NBSIR 75-911

A National Network Data Base System

Richard H. F. Jackson

Applied Mathematics Division
Institute for Basic Standards
Washington, D. C. 20234

September 1975

Technical Report to:

Office of the Secretary
U.S. Department of Transportation

Bureau of Economic Analysis
U.S. Department of Commerce

Office of the Chief of Engineers
U.S. Department of the Army

U S. DEPARTMENT OF COMMERCE

NATIONAL BUREAU OF STANDARDS

NBSIR 75-911

A NATIONAL NETWORK DATA BASE SYSTEM

Richard H. F. Jackson

Applied Mathematics Division
Institute for Basic Standards
Washington, D. C. 20234

September 1975

Technical Report to:

Office of the Secretary
U.S. Department of Transportation

Bureau of Economic Analysis
U.S. Department of Commerce

Office of the Chief of Engineers
U.S. Department of the Army

ABSTRACT

This report documents a data base "system" created at the National Bureau of Standards. The National Network Data Base System (NNDBS) provides information on intercity flows of freight and passengers throughout the United States. It consists of a set of FORTRAN programs written for the NBS computer (but transportable) and some basic data tapes. In addition to providing basic data on the modal transportation networks, and interregional flows of passengers and freight, the NNDBS can make estimates of market splits among modes, can perform aggregations over certain "zones" in the U.S., and is capable of easy extension to other uses. This report is intended as a user's guide and includes discussions of the data tapes and each of the programs. Complete listings and tape formats are also included.

Keywords: Commodity flows, data base, freight mode choice, market split, modal choice, modal freight flows, national transport analyses, regional transport analysis, transportation network

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This report documents the National Network Data Base System (NNDBS) from a programmer/user point of view. The NNDBS was created at the National Bureau of Standards in order to provide information on flows of freight and passengers on the major intercity transportation network of the United States. The reasons for developing such a data base, the rationale used in producing the basic data, and the checks used to assure validity, are all given in [4]. The present report aims to provide documentation sufficient to permit ready use of the Data Base system by any competent programmer. It could, therefore, be viewed as a user's guide to the NNDBS.

The NNDBS is properly called a "system" because some of the data available from it are not stored explicitly, but can be obtained by applying one or more of its programs to one of its basic data tapes. The alternative to such a "systematized data base" would be actual storage of all the various manifestations of the basic data tapes. This would result in at least 12 separate and nearly-full reels of magnetic tape. It was felt that the creation and use of so large a conventional data base would involve too great a risk of crippling physical tape problems such as unrecoverable parity errors which could completely destroy the usefulness of a given data base tape. Although reliability for such a conventional data base can be obtained through maintenance of complete listings and backup tapes, the required costs, not only in terms of data-storage, but also in personnel time to maintain the massive structure, appeared too high to justify such an approach. It was therefore decided to create a data base "system" in the sense described above. This approach would keep at a minimum the number of tapes stored, thus keeping at a minimum the number of backup tapes and complete listings. The programs are kept on mass storage files.

A pictorial representation of the NNDBS is given in Figure 1. From that representation, it can quickly be seen that all tapes are derivable from the single tape called NET70, so that just NET70 and a number of programs could be kept as "the system". However, it was decided to keep three tapes, NET70, NET80, and NET90 as the basic data tapes in the system for a number of reasons. These include the fact that the program FRATAR (see p. 223 of [5]), which produces NET80 and NET90 from NET70 by a technique known as Fratar projection (see [2]), was written for the IBM 360, and that it need only be run once for each tape (NET80 and NET90) in question. The cost of converting FRATAR to be acceptable to the NBS UNIVAC 1108, and of updating it for use in the NNDBS was deemed higher than the extra two-tape-maintenance-costs incurred by having the three basic data tapes.

Before moving on to a discussion of the data on the tapes and to a presentation of the programs, we note that there are two other tape files which do not appear in Figure 1: NETTS and NETPS. NETTS is a test tape which contains an "alternative network" to the one represented in the other data base tapes. More on NETTS can be found in the

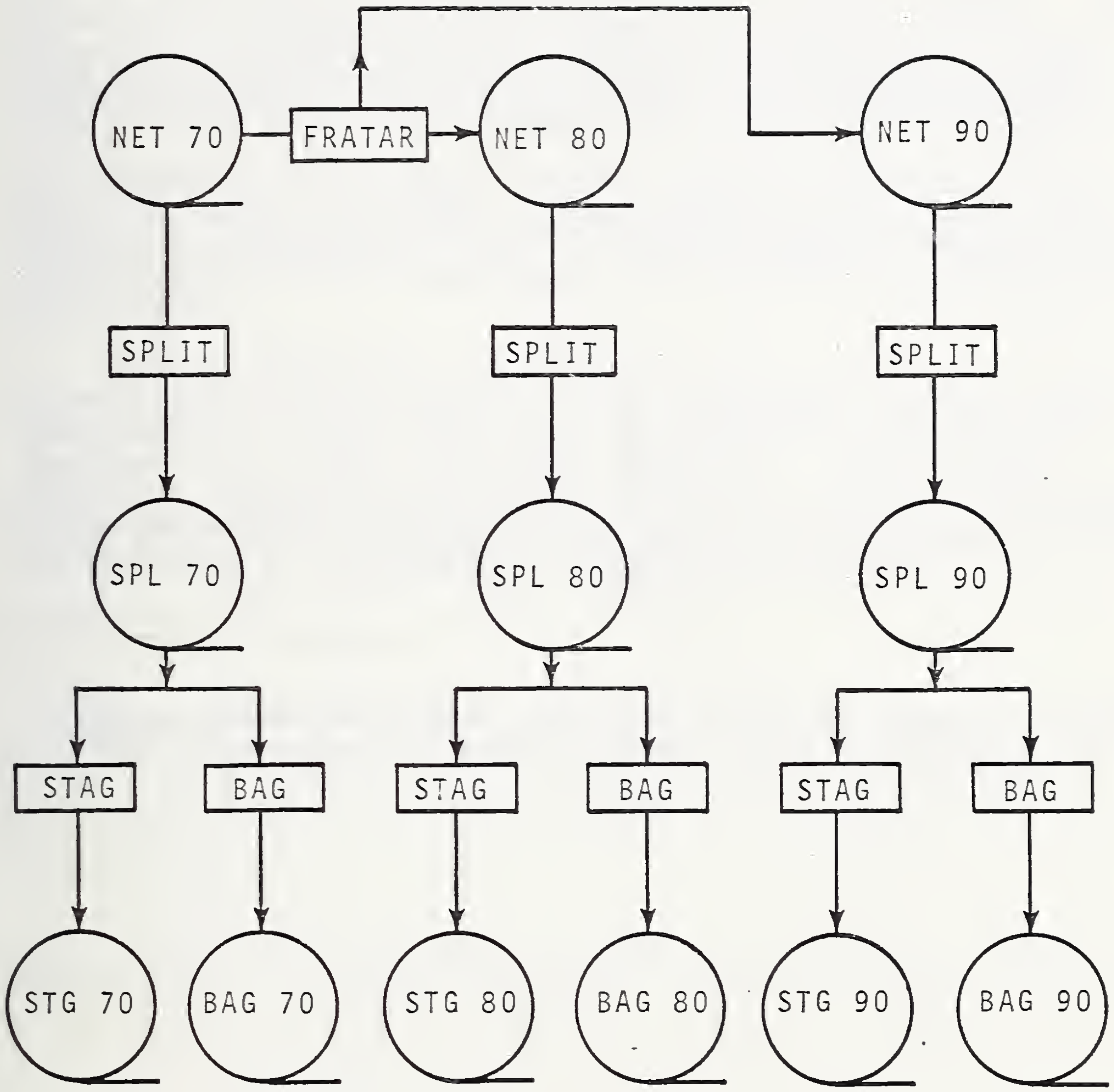

FIGURE 1: THE TAPES AND PROGRAMS OF THE NATIONAL NETWORK DATA BASE SYSTEM

discussion of the program SPLIT, and its format is given in Appendix C. NETPS contains information on intercity passenger movements throughout the national transportation system. Nothing more than storage is currently planned for NETPS, so that nothing more about it will be presented in the present document except for its computer format, given in Appendix B.

The programs in the NNDBS were written in FORTRAN for the NBS UNIVAC 1108,* but none are machine dependent. Some will, however, require modification before being transported to another computer. The modifications required are all related to block transfer of data and occur only in the driver programs discussed in Sections 3, 4, and 5. Some form of binary block transfer of data should be substituted for each reference to the utility package NTRAN found in those programs.

Section 2 of this report presents a detailed discussion of the basic data tapes in the NNDBS, while Sections 3, 4, and 5 take up the programs used to move from level to level in Figure 1. There are six Appendices giving tape formats, and three more containing listings of the programs.

To be most effective, reading of Sections 2, 3, 4, and 5 should be accompanied by regular reference to the formats and listings given in the Appendices. A word here on the organization of the listings in Appendices G, H, and I, is therefore appropriate. Each of those three appendices contains three separate listings: the first is of the control stream required for a run; the second, of the driver program in the run; and the third, of the respective "workhorse" subroutine.

---

*Although program development took place on the NBS computer (a UNIVAC 1108), no endorsement of UNIVAC or UNIVAC equipment is intended.

2. THE BASIC DATA TAPES--NET70, NET80, AND NET90

This section discusses the basic data tapes NET70, NET80, and NET90. Since NET80 and NET90 are directly derivable from NET70 and their computer format is exactly the same, the discussion will center around NET70. Differences between the 1970 tape (NET70) and the 1980 and 1990 tapes (NET80 and NET90, respectively) are noted at the end of the section.

NET70 contains information on freight movements throughout the U. S. and its possessions for the year 1970. The information is stored as flows among certain "zones" in the U. S. These zones, called National Network Simulation (NNS) zones, were mapped out previous to this project. (For more on NNS zones, see [3, 4].) There are 512 such NNS zones, and each record on NET70 contains data on one-way freight movements between a particular pair of zones. All the data stored on NET70 falls into two categories: amount of goods shipped, and the impedances (distance and time) involved in shipping it.

The total flow of freight from one zone to another has been broken down into eight commodity classes, measured in tons:

1) large, high value shipments,
2) small, high value shipments,
3) large, medium value shipments,
4) small, medium value shipments,
5) large, low value shipments,
6) small, low value shipments,
7) large, crude oil shipments,
8) large, refined oil shipments.

Therefore, a tape record contains an entry for each of the above commodity classes, representing the average daily flow of goods*, from the origin NNS zone to the destination NNS zone, of that commodity class and size. The identification numbers of the origin and destination NNS zones also appear in the record.

The interzonal impedances are associated with the various modes of transportation. The NNDBS deals with six possible modes of shipping a commodity from one zone to another:

1) truck,
2) rail,
3) air,
4) water,
5) crude oil pipeline,
6) refined oil pipeline.

* The flow of goods is measured in tons.

Each tape record contains an entry for each of the above six modes, which entry represents the distance in statute miles by each mode between the zone pair for that record. The record also contains time entries, namely the times (in minutes) required to traverse each of the above distances via the associated mode. Each record on NET70, therefore, contains 22 entries: the origin-destination NNS zone numbers, 6 distance entries, 6 time entries, and 8 commodity-class quantity entries.

The tape contains one record for every zonal pair that is connected and has a nonzero flow. When the data tape was first created, the NNS zones were considered to be completely connected--a link was presumed to exist between each possible pairing. However, it was soon discovered that many such links had zero flow for all commodity classes. A culling process was then performed to remove these "zero" records from the tape.* This cut the number of records on the tape file from 512*512=262,144 to only 68,227, permitting the complete file to reside on one tape, and considerably reducing the time and effort involved for data manipulations.

Some time and distance entries on the tape have been set to a "flag value" of 99999. This means that the connection via the associated transportation mode for that particular zone pair either does not exist or is so expensive to use as not to be a serious consideration. It is important to bear in mind that one flagged value does not completely remove a link from consideration in the "zero flow" sense described earlier. There are six modes of transportation available between pairs. So long as even one of these is not flagged, the link is still a valid connection.

NET80 and NET90 are copies of NET70, but with all flows expanded by Fratar projection so that they are predictions of 1980 and 1990 movement levels. The time and distance entries are the same on all three tapes, the only differences being in the commodity class flow entries.

Appendix A contains a succinct description of the computer format of the three tapes discussed in this section.

---

*This step is not irreversible. Zones could be reconnected, with a positive flow, by use of a simple edit program.

3. THE MODAL-SPLIT PROGRAMS--SPLIT

SPLIT is a subroutine whose inputs are the records from NET70 and whose outputs are records in which the commodity flows have been allocated among the available modes of transportation. In other words, whereas NET70 contains what must be shipped from zone to zone, SPL70 (the output tape from a modal-split run) shows how much of each commodity will be shipped by each mode.

The basic idea in a modal split is to compute a set of "weights" u_{ij}, i=1,2,...,8, and j=1,2,...,6, satisfying

$$\sum_{j=1}^{6} u_{ij} = 1, \qquad \text{for all } i.$$

These weights can then be used to distribute commodity i among the possible modes j. The model used to compute the weights in subroutine SPLIT is given by

$$\omega_{ij} = e^{\rho_{ij}} \left[c_{ij}\,(d_j)\right]^{\beta_{ij}} t_j^{\;\alpha_{ij}},$$

where d_j and t_j are the distance and time associated with transportation mode j, $c_{ij}(.)$ is a cost function dependent on distance, and α, β, and ρ are parameters, and e is the base of the natural logarithms. The weights are obtained by normalizing the ω_{ij}:

$$u_{ij} = \omega_{ij} \Big/ \sum_{k=1}^{6} \omega_{ik}, \qquad \text{for all } i.$$

More on the general concept of modal splitting, this particular choice of a model and the calibrations performed to choose appropriate values of α, β, and ρ can be found in [4]. This brief account is included here only to provide the programmer/user with background information on the theoretical foundation for subroutine SPLIT.

The key to understanding the inner workings of SPLIT, is to recall that the NNDBS involves 8 commodity classes and 6 modes of transportation, and to note that not all modes are available to all commodities. For example, the only commodity that can be transported via the crude oil pipeline is, of course, crude oil. There are other combinations that are theoretically possible, but in practice are not used because cost is prohibitive; an example is sending crude oil by air. The complete picture of possible connections is given in Figure 2, where a "0" indicates a usage which is impossible (in the sense outlined above) and a "1" indicates a valid connection.

	Transportation Modes					
Commodity Classes	Truck	Rail	Air	Water	Crude Pipeline	Refined Pipeline
High large (*)	1	1	0	1	0	0
High small	1	1	1	0	0	0
Medium large	1	1	0	1	0	0
Medium small	1	1	0	0	0	0
Low large	1	1	0	1	0	0
Low small	1	1	0	0	0	0
Crude large	0	0	0	1	1	0
Refined large	1	1	0	1	0	1

*) "High" (or "medium" or "low") refers to dollar-value per unit for the
ommodity class; "large" or "small" refers to shipment size.

FIGURE 2: THE MATRIX OF POSSIBLE CONNECTIONS

To facilitate program organization, it was decided to store ω_{ij} as a one-dimensional "packed" list (OMEGA) corresponding to the matrix given in figure 2, but with all elements whose value is zero removed from consideration. Since the same storage scheme was applied to α_{ij}, β_{ij} and ρ_{ij}, there was no problem in matching the correct values of those arrays with ω_{ij} for a modal-split computation. However, with this storage scheme, the simple row-column indexing that would yield commodity class and transportation mode is not available. That information is needed to sum across modes for a given commodity, and to retrieve correct distances for flag-value checking.

The row indexing required in SPLIT is provided by the array COMIND (see the listing in Appendix G). For any I=1,22, COMIND (I) is the commodity class reference (row number in Figure 2) for the corresponding element I of the arrays OMEGA, ALPHA, BETA, and RHO.

It turns out that transportation-mode indexing (column number in Figure 2) is not required in SPLIT. What is required is retrieval, for each element of OMEGA, of the distance to be traveled by the commodity quantity in question. This retrieval information is provided by the array DSTIND (again, see the listing in Appendix G). For I=1,22, DSTIND (I) is an index into the input record (array IR) where the desired distance can be found.

The last indexing array to be explained is the one represented by CSTIND, which is an indexing scheme for the cost functions. For each of the pipeline refined, pipeline crude, water and air modes of transportation, there is a cost function to be applied to all commodities shipped by that mode. Rail and truck, however, have three cost functions, the applicable one depending on whether the shipment is high value, medium value, or low value, according to a prescribed classification scheme. This yields a total of 10 cost functions, and these are stored in the array COST. Therefore CSTIND (I), for each I=1,22, is an index into COST for retrieving the correct cost function to be used in the modal-split computation.

It must be expected that cost functions will change over time. Moreover, their number can also change (e.g., a disaggregation of the "water" mode might be used for some studies). Given the indexing scheme for the cost functions outlined above, implementation of any such changes would be simple and straightforward.

The actual work of "splitting" the flows is now straightforward. SPLIT works on one record at a time from NET70, NET80 or NET90. First it computes the costs to be stored in COST, which are based on the interzonal travel distances by each mode. It then launches into the computation of the weights ω_{ij}, relying on the indexing presented above. Having done so, it splits each commodity-class flow among the available modes by multiplying by the weights. An output record is then created following the computer format given in Appendix D, and that record is passed back to the main program, along with control.

The main driving program which controls the modal-split run is concerned with reading in from the input network tape (NET70, NET80, NET90), mounted on logical unit 7, a block of 100 records, and passing each record off to subroutine SPLIT. After processing is complete for the block of 100, the driver writes the "splitted" version of the records onto the output SPLIT tape (SPL70, SPL80, or SPL90), mounted on logical unit 8, and goes back to read in the next block of 100 records. The data blocks are transferred by a UNIVAC 1108 software package (NTRAN) which performs binary transfers of blocks of any size.

The printer output from a modal split run consists of a table giving accumulations by ton, ton-mile, and revenues, plus a second table giving sums over all fields of the output tape. Both tables are useful for checking purposes.

There is another capability that has been provided in the driver program for SPLIT. It is an option to read an alternate set of impedances from a tape mounted on logical unit 9. This alternate tape could be NETTS (see Appendix C), or another tape with similar format. This capability is useful for testing and experimentation purposes. It could, for example, be used in discovering the effect of adding and deleting links from the network. To make use of this capability one simply creates the "alternate impedance tape", mounts it on unit 9 along with the normal input and output network tapes on units 7 and 8, and changes the value of the logical variable ALT in the driver program from .FALSE. to .TRUE.

Note that it is not necessary to carry the complete set of impedances on the alternate impedance tape. One may desire to change only the highway impedances or only those for rail, or some combination. The point is that the alternate tape need only carry information that is changed. Program changes, to accommodate such partial modifications of the impedances, are required only in lines 19 and 20 of the driver program for SPLIT. For example, suppose it is desired to change the rail impedances, but keep all others fixed. One then creates an alternate tape with 68,227 logical records, each one containing four words: the origin NNS zone, the destination NNS zone, the (new) rail distance, and the (new) rail time. Lines 19 and 20 of the driver program would then be changed from the existing DO loop to the following:

```
IR(5,J) = AR(3,J)
IR(6,J) = AR(4,J)
```

4. THE STATE AGGREGATION PROGRAMS--STAG

STAG is a subroutine whose inputs are records from one of the "split tapes" (SPL70, SPL80, or SPL90), and whose outputs are a set of matrices showing: a) the average daily flow of goods by commodity-class among the 50 states, the District of Columbia, and the U. S. possessions; and b) the costs of transporting that amount of goods.

Just as with SPLIT, STAG is the major routine which does the aggregation on individual logical records that have been passed to it by the driver routine. The driver does all the tape handling and the breakdown of the blocks to logical records. Since STAG's driver is similar to SPLIT's, discussed in Section 3, the rest of the discussion in this section will be confined to subroutine STAG.

The key to understanding STAG lies in Figure 3, which shows which NNS zones are mapped into which states. For example, the top line indicates that all flows originating from NNS zones 1 through 5 should be aggregated to represent a total flow originating from Maine. The meaning of the "partial zones" column is straightforward: the specified percentages of the flows for the given zones are to be aggregated to make up the flow to or from the associated state. The need for this column arises because some NNS zones cross state boundaries. The percentages are based on total population figures and were obtained from Allan Olson of the Bureau of Economic Analysis, U.S. Dept. of Commerce.

The manner in which the mapping scheme is implemented in STAG, however, is the inverse of the mapping described in Figure 3. That is, for a given I ranging from 1 to 512 (the NNS zone numbers that appear on the NNDBS tapes), NDX(I) is the state number to which the flows from NNS zone I should be added. If NDX(I) is negative, ABS(NDX(I)) is a pointer to a row in the arrays PCT and NPCT which respectively contain the percentages and the states to which the percentage flows are to be added.

There are 10 matrices that are created by STAG. They represent, respectively, the volumes of flow and the cost of transporting that flow, among the states for the following situations:

1.) high value shipments, both large and small,
2.) medium value shipments, both large and small,
3.) low value shipments, both large and small,
4.) all crude oil shipments, and
5.) all refined oil shipments.

As each record is passed into STAG by the driver, the amount to be added to each one of the 10 matrices is computed in turn and control is

STATE NUMBER	STATE	WHOLE NNS ZONES	PARTIAL NNS ZONES
1	Maine	1-5 incl	-
2	New Hamp.	6-9 incl	-
3	Vermont	10-11 incl	-
4	Mass.	12-26 incl	-
5	R.I.	27	25% of 28
6	Conn.	29-40 incl	75% of 28
7	N.Y.	41-50 incl; 52-58 incl	-
8	N.J.	59-65 incl	20% of 66; 21% of 69; 12% of 84
9	Penn.	51, 67, 68, 70-83 incl	80% of 66; 79% of 69
10	Del.	85	77% of 84
11	Md.	86-89 incl	11% of 84; 43% of 90
12	D.C.		28% of 90
13	Mich.	149-167 incl	16% of 113
14	Ohio	91-97 incl; 100-105 incl; 107-112 incl;114	58% of 98; 81% of 106; 84% of 113; 44% of 99; 22% of 200
15	Indiana	115-123 incl; 125-130 incl	3% of 106; 82% of 124; 16% of 205
16	Illinois	131-134 incl; 136-146 incl; 148	54% of 135; 23% of 323
17	Wisconsin	168-181 incl	78% of 296
18	Minn.	295, 297-301 incl	22% of 296; 39% of 323
19	Iowa	302, 304-314 incl	46% of 135; 89% of 303; 16% of 343
20	Missouri	315-317 incl; 319-322 incl	65% of 318; 77% of 323
21	N. Dak.	324-327 incl	61% of 328
22	S. Dak.	329-333 incl	-
23	Nebraska	334-342 incl	11% of 303; 84% of 343
24	Kansas	344-352 incl	35% of 318
25	Virginia	182-194 incl	29% of 90
26	W. Va.	195-199 incl	42% of 98; 56% of 99; 57% of 200
27	Ky.	201-204 incl; 206	16% of 106; 18% of 124; 21% of 200; 84% of 205
28	Tennessee	207; 209-211 incl; 213-215 incl	94% of 208; 83% of 212
29	N. Car.	216-229 incl	-
30	S. Car.	230-235 incl	36% of 239
31	Georgia	241;236-238 incl;240;242-244 incl;246-247 incl;	17% of 212; 64% of 239; 81% of 245
32	Florida	248-260 incl	-
33	Alabama	261-271 incl	19% of 245
34	Miss.	272-275 incl	-
35	Louisiana	276-285 incl	-
36	Arkansas	286-291 incl; 293-294 incl	6% of 208; 65% of 292; 40% of 362
37	Oklahoma	353-361 incl	35% of 292
38	Texas	363-399 incl; 412	60% of 362
39	N. Mex.	406-411 incl	-
40	Arizona	400-405 incl	-
41	Montana	413-420 incl	-
42	Idaho	421-425 incl	-
43	Wyoming	426-429 incl	-
44	Colorado	437-445 incl	-
45	Utah	430-436 incl	-
46	Wash.	446-452 incl	14% of 455
47	Oregon	453-454 incl; 456-462 incl	86% of 455
48	Nevada	485-490 incl	-
49	Calif.	463-479 incl; 480-484 incl;	-
50	Alaska	497-505 incl	-
51	Hawaii	506-509 incl	-
52	Possessions	510-512 incl	-

FIGURE 3: MAPPING FROM NNS ZONE TO STATE.

passed to the internal subroutine ADD. ADD does all the work of stepping through the indexing scheme so that the additions are made to the proper element of each matrix. When each of the 10 matrices has been properly adjusted for the record at hand, control is returned to the driver routine which gets the next record to be processed.

When the last record from the input tape file is reached, STAG closes out the run by computing and printing out row and column sums for each matrix. The driver then writes the matrices onto logical unit 8 according to the format given in Appendix E.

5. THE BEA ZONE AGGREGATION PROGRAMS--BAG

The BAG programs are very similar in form and content to the programs discussed in Section 4. In these programs, however, the aggregation is by BEA zone (see [3, 4] for a discussion of BEA zones), rather than state, and there are 16 matrices (described below) computed rather than just 10. These two changes required several alterations to the basic programs written to handle the state aggregations. One such change had to do with storage: there are 174 BEA zones, so that the 16 matrices (with row and column sums) require 16*175*175 = 490,000 words just for data storage. Since that is more than the NBS UNIVAC 1108 can store "in core", the programs were modified to compute one matrix at a time with one pass through the data tape, then to rewind and compute another, and so on.

Another necessary modification concerned the structure of what were called in STAG the PCT and NPCT tables. Recall that these tables provided the indexing and percentages for those zones that crossed state boundaries. The same situation obtains here, but in BAG, the two tables have been combined and more information has been stored in the resultant table.

The following discussion refers to two arrays: PCT and IPCT. These two arrays have been "equivalenced: so that both integer and real values can be stored in adjacent words of the same array. Letting I be an NNS zone number, then as before NDX(I) is the BEA zone number to which the flows to and from NNS zone I should be added. But if NDX(I) < 0, then letting IAD=-NDX(I), we now have that IPCT(IAD) contains two numbers "packed" into the same word. The leftmost number, retrieved by letting NL=IPCT(IAD)/1000, is a pointer back into NDX. The rightmost number, retrieved by letting NR=IPCT(IAD)-NL*1000, is the number of BEA zones that are intersected by NNS zone I. There will be NR groups of 3 words each in the PCT table following word IAD. In each group, the third word is the zone number of one of the BEA zones intersected by NNS zone I. The second word is the proportion of total earnings of NNS zone I that accrued from that BEA zone, and the first word is the proportion of total income of NNS zone I that accrued from the BEA zone. The income proportion is used in the aggregation when the NNS zone in question appears as a destination zone in the input record from the split tape. The earnings proportion is used when the zone appears as an origin in that record.

The DATA statements in BAG that initialize the PCT table were created by a set of programs not included in the NNDBS. Documentation of those programs, and the source of the earnings and income proportions used in the DATA statements, appears in reference [3].

The subroutine BAG works similarly to STAG. A logical record is passed into it, volume or costs are computed and aggregated, and the internal subroutine ADD is called to step through the indexing scheme and add the quantities to the proper element of the matrix currently being computed. The number (1-16) of the matrix being computed is passed in the calling sequence in the variable M, and the matrix itself is passed in BMTX.

The matrices computed correspond respectively to the volume and cost matrices for the same eight commodity classes listed in Section 2.

The only major difference between the driver program for BAG and that for STAG is that the computation of each matrix in BAG requires one complete pass through the tape. After each matrix is computed, its row and column sums are computed, it is dumped out in binary form on unit 8, the input tape (unit 7) is rewound, and the next phase of matrix computation is begun.

The computer format of the output tape from a run of BAG is given in Appendix F.

6. CONCLUSION

The National Network Data Base System was designed with one main goal in mind: to provide a maximum of useful data on the inter-city freight and passenger flows along the major transportation network of the United States, while keeping the number of data words actually stored to a minimum. This approach also permits a great deal of flexibility in data-base usage: a capability to change easily the modal split equation, to revise cost functions, to aggregate over a slightly modified index set, to modify the network, and so on.

In fact, this ability to exercise the data-base system and to derive a set of data different from that built into it has already been used successfully. A request was made to compute state and BEA zone aggregations for 1972, a year for which no flow data existed in the data base. What was provided, however, was a set of state and BEA zone growth rates from 1970 to 1972, based on total earnings and total income for the states and BEA zones. The request was satisfied with ease by applying the given growth rates to the row and column sums of the matrices already produced in the system, normalizing the new sums with respect to the "expanded grand total" of the matrices, and then using a standard matrix scaling algorithm (*) to adjust the elements to the new sums.

This is one of the many possible examples of the flexibility of such a data base "system". Other uses are anticipated.

(*) The program (SCALE) is a part of NBS's Mathematical Programming Algorithms collection and is based on the procedure outlined on p. 115 of [1].

7. BIBLIOGRAPHY

1. Deming, W. E., Statistical Adjustment of Data, John Wiley and Sons, New York, 1938.

2. Fratar, T. J., "Vehicular Trip Distribution by Successive Approximations", Traffic Quarterly, Vol. VIII, No. 1, January 1954, pp. 53-65.

3. Lozier, D. W., and Stiehler, J. R., "Machine-Readable Mapping from National Network Simulation (NNS) Zones to Bureau of Economic Analysis (BEA) Areas", NBSIR 75-918, September, 1975.

4. Schofer, R., "Final Report on the National Network Simulation", NBSIR 75-912. (in preparation)

5. U. S. Department of Transportation, FHWA Computer Programs for Urban Transportation Planning, July, 1974.

APPENDIX A: COMPUTER FORMAT OF NET70, NET80, AND NET90

DENSITY: 556 f.p.i.
PARITY: even
TRACKS: 7
FORMAT: binary
LOGICAL RECORD LENGTH: 22 words
BLOCKING FACTOR: 100
NUMBER OF LOGICAL RECORDS: 68,227
RECORD INTERPRETATION:

WORD	MEANING	SYMBOL
1	Origin NNS zone	O
2	Destination NNS zone	D
3	Distance by truck (mi)	TD
4	Time by truck (min.)	TT
5	Distance by rail	RD
6	Time by rail	RT
7	Distance by air	AD
8	Time by air	AT
9	Distance by water	WD
10	Time by water	WT
11	Distance by crude pipeline	PCD
12	Time by crude pipeline	PCT
13	Distance by refined pipeline	PRD
14	Time by refined pipeline	PRT
15	Large, high value shipment volume (tons)	HL
16	Small, high value shipment volume (tons)	HS
17	Large, medium value shipment volume (tons)	ML
18	Small, medium value shipment volume (tons)	MS
19	Large, low value shipment volume (tons)	LL
20	Small, low value shipment volume (tons)	LS
21	Large, crude oil shipment volume (tons)	CL
22	Large, refined oil shipment volume (tons)	RL

APPENDIX B: COMPUTER FORMAT OF THE PASSENGER TAPE--NETPS

DENSITY: 556 f.p.i.
PARITY: even
TRACKS: 7
FORMAT: binary
LOGICAL RECORD LENGTH: 14 words
BLOCKING FACTOR: 100
NUMBER OF LOGICAL RECORDS: 54,128
RECORD INTERPRETATION:

WORD	MEANING
1	Origin NNS zone number
2	Destination NNS zone number
3	Distance by auto (mi)
4	Time by auto (min.)
5	Number of daily auto passengers, non-business purposes
6	Number of daily auto passengers, business purposes
7	Distance by air (mi)
8	Time by air (min.)
9	Number of daily air passengers, non-business purposes
10	Number of daily air passengers, business purposes
11	Distance by rail (mi)
12	Time by rail (min.)
13	Number of daily rail passengers, non-business purposes
14	Number of daily rail passengers, business purposes.

APPENDIX C: COMPUTER FORMAT OF THE ALTERNATE IMPEDANCE TAPE--NETTS

DENSITY: 556 f.p.i.
PARITY: even
TRACKS: 7
FORMAT: binary
LOGICAL RECORD LENGTH: 14
BLOCKING FACTOR: 100
NUMBER OF LOGICAL RECORDS: 68,227
RECORD INTERPRETATION: Exactly the same as the first 14 words listed in Appendix A.

APPENDIX D: COMPUTER FORMAT OF THE SPLIT TAPES--SPL70, SPL80, AND SPL90

DENSITY: 800 f.p.i.
PARITY: even
TRACKS: 7
FORMAT: binary
LOGICAL RECORD LENGTH: 37 words
BLOCKING FACTOR: 100
NUMBER OF LOGICAL RECORDS: 68,227
RECORD INTERPRETATION:

WORD	MEANING	SYMBOL
1	Origin NNS zone	O
2	Destination NNS zone	D
3	Distance by truck (mi)	TD
4	Time by truck (min.)	TT
5	Distance by rail	RD
6	Time by rail	RT
7	Distance by air	AD
8	Time by air	AT
9	Distance by water	WD
10	Time by water	WT
11	Distance by crude pipeline	PCD
12	Time by crude pipeline	PCT
13	Distance by refined pipeline	PRD
14	Time by refined pipeline	PRT
15	Tons of high value, large shipments sent by truck	THL
16	Tons of high value, small shipments sent by truck	THS
17	Tons of medium value, large shipments sent by truck	TML
18	Tons of medium value, small shipments sent by truck	TMS
19	Tons of low value, large shipments sent by truck	TLL
20	Tons of low value, small shipments sent by truck	TLS
21	Tons of large, refined oil shipments sent by truck	TRL
22	Tons of high value, large shipments sent by rail	RHL
23	Tons of high value, small shipments sent by rail	RHS
24	Tons of medium value, large shipments sent by rail	RML
25	Tons of medium value, small shipments sent by rail	RMS
26	Tons of low value, large shipments sent by rail	RLL
27	Tons of low value, small shipments sent by rail	RLS
28	Tons of large, refined oil shipments sent by rail	RRL
29	Tons of high value, small shipments sent by air	AHS
30	Tons of high value, large shipments sent by water	WHL
31	Tons of medium value, large shipments sent by water	WML
32	Tons of low value, large shipments sent by water	WMS
33	Tons of large, crude oil shipments sent by water	WCL
34	Tons of large, refined oil shipments sent by water	WRL
35	Tons of large, crude oil shipments sent by pipeline	PCL
36	Tons of large, refined oil shipments sent by pipeline	PRL
37	Sum over fields 15 through 37	

APPENDIX E: COMPUTER FORMAT OF THE STAG TAPES--STG70, STG80, AND STG90

DENSITY: 556 f.p.i.
PARITY: even
TRACKS: 7
FORMAT: binary
LOGICAL RECORD LENGTH: 2809
BLOCKING FACTOR: 0
NUMBER OF LOGICAL RECORDS: 10
RECORD INTERPRETATION: Each record contains one of the 53 x 53 matrices created by STAG, written out column by column. They are, in order of appearance on the tape, state-to-state aggregations of the following:

1) high value, large and small flows (tons).
2) the transport costs of number 1 (dollars),
3) medium value, large and small flows,
4) the transport costs of number 3,
5) low value, large and small flows,
6) the transport costs of number 4,
7) crude oil flows,
8) the transport costs of number 7,
9) refined oil flows,
10) the transport costs of number 8.

Reference Figure 3 for which state is associated with each row and column number. The row and column sums appear in the last row and column.

APPENDIX F: COMPUTER FORMAT OF THE BAG TAPES--BAG70, BAG80, AND BAG90

DENSITY: 556 f.p.i.
PARITY: even
TRACKS: 7
FORMAT: binary
LOGICAL RECORD LENGTH: 30625
BLOCKING FACTOR: 0
NUMBER OF LOGICAL RECORDS: 16
RECORD INTERPRETATION: Each record contains one of the 175 x 175 matrices created by BAG, written out in column order. The matrices are, in order of appearance on the tape, BEA zone to BEA zone aggregations of the following:

1.) high value, large flows (tons),
2.) the transport costs of number 1 (dollars),
3.) high value, small flows,
4.) the transport costs of number 3,
5.) medium value, large flows,
6.) the transport costs of number 5,
7.) medium value, small flows,
8.) the transport costs of number 7,
9.) low value, large flows,
10.) the transport costs of number 7,
11.) low value, small flows,
12.) the transport costs of number 9,
13.) crude oil flows,
14.) the transport costs of number 13,
15.) refined oil flows,
16.) the transport costs of number 15.

APPENDIX G: LISTING OF THE SPLIT PROGRAMS

LISTING OF CONTROL STREAM FOR A RUN OF SPLIT

```
@RUN,N/R RICJAK,36131-JACKSO,BEA,15,100,D2000
@MSG,W 36131-JACKSO.  WRITE ENABLE REEL SPL70.
@MSG THANKS....RIC
@ASG,TJM 7.,U,NET70
@ASG,TJH 8.,U,SPL70W
@ASG,AX SUB.
@ASG,AX DRV.
@FOR,S DRV.SPLIT,TPF$.A
@FOR,S SUB.SPLIT,TPF$.B
@FREE SUB.
@FREE DRV.
@XQT
@CHARGE

END ONSITE PRINTOUT ON OCTOBER 6, 1975 AT 13:42:49
BEA*RUN(1).SPLIT(5)
```

LISTING OF DRIVER FOR SUBROUTINE SPLIT

```
      PARAMETER MIR=22,MOR=37,MAR=1
      IMPLICIT INTEGER (A-Z)
      LOGICAL ALT
      DIMENSION IR(MIR,100),OR(MOR,100),S(MOR),AR(MAR,100)
      DATA ALT/.FALSE./
      WRITE(6,6)
C**** READ A BLOCK FROM THE INPUT TAPE.
   10 CALL NTRAN(7,2,MIR*100,IR(1,1),L,22)
      IF (L.LT.0) GO TO 80
C**** IF NECESSARY, READ BLOCK FROM ALTERNATE TAPE.
      IF (ALT) CALL NTRAN(9,2,MAR*100,AR(1,1),N,22)
      IF (N.LT.0) GO TO 100
C**** BEGIN LOOP WHICH PASSES OFF EACH LOGICAL RECORD, IN TURN,
C**** TO THE SUBROUTINE FOR PROCESSING.
      DO 50 J=1,100
      IF (.NOT.ALT) GO TO 30
      IF (IR(1,J).NE.AR(1,J).OR.IR(2,J).NE.AR(2,J)) GO TO 110
C**** IF ALTERNATE TAPE UP AND RECORDS MATCH, TRANSFER THE IMPEDANCES.
      DO 20 I=1,MAR
   20 IR(I,J)=AR(I,J)
   30 C=C+1
      CALL SPLIT(IR(1,J),MIR,OR(1,J),MOR)
C**** IF FIRST WORD IS ZERO, LAST BLOCK ON TAPE HAS BEEN REACHED.
      IF (IR(1,J).EQ.0) GO TO 900
C**** SUM ALL FIELDS OF OUTPUT TAPE.
      DO 40 I=1,MOR
   40 S(I)=S(I)+OR(I,J)
   50 CONTINUE
C**** WRITE A BLOCK TO THE OUTPUT TAPE.
   60 CALL NTRAN(8,1,MOR*100,OR(1,1),M,22)
      IF (M.LT.0) GO TO 90
      DO 70 I=1,MOR
      DO 70 J=1,100
   70 OR(I,J)=0
      GO TO 10
   80 WRITE(6,2) L,C
      GO TO 900
   90 WRITE(6,1) M,C
      GO TO 900
  100 WRITE(6,7) N,C
      GO TO 900
  110 WRITE(6,8)
C**** WRITE OUT LAST BLOCK AND ENDFILE.
  900 CALL NTRAN(8,1,MOR*100,OR(1,1),M,22)
      CALL NTRAN(8,9)
      WRITE(6,4) C
      WRITE(6,5)
      WRITE(6,3) (I,S(I),I=1,MOR)
    1 FORMAT(' ** ERROR M='I5,' C='I5)
    2 FORMAT(' ** ERROR L='I5,' C='I5)
    3 FORMAT(2XI5,I15)
    4 FORMAT(/1XI10,' LOGICAL RECORDS WERE TRANSFERRED.')
    5 FORMAT(//' SUM OVER ALL OUTPUT TAPE FIELDS'///2X'FIELD'12X'SUM')
    6 FORMAT('1')
    7 FORMAT(' ** ERROR N='I5,' C='I5)
    8 FORMAT(' ** ERROR C='I5,' TAPE RECORDS DO NOT MATCH.')
      END
```

```
      SUBROUTINE SPLIT(IR,MIR,OR,MOR)
      LOGICAL NFIRST
      INTEGER OR,COMIND,CSTIND,DSTIND,TYPE,SUMNO,SUMTN,SUMTM
      INTEGER SUMRV
      DIMENSION IR(MIR),OR(MOR),ALPHA(22),BETA(22),RHO(22),OMEGA(22)
      DIMENSION COST(10),CSTIND(22),DSTIND(22),COMIND(22),SUM(8)
      DIMENSION TYPE(22),SUMNO(22),SUMTN(22),SUMTM(22),SUMRV(22)
      DIMENSION R(3)
      DATA NFIRST/.FALSE./
      DATA RHO/7.8124,15.4762,8.1321,10.1584,6.9884,8.0848,19.0570,1.0,
     *          1.0,1.0,1.0,1.0,1.0,1.0,-7.4047,.8019,1.4242,-.0638,
     *          1.0,.768,8.623,1.8/
      DATA BETA/-.9299,-3.2584,-.9299,-.869,-1.4726,-.869,-2.785,-.2192,
     *          -2.1104,-.2192,-.2288,-.7732,-.2288,-1.1186,-1.5626,
     *          -.4888,-.4888,-.7498,-.2466,-.9052,-.7928,-.8466/
      DATA ALPHA/-.2944,-1.4294,-.2944,-.8458,-.5345,-.8458,-1.3691,
     *          0.0,-.7922,0.0,-.5756,-.2251,-.5756,-.6018,-.1332,
     *          -.0892,-.0892,-.1184,-.7255,-.3275,-1.0826,-.5963/
      DATA TYPE/'THL','THS','TML','TMS','TLL','TLS','TRL','RHL',
     *          'RHS','RML','RMS','RLL','RLS','RRL','AHS','WHL',
     *          'WML','WLL','WCL','WRL','PCL','PRL'/
      DATA COMIND/1,2,3,4,5,6,8,1,2,3,4,5,6,8,2,1,3,5,7,8,7,8/
      DATA CSTIND/1,1,2,2,3,3,3,4,4,5,5,6,6,6,7,8,8,8,8,8,9,10/
      DATA DSTIND/3,3,3,3,3,3,3,5,5,5,5,5,5,5,7,9,9,9,9,9,11,13/
C**** CHECK IF THIS IS LAST LOGICAL RECORD.
      IF (IR(1).EQ.0) GO TO 800
      IF (NFIRST) GO TO 100
C**** IF FIRST TIME THROUGH, RESET RHO.
      DO 10 I=1,22
   10 RHO(I)=EXP(RHO(I))
      NFIRST=.TRUE.
C**** COMPUTE COSTS (PENNIES/TON).  C(1)-C(10) ARE, IN ORDER:
C**** CTH,CTM,CTL,CRH,CRM,CRL,CA,CW,CC,CR.
  100 COST(1)=.8817*(-1216.0+924.0*IR(3)**.421)
      COST(2)=.4961*(-1216.0+924.0*IR(3)**.421)
      COST(3)=.4188*(-1216.0+924.0*IR(3)**.421)
      COST(4)=718.+2.96*IR(5)
      COST(5)=341.+1.70*IR(5)
      COST(6)=21.4+1.47*IR(5)
      COST(7)=10540.+18.92*IR(7)
      COST(8)=296.5+.644*IR(9)
      COST(9)=25.84+.2324*IR(11)
      COST(10)=.4148*IR(13)
      R(1)=1349.5251+3.5187860*IR(3)
      R(2)=992.43256+3.1580756*IR(3)
      R(3)=768.85476+2.6590308*IR(3)
      DO 150 I=1,3
      COST(I)=MIN(MAX(COST(I),R(I)),1.17*R(I))
  150 CONTINUE
C**** COMPUTE THE UNNORMALIZED WEIGHTS.
  200 DO 210 I=1,22
      OMEGA(I)=0.0
      J1=DSTIND(I)
      IF (IR(J1).GE.99999.OR.IR(J1+1).GE.99999) GO TO 210
      J2=CSTIND(I)
      IF (COST(J2).EQ.0.0.OR.IR(J1+1).EQ.0) GO TO 210
      OMEGA(I)=RHO(I)*COST(J2)**BETA(I)*FLOAT(IR(J1+1))**ALPHA(I)
  210 CONTINUE
```

LISTING OF SUBROUTINE SPLIT

```
C**** SUM THE WEIGHTS ACROSS MODE TO GET NORMALIZING FACTOR.
      DO 310 I=1,8
  310 SUM(I)=0.0
      DO 320 I=1,22
      J=COMIND(I)
  320 SUM(J)=SUM(J)+OMEGA(I)
C**** NORMALIZE THE WEIGHTS.
      DO 420 I=1,22
      J=COMIND(I)
      IF (SUM(J).GT.0) GO TO 410
      OMEGA(I)=0.0
      GO TO 420
  410 OMEGA(I)=OMEGA(I)/SUM(J)
  420 CONTINUE
C**** SPLIT THE SHIPMENTS BY MULTIPLYING THEM BY THE WEIGHTS.
      DO 510 I=1,22
      J=COMIND(I)+14
  510 OMEGA(I)=OMEGA(I)*FLOAT(IR(J))
C**** BUILD THE OUTPUT RECORD.
      DO 610 I=1,14
  610 OR(I)=IR(I)
      OR(37)=0
      DO 620 I=1,22
      OR(I+14)=OMEGA(I)+.5
  620 OR(37)=OR(37)+OR(I+14)
C**** SUM THE TONS, TON-MILES, AND REVENUES.
      DO 720 I=1,22
      J1=DSTIND(I)
      IF (IR(J1).GE.99999) GO TO 720
      SUMTM(I)=SUMTM(I)+OR(I+14)*IR(J1)
      J2=CSTIND(I)
      SUMRV(I)=SUMRV(I)+OR(I+14)*COST(J2)
      SUMTN(I)=SUMTN(I)+OR(I+14)
      SUMNO(I)=SUMNO(I)+1
  720 CONTINUE
      GO TO 999
C**** OUTPUT THE TABLE OF SUMS.
  800 WRITE(6,1)
      WRITE(6,2)(TYPE(I),SUMNO(I),SUMTN(I),SUMTM(I),SUMRV(I),I=1,22)
    1 FORMAT(//12X,'TOTAL'7X'TOTAL'11X'TOTAL'10X'TOTAL'/1X'TYPE'7X
     *       'COUNT'8X'TONS'7X'TON-MILES'7X'REVENUES'//)
    2 FORMAT(2XA3,I12,I12,I16,I15)
  999 RETURN
      END
```

APPENDIX H: LISTING OF THE STAG PROGRAMS

LISTING OF CONTROL STREAM FOR A RUN OF STAG

```
@RUN,M/R RICJAK,36131-JACKSO,BEA,5,100
@ASG,AX DRV.
@ASG,AX SUB.
@ASG,A STOUT.
@ASG,TJH 7.,U,SPL70
@USE 8.,STOUT.
@FOR,S DRV.STAG,TPF$.A
@FOR,S SUB.STAG,TPF$.B
@FREE DRV.
@FREE SUB.
@XQT
@COST
```

LISTING OF DRIVER FOR SUBROUTINE STAG

```
      PARAMETER MIR=37,MVD=53
      INTEGER C
      DIMENSION IR(MIR,100)
      DIMENSION VOLS(MVD,MVD,5),DOLS(MVD,MVD,5)
C**** READ A BLOCK FROM INPUT TAPE.
   10 CALL NTRAN(7,2,MIR*100,IR(1,1),L,22)
      IF (L.LT.0) GO TO 80
C**** BEGIN LOOP TO PASS OFF EACH LOGICAL RECORD TO SUBROUTINE.
      DO 50 J=1,100
      CALL STAG(IR(1,J),VOLS,DOLS)
      IF (IR(1,J).EQ.0) GO TO 100
      C=C+1
   50 CONTINUE
      GO TO 10
   80 WRITE(6,2) L,C
C**** WRITE OUT AGGREGATED MATRICES ONTO UNIT 8.
  100 DO 110 K=1,5
      CALL NTRAN(8,1,MVD*MVD,VOLS(1,1,K),M,22)
  110 CALL NTRAN(8,1,MVD*MVD,DOLS(1,1,K),M,22)
      CALL NTRAN(8,9)
C**** PRINT OUT ROW AND COLUMN SUMS.
      DO 150 K=1,5
      WRITE(6,3) K
      WRITE(6,4) (VOLS(I,MVD,K),I=1,MVD)
      WRITE(6,5) K
      WRITE(6,4) (VOLS(MVD,I,K),I=1,MVD)
      WRITE(6,6) K
      WRITE(6,4) (DOLS(I,MVD,K),I=1,MVD)
      WRITE(6,7) K
  150 WRITE(6,4) (DOLS(MVD,I,K),I=1,MVD)
    2 FORMAT(' ** ERROR L='I5,' C='I5)
    3 FORMAT(/' ROW SUMS FOR VOLUME MATRIX NUMBER'I3/)
    4 FORMAT(10(E13.7))
    5 FORMAT(/' COLUMN SUMS FOR VOLUME MATRIX NUMBER'I3/)
    6 FORMAT(/' ROW SUMS FOR COST MATRIX NUMBER'I3/)
    7 FORMAT(/' COLUMN SUMS FOR COST MATRIX NUMBER'I3/)
      END
```

```
      SUBROUTINE STAG(IR,VOLS,DOLS)
      DIMENSION VOLS(53,53,5),DOLS(53,53,5)
      DIMENSION IR(37),PCT(26,3),NPCT(26,3),NDX(512)
      DATA ((PCT(I,J),NPCT(I,J),J=1,3),I=1,19)/
     1 .25,   5, .75,   6, .00,   0,
     2 .20,   8, .80,   9, .00,   0,
     3 .21,   8, .79,   9, .00,   0,
     4 .12,   8, .77,  10, .11,  11,
     5 .43,  11, .28,  12, .29,  25,
     6 .58,  14, .42,  26, .00,   0,
     7 .44,  14, .56,  26, .00,   0,
     8 .81,  14, .03,  15, .16,  27,
     9 .16,  13, .84,  14, .00,   0,
     * .82,  15, .18,  27, .00,   0,
     1 .54,  16, .46,  19, .00,   0,
     2 .22,  14, .57,  26, .21,  27,
     3 .16,  15, .84,  27, .00,   0,
     4 .94,  28, .06,  36, .00,   0,
     5 .83,  28, .17,  31, .00,   0,
     6 .36,  30, .64,  31, .00,   0,
     7 .81,  31, .19,  33, .00,   0,
     8 .65,  36, .35,  37, .00,   0,
     9 .78,  17, .22,  18, .00,   0/
      DATA ((PCT(I,J),NPCT(I,J),J=1,3),I=20,26)/
     * .89,  19, .11,  23, .00,   0,
     1 .65,  20, .35,  24, .00,   0,
     2 .23,  16, .77,  20, .00,   0,
     3 .39,  18, .61,  21, .00,   0,
     4 .16,  19, .84,  23, .00,   0,
     5 .40,  36, .60,  38, .00,   0,
     6 .14,  46, .86,  47, .00,   0/
C****      1    2    3    4    5    6    7    8    9   10
      DATA (NDX(I),I=1,190)/
     1   1,   1,   1,   1,   1,   2,   2,   2,   2,   3,
     2   3,   4,   4,   4,   4,   4,   4,   4,   4,   4,
     3   4,   4,   4,   4,   4,   4,   5,  -1,   6,   6,
     4   6,   6,   6,   6,   6,   6,   6,   6,   6,   6,
     5   7,   7,   7,   7,   7,   7,   7,   7,   7,   7,
     6   9,   7,   7,   7,   7,   7,   7,   7,   8,   8,
     7   8,   8,   8,   8,   8,  -2,   9,   9,  -3,   9,
     8   9,   9,   9,   9,   9,   9,   9,   9,   9,   9,
     9   9,   9,   9,  -4,  10,  11,  11,  11,  11,  -5,
     *  14,  14,  14,  14,  14,  14,  14,  -6,  -7,  14,
     1  14,  14,  14,  14,  14,  -8,  14,  14,  14,  14,
     2  14,  14,  -9,  14,  15,  15,  15,  15,  15,  15,
     3  15,  15,  15, -10,  15,  15,  15,  15,  15,  15,
     4  16,  16,  16,  16, -11,  16,  16,  16,  16,  16,
     5  16,  16,  16,  16,  16,  16,  16,  16,  13,  13,
     6  13,  13,  13,  13,  13,  13,  13,  13,  13,  13,
     7  13,  13,  13,  13,  13,  13,  13,  17,  17,  17,
     8  17,  17,  17,  17,  17,  17,  17,  17,  17,  17,
     9  17,  25,  25,  25,  25,  25,  25,  25,  25,  25/
      DATA (NDX(I),I=191,380)/
     *  25,  25,  25,  25,  26,  26,  26,  26,  26, -12,
     1  27,  27,  27,  27, -13,  27,  28, -14,  28,  28,
     2  28, -15,  28,  28,  28,  29,  29,  29,  29,  29,
     3  29,  29,  29,  29,  29,  29,  29,  29,  29,  30,
     4  30,  30,  30,  30,  30,  31,  31,  31, -16,  31,
```

```
     5 31, 31, 31, 31,-17, 31, 31, 32, 32, 32,
     6 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
     7 33, 33, 33, 33, 33, 33, 33, 33, 33, 33,
     8 33, 34, 34, 34, 34, 35, 35, 35, 35, 35,
     9 35, 35, 35, 35, 35, 36, 36, 36, 36, 36,
     * 36,-18, 36, 36, 18,-19, 18, 18, 18, 18,
     1 18, 19,-20, 19, 19, 19, 19, 19, 19, 19,
     2 19, 19, 19, 19, 20, 20, 20,-21, 20, 20,
     3 20, 20,-22, 21, 21, 21, 21,-23, 22, 22,
     4 22, 22, 22, 23, 23, 23, 23, 23, 23, 23,
     5 23, 23,-24, 24, 24, 24, 24, 24, 24, 24,
     6 24, 24, 37, 37, 37, 37, 37, 37, 37, 37,
     7 37,-25, 38, 38, 38, 38, 38, 38, 38, 38,
     8 38, 38, 38, 38, 38, 38, 38, 38, 38, 38/
      DATA (NDX(I),I=381,512)/
     9 38, 38, 38, 38, 38, 38, 38, 38, 38, 38,
     * 38, 38, 38, 38, 38, 38, 38, 38, 38, 40,
     1 40, 40, 40, 40, 40, 39, 39, 39, 39, 39,
     2 39, 38, 41, 41, 41, 41, 41, 41, 41, 41,
     3 42, 42, 42, 42, 42, 43, 43, 43, 43, 45,
     4 45, 45, 45, 45, 45, 45, 44, 44, 44, 44,
     5 44, 44, 44, 44, 44, 46, 46, 46, 46, 46,
     6 46, 46, 47, 47,-26, 47, 47, 47, 47, 47,
     7 47, 47, 49, 49, 49, 49, 49, 49, 49, 49,
     8 49, 49, 49, 49, 49, 49, 49, 49, 49, 49,
     9 49, 49, 49, 49, 48, 48, 48, 48, 48, 48,
     *  0,  0,  0,  0,  0,  0, 50, 50, 50, 50,
     1 50, 50, 50, 50, 50, 51, 51, 51, 51, 52,
     2 52, 52/
C****
      IF (IR(1).EQ.0) GO TO 100
      CTH=.8817*(-1216.0+924.0*IR(3)**.421)
      CTM=.4961*(-1216.0+924.0*IR(3)**.421)
      CTL=.4188*(-1216.0+924.0*IR(3)**.421)
      RTH=1349.5251+3.5187860*IR(3)
      RTM=992.43256+3.1580756*IR(3)
      RTL=768.85476+2.6590308*IR(3)
      CTH=MIN(MAX(CTH,RTH),1.17*RTH)
      CTM=MIN(MAX(CTM,RTM),1.17*RTM)
      CTL=MIN(MAX(CTL,RTL),1.17*RTL)
      CRH=718.+2.96*IR(5)
      CRM=341.+1.70*IR(5)
      CRL=21.4+1.47*IR(5)
      CA=10540.+18.92*IR(7)
      CW=296.5+.644*IR(9)
      CPC=25.84+.2324*IR(11)
      CPR=.4148*IR(13)
C**** DO COMPUTATIONS FOR HIGH VALUE, LARGE AND SMALL.
      DOL=(IR(23)+IR(22))*CRH+(IR(15)+IR(16))*CTH+IR(30)*CW+IR(29)*CA
      VOL=IR(23)+IR(22)+IR(15)+IR(16)+IR(30)+IR(29)
      CALL ADD(1)
C**** DO COMPUTATIONS FOR MED VALUE,LARGE AND SMALL.
      DOL=(IR(24)+IR(25))*CRM+(IR(17)+IR(18))*CTM+IR(31)*CW
      VOL=IR(24)+IR(25)+IR(17)+IR(18)+IR(31)
      CALL ADD(2)
C**** DO COMPUTATIONS FOR LOW VALUE, LARGE AND SMALL.
      DOL=(IR(26)+IR(27))*CRL+(IR(19)+IR(20))*CTL+IR(32)*CW
      VOL=IR(26)+IR(27)+IR(19)+IR(20)+IR(32)
```

```
      CALL ADD(3)
C**** DO COMPUTATIONS FOR CRUDE.
      DOL=IR(33)*CW+IR(35)*CPC
      VOL=IR(33)+IR(35)
      CALL ADD(4)
C**** DO COMPUTATIONS FOR REFINED.
      DOL=IR(28)*CRL+IR(21)*CTL+IR(34)*CW+IR(36)*CPR
      VOL=IR(28)+IR(21)+IR(34)+IR(36)
      CALL ADD(5)
      GO TO 200
  100 DO 110 K=1,5
      DO 110 J=1,52
      DO 110 I=1,52
      VOLS(53,53,K)=VOLS(53,53,K)+VOLS(I,J,K)
      VOLS(I,53,K)=VOLS(I,53,K)+VOLS(I,J,K)
      VOLS(53,J,K)=VOLS(53,J,K)+VOLS(I,J,K)
      DOLS(53,53,K)=DOLS(53,53,K)+DOLS(I,J,K)
      DOLS(I,53,K)=DOLS(I,53,K)+DOLS(I,J,K)
  110 DOLS(53,J,K)=DOLS(53,J,K)+DOLS(I,J,K)
  200 RETURN
C**** SUBROUTINE TO INTERPRET SUBSCRIPTS AND DO THE AGGREGATION.
      SUBROUTINE ADD(AK)
      IMPLICIT INTEGER (A)
      AI=IR(1)
      AI=NDX(AI)
      AJ=IR(2)
      AJ=NDX(AJ)
      IF (AI) 30,900,10
   10 IF (AJ) 60,900,20
C**** AGGREGATE FOR CASE OF NO SMSA'S.
   20 VOLS(AI,AJ,AK)=VOLS(AI,AJ,AK)+VOL
      DOLS(AI,AJ,AK)=DOLS(AI,AJ,AK)+DOL
      GO TO 999
   30 IF (AJ) 80,900,40
C**** AGGREGATE FOR CASE OF ORG AN SMSA.
   40 DO 50 A1=1,3
      A2=NPCT(-AI,A1)
      IF (A2.EQ.0) GO TO 999
      VOLS(A2,AJ,AK)=VOLS(A2,AJ,AK)+PCT(-AI,A1)*VOL
   50 DOLS(A2,AJ,AK)=DOLS(A2,AJ,AK)+PCT(-AI,A1)*DOL
      GO TO 999
C**** AGGREGATE FOR CASE OF DEST AN SMSA.
   60 DO 70 A1=1,3
      A2=NPCT(-AJ,A1)
      IF (A2.EQ.0) GO TO 999
      VOLS(AI,A2,AK)=VOLS(AI,A2,AK)+PCT(-AJ,A1)*VOL
   70 DOLS(AI,A2,AK)=DOLS(AI,A2,AK)+PCT(-AJ,A1)*DOL
      GO TO 999
C**** AGGREGATE FOR CASE OF BOTH ORG AND DEST AN SMSA.
   80 DO 90 A1=1,3
      A2=NPCT(-AI,A1)
      IF (A2.EQ.0) GO TO 999
      DO 90 A3=1,3
      A4=NPCT(-AJ,A3)
      IF (A4.EQ.0) GO TO 90
      VOLS(A2,A4,AK)=VOLS(A2,A4,AK)+PCT(-AI,A1)*PCT(-AJ,A3)*VOL
   90 DOLS(A2,A4,AK)=DOLS(A2,A4,AK)+PCT(-AI,A1)*PCT(-AJ,A3)*DOL
      GO TO 999
```

```
  900 WRITE(6,1) AI,AJ,AK,IR
    1 FORMAT(' ERROR** '/(10I10))
  999 RETURN
      END

END ONSITE PRINTOUT ON OCTOBER 6, 1975 AT 13:48:57
BEA*SUB(1).STAG(5)
```

APPENDIX I: LISTING OF THE BAG PROGRAMS

LISTING OF CONTROL STREAM FOR A RUN OF BAG

```
@RUN,N RICJAK,36131-JACKSO,BEA,40/D2000,125
@ASG,AX DRV.
@ASG,AX SUB.
@ASG,TJH 7.,U,SPL70
@ASG,A BAGOUT.
@USE 8.,BAGOUT.
@FOR,S DRV.BAG,TPF$.A
@FOR,S SUB.BAG,TPF$.B
@FREE DRV.
@FREE SUB.
@XQT
@COST
```

LISTING OF DRIVER ROUTINE FOR SUBROUTINE BAG

```
      PARAMETER MIR=37,MB=175,MB1=MB-1
      INTEGER C
      DIMENSION IR(MIR,100)
      DIMENSION B(MB,MB)
C**** LOOP TO PROCESS ALL 16 MATRICES.
      DO 300 MAT=1,16
      C=0
C**** READ A BLOCK FROM THE INPUT TAPE.
   10 CALL NTRAN(7,2,MIR*100,IR(1,1),L,22)
      IF (L.LT.0) GO TO 80
C**** BEGIN LOOP TO PASS OF EACH LOGICAL RECORD TO SUBROUTINE.
      DO 50 J=1,100
      IF (IR(1,J).EQ.0) GO TO 100
      C=C+1
   50 CALL BAG(IR(1,J),MAT,B)
      GO TO 10
   80 WRITE(6,2) L,C
    2 FORMAT(' ** ERROR L='I5,' C='I5)
C**** COMPUTE ROW AN COLUMN SUMS.
  100 DO 110 I=1,MB1
      DO 110 J=1,MB1
      B(I,MB)=B(I,MB)+B(I,J)
  110 B(MB,J)=B(MB,J)+B(I,J)
      B(MB,MB)=0.0
      DO 120 I=1,MB1
  120 B(MB,MB)=B(MB,MB)+B(I,MB)
C**** REWIND THE INPUT TAPE AND WRITE OUT THE MATRIX.
      CALL NTRAN(7,10)
      CALL NTRAN(8,1,30625,B,M,22)
      WRITE(6,3) MAT
      WRITE(6,1) (B(I,MB),I=1,MB)
      WRITE(6,4) MAT
      WRITE(6,1) (B(MB,J),J=1,MB)
C**** ZERO OUT THE MATRIX FOR THE NEXT PASS.
      DO 200 I=1,MB
      DO 200 J=1,MB
  200 B(I,J)=0.0
    1 FORMAT(10E13.7)
    3 FORMAT(/' ROW SUMS FOR MATRIX NUMBER'I3/)
    4 FORMAT(/' COL SUMS FOR MATRIX NUMBER'I3/)
  300 CONTINUE
      END
```

```
      SUBROUTINE BAG(IR,M,PMTX)
      DIMENSION IR(37),PCT(1913),IPCT(1913)
      DIMENSION NDX(512),PMTX(175,175)
      EQUIVALENCE (PCT,IPCT)
      DATA (NDX(I),I=  1, 40)/
     1    1,   -1,    2,    2,    2,   -8,    3,  -15,    4,    3,
     2  -25,    4,    4,    4,    4,    4,    4,    4,    4,    4,
     3  -35,    6,  -42,  -49,    4,  -56,    4,  -63,  -70,    5,
     4    5,    5,    5,    5,  -77,    5,    5,  -84,   14,   14/
      DATA (NDX(I),I= 41,230)/
     1  -91,    6,    6,    7,    7,    7,    8,    9,    9,    9,
     2   10,  -98,   12, -108,    6,   14,   14,   14,   14,   14,
     3   14, -118,   15,   15,   15,   15,   16,   15,   15, -125,
     4   13,   13, -132, -142, -152,   66,   66,   66,   16,   16,
     5   16, -162,   16,   15,   17,   17,   17, -172,   18,   18,
     6   68,   68,   67,   68,   68,   68, -179,   66,   66, -186,
     7   68, -193,   64, -200, -207,   62,   62,   63,   63, -217,
     8   69, -227,   70, -237, -244,   75, -251,   61,   61, -267,
     9 -277,   60, -287,   55,   56, -300,   59, -310,   76,   77,
     *   77,   82, -317, -330,   79, -340,   78,   78, -350,   58,
     1 -357,   57,   57,   57, -373, -380,  114, -390, -400, -410,
     2   74,   74,   71,   70,   71,   71, -420,   72,   72,   71,
     3   74, -427,   73,   73, -434, -441, -448, -455, -462, -469,
     4 -482,   85,   84,   84,   84, -495,   83, -502, -512, -522,
     5 -532,   17, -539,   22,   22, -549,   21, -556, -563, -573,
     6   20,   20, -580,   51, -590,   19, -597, -607,   52,   52,
     7 -620,   53, -633, -643,   54, -656, -669,   46, -682,   49,
     8 -689, -699, -706,   50, -713, -720,   27, -727,   26, -740,
     9   25, -750,   23,   23, -760, -767,   23, -774,   24, -781/
      DATA (NDY(I),I=231,420)/
     1 -791, -798,   29, -811,   28, -827,   44, -834,   32, -844,
     2   33, -857,   42, -870,   43,   41, -880,   38, -887,   34,
     3 -894,   35,   37, -904, -914,   36,   36,   36, -921,   39,
     4 -928,  137, -938,   40, -951,   45,   45, -964,   47,   45,
     5 -971, -981, -994,  135,-1007,  138,-1023,  138,-1030,  139,
     6  139,-1040,  133,-1047,  132,-1054,-1061,  117,  117,-1074,
     7-1081,  118,-1088,-1101,   87,   87,-1108,-1121,-1131,   91,
     8-1138,-1148,  103,-1161,-1168,  105,   81,-1178,-1191,   80,
     9-1204,-1214,  106,-1227,-1234,  111,  111,  111,-1244,  116,
     *-1257,-1264,  114,   93,-1274,-1287,-1294,   97,  100,-1301,
     1  100,-1311,   99,  101,  102,-1321,-1328,-1338,  102,-1345,
     2  108,-1355,  107,  109,-1362,  111,-1372,  110,-1382,  110,
     3-1389,  109,  119,-1396,-1403,-1419,  120,-1432,  121,-1439,
     4  122,  131,-1446,-1456,  130,  140,  141,-1466,-1485,  143,
     5-1498,  144,  144,  143,-1505,  142,  129,-1512,  128,  127,
     6  127,  127,-1522,-1529,  121,-1551,  122,  122,  122,-1564,
     7  123,-1571,  125,  124,  124,-1590,  126,-1600,  127,  162,
     8-1607,  163,  162,  162,  162,-1614,-1621,-1628,-1635,  146,
     9  145,  141,-1645,-1652,   94,-1659,-1666,   95,   95,-1679/
      DATA (NDX(I),I=421,512)/
     1-1686,-1693,-1703,-1716,-1729,-1736,-1743,-1750,-1757,  151,
     2  151,  151,  151,  151,-1767,-1774,  149,-1784,  148,  148,
     3-1797,  147,  147,-1804,  149,-1811,  154,-1818,-1825,  155,
     4-1835,  155,  157,  157,  157,-1842,  156,  159,  158,  158,
     5  158,  157,-1849,-1856,-1863,  168,  171,  171,  167,-1873,
     6-1886,  171,  171,  166,-1893,  166,-1900,  165,  165,  165,
     7  165,  165,  164,  165,  161,  161,  160,  160,  160,-1907,
     8    0,    0,    0,    0,    0,    0,  172,  172,  172,  172,
```

```
9  172,  172,  172,  172,  172,  173,  173,  173,  173,  174,
*  174,  174/
 DATA (PCT(I),I=   1,   7)/002002,
1.0792,.0626,001,
2.9204,.9374,002/
 DATA (PCT(I),I=   8,  14)/006002,
1.2723,.3296,002,
2.7277,.6704,004/
 DATA (PCT(I),I=  15,  24)/008003,
1.0935,.0755,003,
2.1845,.1641,005,
3.7220,.7604,004/
 DATA (PCT(I),I=  25,  34)/011003,
1.1900,.1700,006,
2.6100,.5900,003,
3.2000,.2400,005/
 DATA (PCT(I),I=  35,  41)/021002,
1.5669,.5410,005,
2.4331,.4590,004/
 DATA (PCT(I),I=  42,  48)/023002,
1.8078,.8274,006,
2.1922,.1726,005/
 DATA (PCT(I),I=  49,  55)/024002,
1.0071,.0070,004,
2.9929,.9930,005/
 DATA (PCT(I),I=  56,  62)/026002,
1.0310,.0330,005,
2.9690,.9670,004/
 DATA (PCT(I),I=  63,  69)/028002,
1.8984,.8929,004,
2.1016,.1071,005/
 DATA (PCT(I),I=  70,  76)/029002,
1.2836,.3609,004,
2.7164,.6391,005/
 DATA (PCT(I),I=  77,  83)/035002,
1.4966,.5057,005,
2.5034,.4943,014/
 DATA (PCT(I),I=  84,  90)/038002,
1.1034,.1245,005,
2.8966,.8755,014/
 DATA (PCT(I),I=  91,  97)/041002,
1.8537,.8823,006,
2.1463,.1177,007/
 DATA (PCT(I),I=  98, 107)/052003,
1.3012,.3009,007,
2.4251,.4648,012,
3.2737,.2342,008/
 DATA (PCT(I),I= 108, 117)/054003,
1.6756,.6715,012,
2.2195,.2313,007,
3.1049,.0972,006/
 DATA (PCT(I),I= 118, 124)/062002,
1.8770,.9156,014,
2.1230,.0844,015/
 DATA (PCT(I),I= 125, 131)/070002,
1.7241,.7453,015,
2.2759,.2547,013/
 DATA (PCT(I),I= 132, 141)/073003,
```

```
1.3890,.3320,012,
2.5308,.5801,011,
3.0802,.0879,013/
 DATA (PCT(I),I= 142, 151)/074003,
1.6628,.6866,011,
2.1651,.1579,066,
3.1721,.1556,009/
 DATA (PCT(I),I= 152, 161)/075003,
1.3399,.3251,066,
2.2970,.3009,010,
3.3631,.3740,067/
 DATA (PCT(I),I= 162, 171)/082003,
1.1061,.1171,013,
2.5700,.5903,016,
3.3239,.2926,015/
 DATA (PCT(I),I= 172, 178)/088002,
1.3178,.3579,066,
2.6822,.6421,017/
 DATA (PCT(I),I= 179, 185)/097002,
1.9659,.9543,068,
2.0341,.0457,066/
 DATA (PCT(I),I= 186, 192)/100002,
1.3757,.3953,068,
2.6243,.6047,070/
 DATA (PCT(I),I= 193, 199)/102002,
1.9623,.9275,064,
2.0377,.0725,066/
 DATA (PCT(I),I= 200, 206)/104002,
1.8229,.8524,064,
2.1771,.1476,052/
 DATA (PCT(I),I= 207, 216)/105003,
1.3910,.3718,062,
2.3544,.3676,064,
3.2546,.2606,052/
 DATA (PCT(I),I= 217, 226)/110003,
1.2650,.2815,069,
2.5555,.5515,063,
3.1796,.1670,064/
 DATA (PCT(I),I= 227, 236)/112003,
1.4940,.4595,068,
2.1478,.1242,069,
3.3583,.4164,064/
 DATA (PCT(I),I= 237, 243)/114002,
1.5782,.5992,075,
2.4218,.4008,070/
 DATA (PCT(I),I= 244, 250)/115002,
1.3281,.2560,075,
2.6719,.7440,076/
 DATA (PCT(I),I= 251, 266)/117005,
1.2422,.2160,075,
2.2397,.2562,061,
3.0948,.0710,059,
4.3892,.4269,060,
5.0341,.0300,076/
 DATA (PCT(I),I= 267, 276)/120003,
1.2026,.1934,062,
2.7171,.7416,061,
3.0803,.0650,060/
```

```
 DATA (PCT(I),I= 277, 286)/121003,
1.7184,.7638,060,
2.1826,.1608,054,
3.0990,.0754,062/
 DATA (PCT(I),I= 287, 299)/123004,
1.1286,.0951,054,
2.6389,.7192,055,
3.0758,.0541,056,
4.1567,.1316,060/
 DATA (PCT(I),I= 300, 309)/126003,
1.6249,.6557,059,
2.1313,.0973,056,
3.2438,.2470,060/
 DATA (PCT(I),I= 310, 316)/128002,
1.1575,.1405,059,
2.8425,.8595,077/
 DATA (PCT(I),I= 317, 329)/133004,
1.2763,.2882,079,
2.2276,.2250,077,
3.0634,.0485,081,
4.4327,.4383,082/
 DATA (PCT(I),I= 330, 339)/134003,
1.1502,.1236,079,
2.7717,.8199,077,
3.0781,.0565,078/
 DATA (PCT(I),I= 340, 349)/136003,
1.7459,.8027,078,
2.1783,.1434,113,
3.0758,.0540,079/
 DATA (PCT(I),I= 350, 356)/139002,
1.3912,.4189,058,
2.6088,.5811,077/
 DATA (PCT(I),I= 357, 372)/141005,
1.1243,.1277,056,
2.2463,.2588,114,
3.3888,.4044,058,
4.1215,.1232,055,
5.1190,.0859,057/
 DATA (PCT(I),I= 373, 379)/145002,
1.8455,.8455,114,
2.1545,.1545,057/
 DATA (PCT(I),I= 380, 389)/146003,
1.1769,.1755,115,
2.6528,.6663,114,
3.1703,.1583,055/
 DATA (PCT(I),I= 390, 399)/148003,
1.5059,.5415,113,
2.1849,.1255,114,
3.3092,.3330,057/
 DATA (PCT(I),I= 400, 409)/149003,
1.1646,.1320,073,
2.2308,.1904,074,
3.6046,.6776,076/
 DATA (PCT(I),I= 410, 419)/150003,
1.6198,.6233,074,
2.2432,.2454,070,
3.1370,.1313,076/
 DATA (PCT(I),I= 420, 426)/157002,
```

```
 DATA (PCT(I),I= 563, 572)/189003,
1.6910,.7490,019,
2.2033,.1699,018,
3.1057,.0811,021/
 DATA (PCT(I),I= 573, 579)/190002,
1.3843,.3130,020,
2.6157,.6870,019/
 DATA (PCT(I),I= 580, 589)/193003,
1.0148,.0104,051,
2.1379,.1068,025,
3.8473,.8828,020/
 DATA (PCT(I),I= 590, 596)/195002,
1.7292,.7092,052,
2.2708,.2908,051/
 DATA (PCT(I),I= 597, 606)/197003,
1.8136,.8368,065,
2.1264,.1171,052,
3.0600,.0461,066/
 DATA (PCT(I),I= 607, 619)/198004,
1.3715,.3719,052,
2.0173,.0129,065,
3.4892,.5379,064,
4.1220,.0773,066/
 DATA (PCT(I),I= 620, 632)/201004,
1.5070,.5463,053,
2.1919,.1899,062,
3.1825,.1618,052,
4.1186,.1019,054/
 DATA (PCT(I),I= 633, 642)/203003,
1.4032,.3985,050,
2.2970,.2856,053,
3.2998,.3159,052/
 DATA (PCT(I),I= 643, 655)/204004,
1.4124,.3817,053,
2.1244,.1233,049,
3.4367,.4747,054,
4.0266,.0203,050/
 DATA (PCT(I),I= 656, 668)/206004,
1.2890,.2749,115,
2.0606,.0374,054,
3.2690,.3191,049,
4.3813,.3686,055/
 DATA (PCT(I),I= 669, 681)/207004,
1.0271,.0136,049,
2.8131,.8196,046,
3.0649,.0577,047,
4.0950,.1090,115/
 DATA (PCT(I),I= 682, 688)/209002,
1.8906,.9024,049,
2.1094,.0976,047/
 DATA (PCT(I),I= 689, 698)/211003,
1.1711,.1256,048,
2.6332,.6808,049,
3.1957,.1936,050/
 DATA (PCT(I),I= 699, 705)/212002,
1.9823,.9878,048,
2.0177,.0122,044/
 DATA (PCT(I),I= 706, 712)/213002,
```

```
1.6017,.6670,048,
2.3983,.3330,050/
 DATA (PCT(I),I= 713, 719)/215002,
1.2814,.2501,050,
2.7186,.7499,051/
 DATA (PCT(I),I= 720, 726)/216002,
1.0687,.0788,048,
2.9313,.9212,027/
 DATA (PCT(I),I= 727, 739)/218004,
1.7764,.7916,026,
2.1419,.1415,025,
3.0653,.0579,027,
4.0165,.0090,028/
 DATA (PCT(I),I= 740, 749)/220003,
1.5123,.5432,026,
2.1849,.1861,023,
3.3028,.2708,025/
 DATA (PCT(I),I= 750, 759)/222003,
1.6228,.6360,025,
2.0430,.0307,020,
3.3342,.3332,023/
 DATA (PCT(I),I= 760, 766)/225002,
1.8259,.8425,023,
2.1741,.1575,022/
 DATA (PCT(I),I= 767, 773)/226002,
1.7255,.7402,024,
2.2745,.2598,023/
 DATA (PCT(I),I= 774, 780)/228002,
1.6884,.6632,023,
2.3116,.3368,024/
 DATA (PCT(I),I= 781, 790)/230003,
1.6552,.6559,030,
2.2632,.2631,029,
3.0817,.0811,026/
 DATA (PCT(I),I= 791, 797)/231002,
1.5707,.4759,031,
2.4293,.5241,028/
 DATA (PCT(I),I= 798, 810)/232004,
1.1464,.1526,032,
2.5426,.5355,031,
3.2759,.2873,029,
4.0351,.0245,033/
 DATA (PCT(I),I= 811, 826)/234005,
1.5139,.5096,028,
2.2832,.3234,031,
3.1232,.1082,026,
4.0319,.0226,032,
5.0478,.0362,029/
 DATA (PCT(I),I= 827, 833)/236002,
1.7615,.7365,044,
2.2385,.2635,048/
 DATA (PCT(I),I= 834, 843)/238003,
1.2169,.2319,042,
2.6365,.6587,044,
3.1466,.1094,032/
 DATA (PCT(I),I= 844, 856)/240004,
1.5959,.6014,033,
2.2034,.2008,032,
```

```
3.1781,.1832,042,
4.0226,.0146,034/
 DATA (PCT(I),I= 857, 869)/242004,
1.0932,.0990,033,
2.4545,.4611,041,
3.1705,.1449,042,
4.2818,.2950,034/
 DATA (PCT(I),I= 870, 879)/244003,
1.5412,.5120,044,
2.1627,.1654,042,
3.2961,.3226,043/
 DATA (PCT(I),I= 880, 886)/247002,
1.8041,.7997,041,
2.1959,.2003,043/
 DATA (PCT(I),I= 887, 893)/249002,
1.7427,.7600,034,
2.2573,.2400,038/
 DATA (PCT(I),I= 894, 903)/251003,
1.4188,.4876,034,
2.1601,.1077,037,
3.4211,.4047,035/
 DATA (PCT(I),I= 904, 913)/254003,
1.2752,.3379,035,
2.5891,.5370,037,
3.1357,.1251,036/
 DATA (PCT(I),I= 914, 920)/255002,
1.7003,.6406,037,
2.2997,.3594,036/
 DATA (PCT(I),I= 921, 927)/259002,
1.6203,.6060,038,
2.3797,.3940,039/
 DATA (PCT(I),I= 928, 937)/261003,
1.8826,.8994,040,
2.0311,.0258,137,
3.0862,.0748,039/
 DATA (PCT(I),I= 938, 950)/263004,
1.3909,.3888,040,
2.1987,.1468,045,
3.1658,.1810,136,
4.2446,.2834,137/
 DATA (PCT(I),I= 951, 963)/265004,
1.5374,.5492,045,
2.2805,.2779,043,
3.0261,.0174,044,
4.1560,.1555,040/
 DATA (PCT(I),I= 964, 970)/268002,
1.3637,.3190,045,
2.6363,.6810,047/
 DATA (PCT(I),I= 971, 980)/271003,
1.1035,.0779,045,
2.5286,.5625,048,
3.3679,.3596,047/
 DATA (PCT(I),I= 981, 993)/272004,
1.0859,.0924,047,
2.4196,.3859,046,
3.4397,.4718,045,
4.0548,.0499,134/
 DATA (PCT(I),I= 994,1006)/273004,
```

```
1.2660,.2501,135,
2.3845,.3963,134,
3.1431,.1502,045,
4.2064,.2033,136/
 DATA (PCT(I),I=1007,1022)/275005,
1.2811,.2639,138,
2.0790,.0691,135,
3.2441,.2321,136,
4.3810,.4217,137,
5.0148,.0131,134/
 DATA (PCT(I),I=1023,1029)/277002,
1.7769,.7596,138,
2.2231,.2404,139/
 DATA (PCT(I),I=1030,1039)/279003,
1.6432,.6411,139,
2.3071,.3100,133,
3.0496,.0489,138/
 DATA (PCT(I),I=1040,1046)/282002,
1.8642,.9055,133,
2.1358,.0945,138/
 DATA (PCT(I),I=1047,1053)/284002,
1.6236,.6481,132,
2.3764,.3519,133/
 DATA (PCT(I),I=1054,1060)/286002,
1.8977,.9165,134,
2.1023,.0835,117/
 DATA (PCT(I),I=1061,1073)/287004,
1.1034,.0994,046,
2.3310,.3450,134,
3.2050,.1877,117,
4.3606,.3678,131/
 DATA (PCT(I),I=1074,1080)/290002,
1.5720,.5452,117,
2.4280,.4548,046/
 DATA (PCT(I),I=1081,1087)/291002,
1.3174,.2898,117,
2.6826,.7102,046/
 DATA (PCT(I),I=1088,1100)/293004,
1.2702,.2295,116,
2.6157,.6711,119,
3.0418,.0383,118,
4.0723,.0610,117/
 DATA (PCT(I),I=1101,1107)/294002,
1.7598,.7525,117,
2.2402,.2475,118/
 DATA (PCT(I),I=1108,1120)/297004,
1.1755,.1644,097,
2.1385,.1246,091,
3.4358,.4506,092,
4.2502,.2603,087/
 DATA (PCT(I),I=1121,1130)/298003,
1.8238,.8087,091,
2.0636,.0722,087,
3.1127,.1191,097/
 DATA (PCT(I),I=1131,1137)/299002,
1.6732,.6694,091,
2.3268,.3306,099/
 DATA (PCT(I),I=1138,1147)/301003,
```

```
1.5230,.4867,091,
2.3873,.4271,090,
3.0896,.0862,089/
 DATA (PCT(I),I=1148,1160)/302004,
1.4608,.4698,104,
2.3472,.3430,103,
3.1000,.1008,105,
4.0919,.0864,099/
 DATA (PCT(I),I=1161,1167)/304002,
1.8878,.8921,105,
2.1122,.1079,104/
 DATA (PCT(I),I=1168,1177)/305003,
1.4157,.4234,081,
2.4865,.4670,105,
3.0979,.1096,103/
 DATA (PCT(I),I=1178,1190)/308004,
1.1643,.1593,107,
2.2276,.1992,106,
3.5241,.5578,104,
4.0840,.0837,103/
 DATA (PCT(I),I=1191,1203)/309004,
1.1406,.1331,080,
2.1385,.1293,105,
3.0699,.0685,104,
4.6510,.6692,106/
 DATA (PCT(I),I=1204,1213)/311003,
1.5674,.5728,080,
2.3276,.3419,079,
3.1051,.0852,081/
 DATA (PCT(I),I=1214,1226)/312004,
1.5164,.4677,106,
2.2969,.3537,113,
3.1356,.1335,079,
4.0511,.0451,080/
 DATA (PCT(I),I=1227,1233)/314002,
1.3712,.3543,106,
2.6288,.6457,107/
 DATA (PCT(I),I=1234,1243)/315003,
1.7058,.7286,112,
2.1396,.1133,113,
3.1546,.1581,114/
 DATA (PCT(I),I=1244,1256)/319004,
1.3551,.3058,111,
2.2626,.2722,112,
3.1161,.0934,116,
4.2661,.3286,114/
 DATA (PCT(I),I=1257,1263)/321002,
1.9467,.9564,116,
2.0533,.0436,114/
 DATA (PCT(I),I=1264,1273)/322003,
1.6411,.6753,115,
2.3307,.3004,114,
3.0282,.0243,116/
 DATA (PCT(I),I=1274,1286)/325004,
1.7409,.7645,092,
2.1335,.1201,097,
3.0735,.0673,093,
4.0521,.0480,096/
```

```
      DATA (PCT(I),I=1287,1293)/326002,
     1.9417,.9473,096,
     2.0583,.0527,093/
      DATA (PCT(I),I=1294,1300)/327002,
     1.6133,.5824,097,
     2.3867,.4176,096/
      DATA (PCT(I),I=1301,1310)/330003,
     1.3122,.3172,099,
     2.5388,.5300,098,
     3.1490,.1528,100/
      DATA (PCT(I),I=1311,1320)/332003,
     1.4385,.4240,099,
     2.5087,.5182,103,
     3.0528,.0578,100/
      DATA (PCT(I),I=1321,1327)/336002,
     1.1721,.1654,102,
     2.8279,.8346,103/
      DATA (PCT(I),I=1328,1337)/337003,
     1.4549,.4486,103,
     2.0530,.0494,102,
     3.4921,.5020,107/
      DATA (PCT(I),I=1338,1344)/338002,
     1.8619,.8723,102,
     2.1381,.1277,148/
      DATA (PCT(I),I=1345,1354)/340003,
     1.5976,.5906,108,
     2.2855,.3140,102,
     3.1168,.0954,107/
      DATA (PCT(I),I=1355,1361)/342002,
     1.1959,.1455,107,
     2.8041,.8545,108/
      DATA (PCT(I),I=1362,1371)/345003,
     1.8315,.8529,111,
     2.0549,.0488,110,
     3.1136,.0983,109/
      DATA (PCT(I),I=1372,1381)/347003,
     1.5221,.5475,116,
     2.4313,.4132,111,
     3.0465,.0394,110/
      DATA (PCT(I),I=1382,1388)/349002,
     1.7259,.7186,110,
     2.2741,.2814,109/
      DATA (PCT(I),I=1389,1395)/351002,
     1.8317,.8326,109,
     2.1683,.1674,110/
      DATA (PCT(I),I=1396,1402)/354002,
     1.7997,.7663,119,
     2.2003,.2337,116/
      DATA (PCT(I),I=1403,1418)/355005,
     1.0345,.0310,120,
     2.1357,.1086,127,
     3.1700,.1709,131,
     4.1051,.1006,118,
     5.5547,.5889,119/
      DATA (PCT(I),I=1419,1431)/356004,
     1.7102,.6825,120,
     2.1698,.1731,121,
     3.0204,.0173,127,
```

```
4.0996,.1272,118/
 DATA (PCT(I),I=1432,1438)/358002,
1.4488,.4311,120,
2.5512,.5689,121/
 DATA (PCT(I),I=1439,1445)/360002,
1.6168,.6045,120,
2.3832,.3955,119/
 DATA (PCT(I),I=1446,1455)/363003,
1.2728,.2645,131,
2.2576,.2465,127,
3.4697,.4890,130/
 DATA (PCT(I),I=1456,1465)/364003,
1.7369,.7631,130,
2.1345,.1121,140,
3.1286,.1248,141/
 DATA (PCT(I),I=1466,1484)/368006,
1.0231,.0267,124,
2.0865,.0687,141,
3.4206,.5076,128,
4.3187,.2861,129,
5.0445,.0338,130,
6.1066,.0772,127/
 DATA (PCT(I),I=1485,1497)/369004,
1.1231,.1320,143,
2.0888,.0645,129,
3.4843,.5065,142,
4.3037,.2970,141/
 DATA (PCT(I),I=1498,1504)/371002,
1.7959,.7925,143,
2.2041,.2075,144/
 DATA (PCT(I),I=1505,1511)/375002,
1.8286,.8199,142,
2.1714,.1801,143/
 DATA (PCT(I),I=1512,1521)/378003,
1.5135,.4539,142,
2.3148,.3383,129,
3.1717,.2078,126/
 DATA (PCT(I),I=1522,1528)/383002,
1.2193,.1733,121,
2.7807,.8267,127/
 DATA (PCT(I),I=1529,1550)/384007,
1.1151,.0857,128,
2.1414,.1304,125,
3.5373,.6059,145,
4.0383,.0235,140,
5.1163,.1087,127,
6.0241,.0220,126,
7.0275,.0237,121/
 DATA (PCT(I),I=1551,1563)/386004,
1.4574,.4177,121,
2.1153,.1195,122,
3.3570,.3924,123,
4.0703,.0704,125/
 DATA (PCT(I),I=1564,1570)/390002,
1.7555,.7268,123,
2.2445,.2732,122/
 DATA (PCT(I),I=1571,1589)/392006,
1.1269,.1073,130,
```

```
2.4247,.4481,124,
3.0207,.0182,126,
4.2557,.2479,125,
5.0849,.0858,123,
6.0870,.0927,122/
 DATA (PCT(I),I=1590,1599)/396003,
1.5354,.5510,124,
2.3702,.3649,142,
3.0944,.0841,126/
 DATA (PCT(I),I=1600,1606)/398002,
1.2197,.2306,145,
2.7803,.7694,124/
 DATA (PCT(I),I=1607,1613)/401002,
1.4967,.5001,163,
2.5033,.4999,162/
 DATA (PCT(I),I=1614,1620)/406002,
1.7959,.8606,145,
2.2041,.1394,146/
 DATA (PCT(I),I=1621,1627)/407002,
1.6724,.6798,146,
2.3276,.3202,149/
 DATA (PCT(I),I=1628,1634)/408002,
1.8276,.8177,146,
2.1724,.1823,122/
 DATA (PCT(I),I=1635,1644)/409003,
1.8598,.8839,122,
2.0894,.0730,145,
3.0508,.0431,146/
 DATA (PCT(I),I=1645,1651)/413002,
1.6793,.6738,153,
2.3207,.3262,094/
 DATA (PCT(I),I=1652,1658)/414002,
1.7705,.7648,153,
2.2295,.2352,094/
 DATA (PCT(I),I=1659,1665)/416002,
1.9464,.9532,094,
2.0536,.0468,095/
 DATA (PCT(I),I=1666,1678)/417004,
1.4664,.4519,093,
2.2940,.3020,095,
3.2119,.2182,094,
4.0277,.0279,096/
 DATA (PCT(I),I=1679,1685)/420002,
1.1210,.1282,094,
2.8790,.8718,095/
 DATA (PCT(I),I=1686,1692)/421002,
1.6337,.5811,154,
2.3663,.4189,152/
 DATA (PCT(I),I=1693,1702)/422003,
1.1382,.1449,159,
2.6362,.6432,152,
3.2255,.2120,154/
 DATA (PCT(I),I=1703,1715)/423004,
1.6049,.6192,152,
2.0478,.0378,151,
3.2705,.2863,154,
4.0768,.0566,159/
 DATA (PCT(I),I=1716,1728)/424004,
```

```
1.3261,.3208,152,
2.6196,.6315,159,
3.0339,.0298,154,
4.0204,.0179,151/
 DATA (PCT(I),I=1729,1735)/425002,
1.9613,.9636,159,
2.0387,.0364,154/
 DATA (PCT(I),I=1736,1742)/426002,
1.9035,.9030,095,
2.0965,.0970,100/
 DATA (PCT(I),I=1743,1749)/427002,
1.8642,.8554,150,
2.1358,.1446,100/
 DATA (PCT(I),I=1750,1756)/428002,
1.9456,.9474,150,
2.0544,.0526,101/
 DATA (PCT(I),I=1757,1766)/429003,
1.3479,.3543,150,
2.5512,.5601,151,
3.1009,.0856,152/
 DATA (PCT(I),I=1767,1773)/435002,
1.4720,.4692,161,
2.5280,.5308,151/
 DATA (PCT(I),I=1774,1783)/436003,
1.2229,.2223,151,
2.2267,.1813,161,
3.5504,.5964,149/
 DATA (PCT(I),I=1784,1796)/438004,
1.0959,.0600,147,
2.7394,.7698,148,
3.1576,.1645,149,
4.0071,.0056,150/
 DATA (PCT(I),I=1797,1803)/441002,
1.7372,.7918,147,
2.2628,.2082,148/
 DATA (PCT(I),I=1804,1810)/444002,
1.9635,.9526,147,
2.0365,.0474,149/
 DATA (PCT(I),I=1811,1817)/446002,
1.2406,.2351,154,
2.7594,.7649,155/
 DATA (PCT(I),I=1818,1824)/448002,
1.5388,.5072,154,
2.4612,.4928,156/
 DATA (PCT(I),I=1825,1834)/449003,
1.6649,.6609,156,
2.3008,.3038,154,
3.0343,.0354,157/
 DATA (PCT(I),I=1835,1841)/451002,
1.8391,.8148,155,
2.1609,.1852,157/
 DATA (PCT(I),I=1842,1848)/456002,
1.9564,.9559,157,
2.0436,.0441,156/
 DATA (PCT(I),I=1849,1855)/463002,
1.5363,.5392,170,
2.4637,.4608,169/
 DATA (PCT(I),I=1856,1862)/464002,
```

```
     1.6039,.6894,169,
     2.3961,.3106,168/
      DATA (PCT(I),I=1863,1872)/465003,
     1.3998,.4547,168,
     2.5477,.4849,171,
     3.0525,.0604,169/
      DATA (PCT(I),I=1873,1885)/470004,
     1.3765,.3946,167,
     2.3228,.2667,168,
     3.2727,.3056,166,
     4.0281,.0331,165/
      DATA (PCT(I),I=1886,1892)/471002,
     1.6541,.7100,167,
     2.3459,.2900,171/
      DATA (PCT(I),I=1893,1899)/475002,
     1.0933,.0850,165,
     2.9067,.9150,166/
      DATA (PCT(I),I=1900,1906)/477002,
     1.3499,.4079,166,
     2.6501,.5921,165/
      DATA (PCT(I),I=1907,1913)/490002,
     1.9897,.9874,160,
     2.0103,.0126,161/
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC SPECIFY WHICH MATRIX IS TO BE COMPUTED CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,
     X1400,1500,1600),M
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR HIGH VALUE - LARGE, VOLUME CCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 100  VOL=IR(15)+IR(22)+IR(30)
      CALL ADD(VOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR HIGH VALUE - LARGE, COST CCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 200  CTH=.8817*(-1216.+924.*IR(3)**.421)
      RTH=1349.5251+3.5187860*IR(3)
      CTH=MIN(MAX(CTH,RTH),1.17*RTH)
      CRH=718.+2.96*IR(5)
      CW=296.5+.644*IR(9)
      DOL=IR(15)*CTH+IR(22)*CRH+IR(30)*CW
      CALL ADD(DOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR HIGH VALUE - SMALL, VOLUME CCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 300  VOL=IR(16)+IR(23)+IR(29)
      CALL ADD(VOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR HIGH VALUE - SMALL, COST CCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 400  CTH=.8817*(-1216.+924.*IR(3)**.421)
      RTH=1349.5251+3.5187860*IR(3)
      CTH=MIN(MAX(CTH,RTH),1.17*RTH)
      CRH=718.+2.96*IR(5)
```

```
      CA=10540.+18.92*IR(7)
      DOL=IR(16)*CTH+IR(23)*CRH+IR(29)*CA
      CALL ADD(DOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR MED VALUE - LARGE, VOLUME CCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 500  VOL=IR(17)+IR(24)+IR(31)
      CALL ADD(VOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR MED VALUE - LARGE, COST CCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 600  CTM=.4961*(-1216.+924.*IR(3)**.421)
      RTM=992.43256+3.1580756*IR(3)
      CTM=MIN(MAX(CTM,RTM),1.17*RTM)
      CRM=341.+1.7*IR(5)
      CW=296.5+.644*IR(9)
      DOL=IR(17)*CTM+IR(24)*CRM+IR(31)*CW
      CALL ADD(DOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR MED VALUE - SMALL, VOLUME CCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 700  VOL=IR(18)+IR(25)
      CALL ADD(VOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR MED VALUE - SMALL, COST CCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 800  CTM=.4961*(-1216.+924.*IR(3)**.421)
      RTM=992.43256+3.1580756*IR(3)
      CTM=MIN(MAX(CTM,RTM),1.17*RTM)
      CRM=341.+1.7*IR(5)
      DOL=IR(18)*CTM+IR(25)*CRM
      CALL ADD(DOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR LOW VALUE - LARGE, VOLUME CCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 900  VOL=IR(19)+IR(26)+IR(32)
      CALL ADD(VOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR LOW VALUE - LARGE, COST CCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1000  CTL=.4188*(-1216.+924.*IR(3)**.421)
      RTL=768.85476+2.6590308*IR(3)
      CTL=MIN(MAX(CTL,RTL),1.17*RTL)
      CRL=21.4+1.47*IR(5)
      CW=296.5+.644*IR(9)
      DOL=IR(19)*CTL+IR(26)*CRL+IR(32)*CW
      CALL ADD(DOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR LOW VALUE - SMALL, VOLUME CCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1100  VOL=IR(20)+IR(27)
```

```
      CALL ADD(VOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR LOW VALUE - SMALL, COST CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1200  CTL=.4188*(-1216.+924.*IR(3)**.421)
      RTL=768.85476+2.6590308*IR(3)
      CTL=MIN(MAX(CTL,RTL),1.17*RTL)
      CRL=21.4+1.47*IR(5)
      DOL=IR(20)*CTL+IR(27)*CRL
      CALL ADD(DOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR CRUDE - LARGE, VOLUME CCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1300  VOL=IR(33)+IR(35)
      CALL ADD(VOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR CRUDE - LARGE, COST CCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1400  CPC=25.84+.2324*IR(11)
      CW=296.5+.644*IR(9)
      DOL=IR(33)*CW+IR(35)*CPC
      CALL ADD(DOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR REFINED - LARGE, VOLUME CCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1500  VOL=IR(21)+IR(28)+IR(34)+IR(36)
      CALL ADD(VOL)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC DO COMPUTATION FOR REFINED - LARGE, COST CCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1600  CTL=.4188*(-1216.+924.*IR(3)**.421)
      RTL=768.85476+2.6590308*IR(3)
      CTL=MIN(MAX(CTL,RTL),1.17*RTL)
      CRL=21.4+1.47*IR(5)
      CW=296.5+.644*IR(9)
      CPR=.4148*IR(13)
      DOL=IR(21)*CTL+IR(28)*CRL+IR(34)*CW+IR(36)*CPR
      CALL ADD(DOL)
      RETURN
      SUBROUTINE ADD(X)
      IMPLICIT INTEGER(A)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC ASSIGN TO AO AND AD, RESPECTIVELY, THE VALUES IN THE NDX TABLE CCCCCC
CCCCC WHICH CORRESPOND TO THE ORIGIN AND DESTINATION NNS ZONES (IF CCCCCCCC
CCCCC THE ZONES LIE BETWEEN ZONES 1 AND 512 INCLUSIVE) CCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      AO=IR(1)
      IF (AO.GT.512) GO TO 6000
      IF (AO.LT.1) GO TO 6000
      AO=NDX(AO)
      AD=IR(2)
      IF (AD.GT.512) GO TO 6100
      IF (AD.LT.1) GO TO 6100
```

```
      AD=NDX(AD)
      IF (AO) 2000,6200,1700
1700  IF (AD) 1900,6200,1800
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC IN THE FOLLOWING COMMENT STATEMENTS THE NOTATION WILL BE: CCCCCCCCCCC
CCCCC O=ORIGIN NNS ZONE;   D=DESTINATION NNS ZONE CCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC AGGREGATE FOR CASE OF BOTH O & D ENTIRELY WITHIN SINGLE BEA ZONES CCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1800  BMTX(AO,AD)=BMTX(AO,AD)+X
      GO TO 9999
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC AGGREGATE FOR CASE OF O IN SINGLE BEA ZONE AND D IN MULTIPLE ZONES CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1900  A3=IPCT(-AD)/1000
      IF (A3.NE.IR(2)) GO TO 6400
      A4=MOD(IPCT(-AD),1000)
      DO 1950 K=1,A4
      ADK=IPCT(3*K-AD)
      PCTDK=PCT(3*K-AD-2)
1950  BMTX(AO,ADK)=BMTX(AO,ADK)+PCTDK*X
      GO TO 9999
2000  IF (AD) 2200,6200,2100
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC AGGREGATE FOR CASE OF O IN MULTIPLE ZONES AND D IN SINGLE BEA ZONE CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2100  A1=IPCT(-AO)/1000
      IF (A1.NE.IR(1)) GO TO 6300
      A2=MOD(IPCT(-AO),1000)
      DO 2150 J=1,A2
      AOJ=IPCT(3*J-AO)
      PCTOJ=PCT(3*J-AO-1)
2150  BMTX(AOJ,AD)=BMTX(AOJ,AD)+PCTOJ*X
      GO TO 9999
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC AGGREGATE FOR CASE OF BOTH O & D IN MULTIPLE BEA ZONES CCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2200  A1=IPCT(-AO)/1000
      A3=IPCT(-AD)/1000
      IF (A1.NE.IR(1)) GO TO 6300
      IF (A3.NE.IR(2)) GO TO 6400
      A2=MOD(IPCT(-AO),1000)
      A4=MOD(IPCT(-AD),1000)
      DO 2250 J=1,A2
      AOJ=IPCT(3*J-AO)
      PCTOJ=PCT(3*J-AO-1)
      DO 2250 K=1,A4
      ADK=IPCT(3*K-AD)
      PCTDK=PCT(3*K-AD-2)
2250  BMTX(AOJ,ADK)=BMTX(AOJ,ADK)+PCTOJ*PCTDK*X
      GO TO 9999
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC THE REMAINDER OF THIS PROGRAM CONSISTS OF ERROR MESSAGES CCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
6000  WRITE(6,6001) IR(1),IR(2)
6001  FORMAT(' ERROR,BAD INPUT RECORD. UNKNOWN ORIGIN. IR(1),IR(2):'2
     +I10)
```

```
      GO TO 9999
6100  WRITE(6,6101) IR(1),IR(2)
6101  FORMAT(' ERROR,BAD INPUT RECORD. UNKNOWN DESTINATION. IR(1),IR(
     +2):'2I10)
      GO TO 9999
6200  WRITE(6,6201) IR(1),IR(2),AO,AD
6201  FORMAT(' ERROR,BAD NDX MAPPING.IR(1),IR(2),AO,AD:'4I10)
      GO TO 9999
6300  WRITE(6,6301) IR(1),IR(2),AO,AD,A1,A3
6301  FORMAT('ERROR,BAD PCT MAPPING (A1-IR(1)). IR(1),IR(2),AO,AD,A1,A3:
     +'6I10)
      GO TO 9999
6400  WRITE(6,6401) IR(1),IR(2),AO,AD,A1,A3
6401  FORMAT('ERROR,BAD PCT MAPPING (A3-IR(2)). IR(1),IR(2),AO,AD,A1,A3:
     +'6I10)
      GO TO 9999
9999  RETURN
      END

END ONSITE PRINTOUT ON OCTOBER 6, 1975 AT 13:49:37
BEA*SUB(1).BAG(14)
```

OF COMM. GRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBSIR 75-911	2. Gov't Accession No.	3. Recipient's Accession No.

TLE AND SUBTITLE

A National Network Data Base System

5. Publication Date
September 1975

6. Performing Organization Code

7. AUTHOR(S)
Richard H.F. Jackson

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No.
2050405/6

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address *(Street, City, State, ZIP)*
U. S. Department of Transportation
U. S. Department of Commerce
U. S. Department of the Army

13. Type of Report & Period Covered
Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

This report documents a data base "system" created at the National Bureau of Standards. The National Network Data Base System (NNDBS) provides information on flows of freight and passengers throughout the transportation system of the United States. It consists of a set of FORTRAN programs written for the NBS UNIVAC 1108 (but transportable) and some basic data tapes. In addition to providing basic data on the transportation network, the NNDBS can produce modal splits, aggregations over certain "zones" in the U.S., and is capable of easy extension to other uses. This report is intended as a user's guide and includes discussions of the data tapes and each of the programs. Complete listings and tape formats are also included.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)* Commodity Flows; Data Base; Freight Mode Choice; Market Split; Modal Choice; Modal Freight; National Transport Analysis; Regional Transport Analysis; Transportation Network.

18. AVAILABILITY [✓ X] Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13

☐ Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PAGES
UNCLASSIFIED	60
20. SECURITY CLASS (THIS PAGE)	22. Price
UNCLASSIFIED	

NBSIR 75-912
(This number has been canceled in
order to allow the records of
Office of Tech. Pubs. to be brought
up to date. A new number will be
assigned when the report has been
approved by WERB)