS GREATER OR EQUAL 1,000,000.
C---
C---  IF ONLY ENTRY, OUTPUT AND RETURN
C---
190   IF (NENTRY.NE.1) GO TO 200
      WRITE (9,944) TYPE,COLNAME,NAME1,VALUE1
      RETURN
C---
C---  BRANCH ON ABSOLUTE VALUE OF SECOND ENTRY
C---
200   IF (ABS(VALUE2).LT.10000.) GO TO 210
      IF (ABS(VALUE2).LT.100000.) GO TO 220
      IF (ABS(VALUE2).LT.1000000.) GO TO 230
      GO TO 240
C---
C---  SECOND ENTRY IS WITHIN 10,000 OF ZERO.
C---  OUTPUT AND RETURN
C---
210   WRITE (9,944) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS BETWEEN 10,000. AND 100,000.
C---  OUTPUT AND RETURN
C---
220   WRITE (9,945) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS BETWEEN 100,000 AND 1,000,000.
C---  OUTPUT AND RETURN
```

```
C---
230   WRITE (9,946) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  SECOND ENTRY IS GREATER OR EQUAL 1,000,000.
C---  OUTPUT AND RETURN
C---
240   WRITE (9,946) TYPE,COLNAME,NAME1,VALUE1,NAME2,VALUE2
      RETURN
C---
C---  FORMAT STATEMENTS
C---
931   FORMAT (4A1,8A1,2X,8A1,2X,F12.6,3X,8A1,2X,F12.6)
932   FORMAT (4A1,8A1,2X,8A1,2X,F12.6,3X,8A1,2X,F12.5)
933   FORMAT (4A1,8A1,2X,8A1,2X,F12.6,3X,8A1,2X,F12.4)
934   FORMAT (4A1,8A1,2X,8A1,2X,F12.6,3X,8A1,2X,F12.3)
935   FORMAT (4A1,8A1,2X,8A1,2X,F12.5,3X,8A1,2X,F12.6)
936   FORMAT (4A1,8A1,2X,8A1,2X,F12.5,3X,8A1,2X,F12.5)
937   FORMAT (4A1,8A1,2X,8A1,2X,F12.5,3X,8A1,2X,F12.4)
938   FORMAT (4A1,8A1,2X,8A1,2X,F12.5,3X,8A1,2X,F12.3)
939   FORMAT (4A1,8A1,2X,8A1,2X,F12.4,3X,8A1,2X,F12.6)
940   FORMAT (4A1,8A1,2X,8A1,2X,F12.4,3X,8A1,2X,F12.5)
941   FORMAT (4A1,8A1,2X,8A1,2X,F12.4,3X,8A1,2X,F12.4)
942   FORMAT (4A1,8A1,2X,8A1,2X,F12.4,3X,8A1,2X,F12.3)
943   FORMAT (4A1,8A1,2X,8A1,2X,F12.3,3X,8A1,2X,F12.6)
944   FORMAT (4A1,8A1,2X,8A1,2X,F12.3,3X,8A1,2X,F12.5)
945   FORMAT (4A1,8A1,2X,8A1,2X,F12.3,3X,8A1,2X,F12.4)
946   FORMAT (4A1,8A1,2X,8A1,2X,F12.3,3X,8A1,2X,F12.3)
      END
C---
C---
C---
C---
C---
C---
$DATA
..1..2..3..4..5..6..7..8..9.10.11.12.13.14.15.16.17.18.19.20
  1  2  3  1  1
$STOP
/*
/*
```

DATE

ILME