

AD-A085 007

BATTELLE COLUMBUS LABS. ON P/O 1/3
THE GENERAL AVIATION DYNAMICS MODEL VOLUME III. SYSTEMS MANUAL. (U)
JUL 79 M A DUFFY; J M MCCREERY DOT-FA779A-4043

UNCLASSIFIED

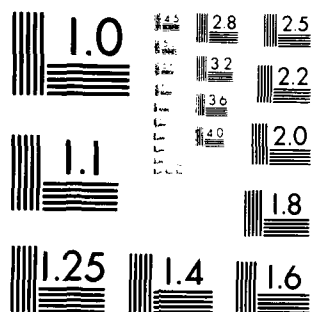
FAA-AVP-79-8-VOL-3

ML

1 of 4

AD
A085007





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Report No. FAA-AVP-79-8

LEVEL

DOT-FA77WA-4043

1072544

THE GENERAL AVIATION DYNAMICS MODEL
Volume III. Systems Manual

ADA085007



July 1979

FINAL REPORT

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION

Office of Aviation Policy
Aviation Forecast Branch
Washington D.C. 20591

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

FILE COPY

80 6 2 024

The contents of this report reflect the views of Battelle's Columbus Laboratories, which is responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policy of the Department of Transportation. This report does not constitute a standard, specification, or regulation.

Accession For	
NTIS GRA&I	☒
DOC TAB	☐
Unannounced	☐
Justification	<i>Per</i>
By _____	
Distribution/	
Availability	
Dist	Available, or special
<i>A</i>	

Distribution Unlimited per Mrs. Margaret Swann, FAA/Library

Technical Report Documentation Page

1. Report No. 19 18 FAA-AVP 79-8 VOL-3 AD-A085007		3. Recipient's Catalog No.	
2. Title and Subtitle 16 The General Aviation Dynamics Model Volume III. Systems Manual.		5. Report Date May 30, 1979	
7. Author(s) 10 M.A. Duffy J.H. McCreery Ph. D.		6. Performing Organization Code	
9. Performing Organization Name and Address Battelle Columbus Laboratories 505 King Avenue Columbus, Ohio 43201		8. Performing Organization Report No.	
12. Sponsoring Agency Name and Address Department of Transportation Federal Aviation Administration Office of Aviation Policy Washington, D.C. 20591 12 312		10. Work Unit No. (TRAIS) 15 Contract or Grant No. DOT-FA77WA-40434 13 Type of Report and Period Covered Final Report. 2 September 2, 1977- 30 May 30, 1979 14 Sponsoring Agency Code DOT/FAA	
15. Supplementary Notes Performed for the Aviation Forecast Branch, Office of Aviation Policy, J.W. Hines, Contract Technical Officer 11 JUL 79			
16. Abstract The model is a dynamic simulation, interactive computer, model built upon the cause-effect interactions displayed between various sectors of the general aviation system. The initial work by Battelle in 1976 was based on data through calendar year 1974 (Report No. FAA-AVP-77-20, General Aviation Dynamics ..., April 1977, three volumes). Under this contract, the model was updated based on data through 1976 and the results were presented in a two volume technical document covering the current model development efforts and all prior work done by Battelle. These volumes, Volume I-Executive Summary and Volume II-Technical Report, were made available to the public through the National Technical Information Service, Springfield, Virginia 22161. Volume III-Systems Manual-provides a thorough description of the computer software aspects of the model. This includes a complete listing of the program, an example of a batch run of the model, a user handbook for the NUCLEUS programming language, and a user's guide for running the model interactively. ←			
17. Key Words Model Update System Dynamics Simulation General Aviation Activity		18. Distribution Statement Upon request only. Contact the Aviation Forecast Branch, AVF-120, Federal Aviation Admin., Wash. DC, 20591, Phone (202) 426-3603	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 311	22. Price

407080

Jm

FINAL REPORT

on

THE
GENERAL AVIATION DYNAMICS
MODEL

VOLUME III. SYSTEMS MANUAL

to the

OFFICE OF AVIATION POLICY
FEDERAL AVIATION ADMINISTRATION

May 30, 1979

by

Jane H. McCreery, Ph.D.

(CONTRACT NO. DOT-FA77WA-4043)

BATTELLE
Columbus Laboratories
505 King Avenue
Columbus, Ohio 43201

TABLE OF CONTENTS

	<u>Page</u>
INTRODUCTION	1
PROGRAM DESCRIPTION	2
APPENDIX A. LISTING OF GAD PROGRAM	
APPENDIX B. A SAMPLE BATCH RUN OF GAD	
APPENDIX C. USER LANGUAGE REFERENCE MANUAL FOR NUCLEUS	
APPENDIX D. USER'S GUIDE	

LIST OF TABLES

TABLE 1. SEGMENT STRUCTURE OF THE GAD SYSTEM	4
TABLE 2. MODELS CONTAINED IN THE SEGMENTS OF THE GAD SYSTEM	7-8

LIST OF FIGURES

FIGURE 1. THE SEGMENT DIAGRAM OF THE GAD SYSTEM	3
FIGURE 2. A PRIMITIVE SEGMENT DIAGRAM	5

FINAL REPORT
on
THE GENERAL AVIATION DYNAMICS MODEL
VOLUME III. SYSTEMS MANUAL

to the
OFFICE OF AVIATION POLICY
FEDERAL AVIATION ADMINISTRATION

- May 30, 1979

by
Jane. H. McCreery, Ph.D.

INTRODUCTION

The General Aviation Dynamics (GAD) model, is implemented in NUCLEUS*, a computer software system developed at Battelle. GAD currently resides on both the Battelle computer and the United Computing Systems (UCS) computer; it can be accessed on either machine via telephone from anywhere in the U.S., using either a remote batch terminal or an interactive terminal. This GAD systems manual contains a complete description of the computer program. Appendix A of this manual consists of a complete listing of the nucleus GAD program, Appendix B

*NUCLEUS: Numerical Classification and EvalUation System.

contains a full batch run of the model, Appendix C is a user handbook for the Nucleus language, and Appendix D is a user's guide for running the GAD model interactively.

PROGRAM DESCRIPTION

NUCLEUS is a Fortran based computer methodology featuring: a structured programming language, dynamic simulation modeling, data base management, report generation, conversational programs and batch or on-line access. The GAD model is a dynamic simulation model written in NUCLEUS and with a conversational dialogue superimposed on it. This conversational dialogue simplifies user interaction with the system. The user is led through the steps of the model, one step at a time, selecting input and output options by answering self-explanatory questions written in English. Thus, even a user unfamiliar with computers should experience no difficulty in using GAD. The user-system interaction introduced by the conversational dialogue also allows direct examination and modification of the input data and assumptions during on-line use of the system. Thus, the effect on the general aviation forecasts of different scenarios may be investigated by the user during a single on-line session with the model.

In more detail, the GAD system is a modularized simulation system with a hierarchical or tree structure imposed on the individual segments of the program. The need for modularization or segmentation arises from the fact that, in its entirety, the system occupies too much computer memory to permit interactive use. With segmentation, only portions or segments of the program are resident in core at any time. The entire program, in the form produced by the NUCLEUS interpreter, resides on a file from which the different segments are loaded into core as required. The segmentation diagram in Figure 1 shows the tree structure of the segments. A description of the function of each segment is given in Table 1. There are ten linked segments organized into three levels of hierarchy or dominance. The linkages between the segments in the diagram reflect decreasing dominance in a top-to-bottom direction.

Level

0

1

2

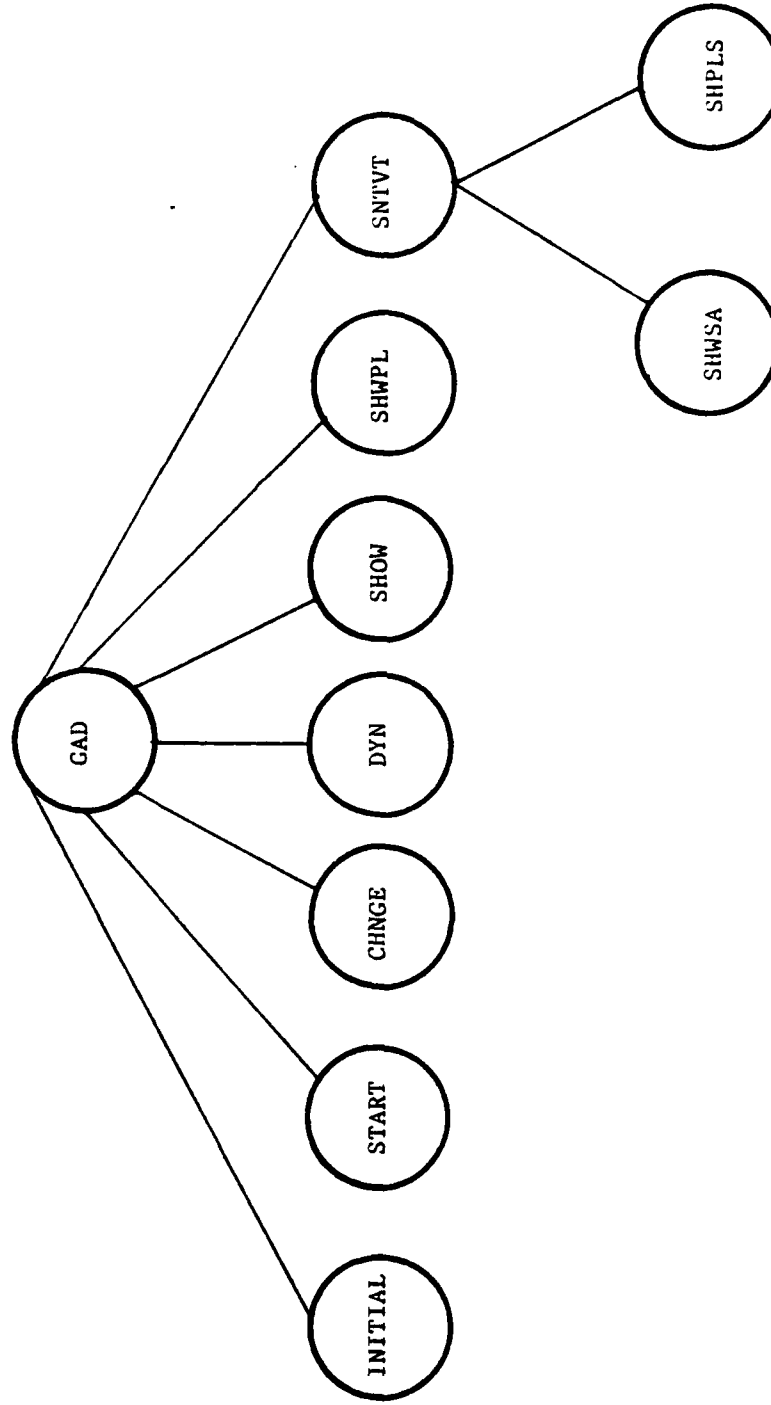


FIGURE 1. THE SEGMENT DIAGRAM OF THE GAD SYSTEM

TABLE 1. SEGMENT STRUCTURE OF THE GAD SYSTEM

Segment	Level	Description
GAD	0	Controls the entire GAD system.
INITIAL	1	Allows the user to select the ending year for the simulation, and to choose between running the simulation with the initial assumptions unchanged or changing the initial assumptions of the forecast.
START	1	Writes the initial assumptions for data out to the data base.
CHNGE	1	Allows the user to change the initial assumptions for the exogenous data.
DYN	1	The dynamic simulation segment of the model.
SHOW	1	Allows the user to select output tables to be printed.
SHWPL	1	Allows the user to select plots of the results of the forecast to be displayed.
SNTVT	1	Controls the display of sensitivity results for comparing one simulation run with another baseline simulation.
SHWSA	2	Allows the user to select sensitivity tables to be printed.
SHPLS	2	Allows the user to select sensitivity plots to be displayed.

For example the master control segment GAD (at the top of the diagram) has dominance over all other segments of the system. Information, available or computed, in the GAD segment is available to all the other segments. Segment GAD is resident in core for the entire period of user interaction with the GAD system. Segments SHWSA and SHPLS at level 2 are dominated by SNTVT at level 1 and GAD at level 0. Only one of SHWSA and SHPLS may be in core at any one time; if either one is in core, then SNTVT and GAD will also be resident in core.

To explain further the concept of segment hierarchy, Figure 2 shows a primitive segment diagram. In the context of this diagram the following general statements apply:

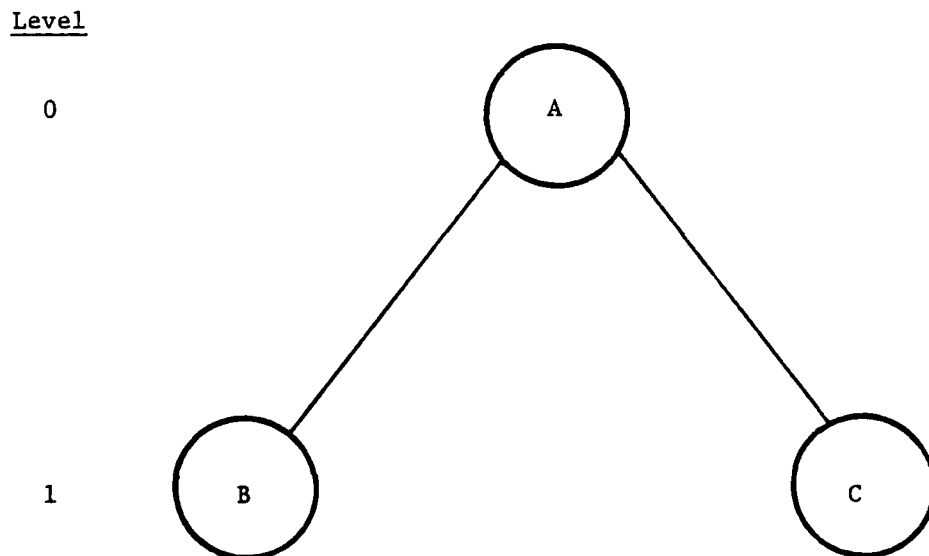


FIGURE 2. A PRIMITIVE SEGMENT DIAGRAM

- A segment may or may not be dominated by another segment. For example, segments B and C are dominated by A, but B does not dominate C, and C does not dominate B.
- If segment B is dominated by A, then segment A cannot be dominated by B.
- If segment A dominates segment B, then all information in segment A is available to segment B.
- The information in segment B is not available to segment C, and vice versa, except for any overlap through segment A. There is no information flow between segments B and C, except through the dominant segment A.
- From the user standpoint, the information in segment B is available only through segment A. Similarly, the information in segment C is available only through segment A.
- Operationally, segment dominance is manifested in terms of the following computer memory (core) configurations: segment A may exist in core alone; segment B can exist in core only with segment A and in the order "A followed by B"; segment C can exist in core only with segment A and in the order "A followed by C"; segments B and C cannot coexist in core. Thus the only possible core configurations are: A, A+B, and A+C.

Appendix A contains the full Nucleus listing of the GAD program. In this program, the segments given above are further structured into models which are groups of equations and instructions that perform one or more functions. Table 2 lists the models contained with the different segments and gives a brief description of their functions. All the models in the system, with the exception of DYNAM, are static, performing functions that do not vary with time. DYNAM is a

TABLE 2. MODELS CONTAINED IN THE SEGMENTS OF THE GAD SYSTEM

Segment	Level	Description
GAD	MAIN	This is the model that controls execution of the entire GAD system.
INITIAL	ASKAGAIN	Allows the user to view the different steps involved in the system and to choose a forecast using the initial assumptions unchanged or a forecast in which the initial assumptions are modified from the base line.
	ENYR	Allows the user to select the ending year for the simulation.
	FIRST	Prints the heading at the start of the GAD run and, if one simulation has already been run, permits the user the option of continuing with another simulation.
START	BUILD	Writes the initial assumptions for the exogenous data on the data base.
CHNGE	PRCH	Allows the user to change the initial assumptions for the pilot variables.
	CHVAR	Allows the user to change the values of the economic variables yearly at a constant rate, or to enter them year by year.
	ECH	Allows the user to change the initial assumptions for the economic data.
	YRCHK	Allows the user to select the year in which changes to the initial assumptions for cost variables are to begin.
	FOOST	Allows the user to change the initial assumptions for the fixed cost.
	VCOST	Allows the user to change the initial assumptions for the variable cost.

TABLE 2. (Continued)

Segment	Level	Description
	CHANGE	Requests the user to select the variables prior to calling the CHANGE model.
DYN	GADYN	Contains the equations for the dynamic simulation.
	DYNAM	Control model for GADYN.
SHOW	PRINT	Allows the user to select variables to be tabulated.
SHWPL	SELC	Requests the user to select an aircraft user category for a plot of a multi-dimensioned variable.
	SELT	Requests the user to select an aircraft type for a plot of a multi-dimensioned variable.
	XAXIS	Allows the user to select the independent variable for a plot.
	PLOT	Allows the user to select the dependent variable for a plot.
SNTVT	LOADD	Controls the output display of sensitivity variables.
SHWSA	PRINS	Allows the user to select sensitivity variables to be tabulated.
SHPLS	SELC	Requests the user to select an aircraft user category for a sensitivity plot of a multi-dimensioned variable.
	SELT	Requests the user to select an aircraft type for a sensitivity plot of a multi-dimensioned variable.
	PLOTS	Allows the user to select a sensitivity variable to be plotted.

dynamic model containing equations that describe the continuous evaluation through time of interacting, time-dependent variables. It is the model DYNAM that generates the simulation results through time.

Appendix B contains a sample batch run of the GAD system including a baseline simulation, a simulation based on an increase in the federal fuel tax commencing in 1979, and sensitivity results comparing the results of the two sets of forecasts. The data cards required to run the simulations are simply the responses to the questions that are printed. Appendix C contains a description of the Nucleus language and details of the commands available in Nucleus. Appendix D is a user's guide to exercising the GAD model on either the Battelle or UCS interactive computer systems.

APPENDIX A. LISTING OF GAD PROGRAM

```

000017 SYSTEM WIDTH=131, INITIAL=OFF
COMMENT
000017 COMMENT
000027 START(GENERAL AVIATION DYNAMICS MODEL)
000027 OPEN(1,41,22000)
COMMENT
000027 COMMENT
000027 INDEX
000027 AGE(13)
000045 AOPS(3)
000063 CATEG(7)
000105 DUMIND(1)
000121 FUEL(2)
000138 TYPE(7)
000160 YR1(1), TIME(1977,1987)
000221 YR2(1), TIME(1975,1987)
000266 END INDEX
COMMENT
000266 COMMENT
000266 DATA
000266 AGATH, (CONTINUATION FLAG), VALUE=0
000304 ENSIM, (SIMULATION ENDING YEAR), VALUE=0
000323 IDASE, (FLAG FOR BASELINE INITIAL ASSUMPTIONS), VALUE=1
000345 ITECH, (FLAG FOR TEACH STEPS), VALUE=0
000363 ISENS, (SENSITIVITY FLAG), VALUE=0
000401 IVAR, (IMO VARIABLE PLOT FLAG), VALUE=0
000420 NSIM, (SIMULATION PERIOD IN YEARS), VALUE=0
000440 NSIMB, (SIMULATION PERIOD FOR BASELINE), VALUE=0
000460 SENS1, (FILE NUMBER FOR BASELINE DATA), VALUE=2
000500 SENS2, (FILE NUMBER FOR SENSITIVITY DATA), VALUE=2
000521 TVAR, (TIME BASE), VALUE=1
000535 TIMTIME, (TIME BASE), VALUE=1
000551 TIM1, (TIME BASE), VALUE=2
000565 TIM2, (TIME BASE), VALUE=3
000601 END

```

```

000601 MODEL MAIN
000602 TIME(1.1977,1987)
000603 COMPUTE TIME=0
000604 TVAR=1
000605 LOAD INITIAL,XEQ=FIRST
000606 LOAD START,XEQ=BUILD
000607 IF IBASE EQ 0
000608 IF ITECH EQ 1
000609 SHOW TEXT=LEFT/(STEP 2 -- DISPLAY AND/OR CHANGE INITIAL ASSUMPTIONS.)/)
000610 END IF ITECH
000611 SYSTEM HEADING=OFF
000612 LOAD CHANGE,XEQ=CHXEQ
000613 SYSTEM HEADING=ON
000614 END IF IBASE
000615 IF ITECH EQ 1
000616 SHOW TEXT=LEFT/(STEP 3 -- THE FORECAST OF GENERAL AVIATION ACTIVITY IS BEING CO
000617 MPUTED.)/)
000618 END IF ITECH
000619 LOAD DYN,XEQ=DYNAM
000620 IF ITECH EQ 1
000621 SHOW TEXT=LEFT/(STEP 4 -- PRINT TABLES OF RESULTS OF THE FORECAST.)/)
000622 END IF ITECH
000623 ASK(00 YOU WANT TO SEE TABLES OF RESULTS OF THE FORECAST) NO
000624 ELSE YES
000625 LOAD SHOW,XEQ=PRINT
000626 END ASK
000627 IF ITECH EQ 1
000628 SHOW TEXT=LEFT/(STEP 5 -- PLOT THE RESULTS OF THE FORECAST.)/)
000629 END IF ITECH
000630 ASK(00 YOU WANT TO SEE PLOTS OF RESULTS OF THE FORECAST) NO
000631 ELSE YES
000632 LOAD SIMPL,XEQ=PLOT
000633 END ASK
000634 IF ISENS EQ 1
000635 IF ITECH EQ 1
000636 SHOW TEXT=LEFT/(STEP 6 -- COMPARE THE RESULTS OF THE PRESENT FORECAST TO THOSE
000637 OF A PREVIOUS FORECAST FOR SENSITIVITY ANALYSIS.)/)
000638 END IF ITECH
000639 ASK(00 YOU WANT SENSITIVITY ANALYSIS, (THE PREVIOUSLY SAVED FORECAST IS THE BASE
000640 LINE), (YES OR NO)) NO
000641 ELSE YES
000642 LOAD SNTVT,XEQ=LOAD0
000643 END ASK
000644 END IF ISENS
000645 AGAIN=1
000646 IF ITECH EQ 1
000647 SHOW TEXT=LEFT/(STEP 9 -- SAVE THE RESULTS OF THIS FORECAST FOR FUTURE)/,
000648 SENSITIVITY ANALYSIS.)/)
000649 END IF ITECH
000650 ASK(WOULD YOU LIKE TO SAVE THE RESULTS OF THIS SESSION FOR LATER SENSITIVITY ANA
000651 LYSIS) NO
000652 ISENS=0
000653 ELSE YES
000654 MSHD=MSH
000655 ISENS=1
000656 SENS1=SENS2
000657

```

001513 SENS2=3
001524 IF SENS1 EQ SENS2
001540 SENS2=2
001551 END IF SENS1
001552 END ASK
001553 MAIN
001556 END MODEL MAIN

```

001557 DATA SEG1
001573 MODEL ASKAGAIN
001601 ASK (STEP 1 -- WOULD YOU LIKE TO COMPUTE THE FORECAST WITH THE INITIAL ASSUMPTIO
001601 NS UNCHANGED (YES OR NO) OR VIEW THE STEPS OF THIS MODEL (TEACH)) YES
001645 IBASE=1
001656 ELSE TEACH
001662 ITECH=1
001673 SHOW TEXT=LEFT((STEP 1 -- COMPUTE THE FORECAST USING THE INITIAL ASSUMPTIONS U
          NCHANGED.)/,
STEP 2 -- DISPLAY AND/OR CHANGE INITIAL ASSUMPTIONS.)/,
STEP 3 -- COMPUTE THE FORECAST OF GENERAL AVIATION ACTIVITY.)/,
STEP 4 -- PRINT TABLES OF RESULTS OF THE FORECAST.)/,
STEP 5 -- PLOT THE RESULTS OF THE FORECAST.)/,
STEP 6 -- COMPARE THE RESULTS OF THE PRESENT FORECAST TO THOSE)/,
          ( OF A PREVIOUS FORECAST FOR SENSITIVITY ANALYSIS.)/,
STEP 7 -- PRINT TABLES FOR SENSITIVITY ANALYSIS.)/,
STEP 8 -- PLOT THE RESULTS OF SENSITIVITY ANALYSIS.)/,
STEP 9 -- SAVE THE RESULTS OF THIS FORECAST FOR FUTURE)/,
          ( SENSITIVITY ANALYSIS.)/)/)
002134 ASKAGAIN
002137 ELSE NO
002143 IBASE=0
002154 END ASK
002155 END ASKAGAIN
002156 COMMENT
002157 COMMENT
002158 MODEL ENVR
002164 READ ENSIM
002173 IF (ENSIM LT 1977) OR (ENSIM GT 1987)
002225 TELL (THE YEAR MUST BE BETWEEN 1977 AND 1987. ENTER NEW VALUE)
002243 ENVR
002246 END IF
002247 MSIM=ENSIM-1976
002263 SET VRI(1-MSIM)
002274 END MODEL ENVR
002275 COMMENT
002276 COMMENT
002277 COMMENT
002278 MODEL FIRST
002279 IF AGAIN EQ 1
002303 ASK (WOULD YOU LIKE TO CONTINUE WITH ANOTHER FORECAST) YES
002341 ELSE NO
002345 TELL (YOU ARE NOW LEAVING THE GENERAL AVIATION DYNAMICS MODEL.)
002363 STOP
002364 END ASK
002355 END IF
002366 IF AGAIN EQ 0
002402 SHOW TEXT=LEFT((YOU ARE ENTERING THE GENERAL AVIATION DYNAMICS MODEL)/,
          (CREATED AT DANIELLE COLUMBUS LABORATORIES.)/,
          (WRITTEN IN THE MODELING LANGUAGE M U C L E U S .)/,
          (IN THIS SESSION YOU WILL PROJECT CERTAIN LEVELS OF)/,
          (GENERAL AVIATION ACTIVITY FOR THE YEARS 1977 TO 1987.)/)
002512 END IF
002513 TELL (ENTER ENDING YEAR FOR SIMULATION)
002524 ENVR

```

002527 17ECM-8
002540 43K6GATM
002543 CMO MODEL FIRST
002544 SAVE INITIAL.DATA-SEG1

DATA	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8	SEG9	SEG10	SEG11	SEG12	SEG13	SEG14	SEG15	SEG16	SEG17	SEG18	SEG19	SEG20	SEG21	SEG22	SEG23	SEG24	SEG25	SEG26	SEG27	SEG28	SEG29	SEG30	SEG31	SEG32	SEG33	SEG34	SEG35	SEG36	SEG37	SEG38	SEG39	SEG40	SEG41	SEG42	SEG43	SEG44	SEG45	SEG46	SEG47	SEG48	SEG49	SEG50	SEG51	SEG52	SEG53	SEG54	SEG55	SEG56	SEG57	SEG58	SEG59	SEG60	SEG61	SEG62	SEG63	SEG64	SEG65	SEG66	SEG67	SEG68	SEG69	SEG70	SEG71	SEG72	SEG73	SEG74	SEG75	SEG76	SEG77	SEG78	SEG79	SEG80	SEG81	SEG82	SEG83	SEG84	SEG85	SEG86	SEG87	SEG88	SEG89	SEG90	SEG91	SEG92	SEG93	SEG94	SEG95	SEG96	SEG97	SEG98	SEG99	SEG100	SEG101	SEG102	SEG103	SEG104	SEG105	SEG106	SEG107	SEG108	SEG109	SEG110	SEG111	SEG112	SEG113	SEG114	SEG115	SEG116	SEG117	SEG118	SEG119	SEG120	SEG121	SEG122	SEG123	SEG124	SEG125	SEG126	SEG127	SEG128	SEG129	SEG130	SEG131	SEG132	SEG133	SEG134	SEG135	SEG136	SEG137	SEG138	SEG139	SEG140	SEG141	SEG142	SEG143	SEG144	SEG145	SEG146	SEG147	SEG148	SEG149	SEG150	SEG151	SEG152	SEG153	SEG154	SEG155	SEG156	SEG157	SEG158	SEG159	SEG160	SEG161	SEG162	SEG163	SEG164	SEG165	SEG166	SEG167	SEG168	SEG169	SEG170	SEG171	SEG172	SEG173	SEG174	SEG175	SEG176	SEG177	SEG178	SEG179	SEG180	SEG181	SEG182	SEG183	SEG184	SEG185	SEG186	SEG187	SEG188	SEG189	SEG190	SEG191	SEG192	SEG193	SEG194	SEG195	SEG196	SEG197	SEG198	SEG199	SEG200	SEG201	SEG202	SEG203	SEG204	SEG205	SEG206	SEG207	SEG208	SEG209	SEG210	SEG211	SEG212	SEG213	SEG214	SEG215	SEG216	SEG217	SEG218	SEG219	SEG220	SEG221	SEG222	SEG223	SEG224	SEG225	SEG226	SEG227	SEG228	SEG229	SEG230	SEG231	SEG232	SEG233	SEG234	SEG235	SEG236	SEG237	SEG238	SEG239	SEG240	SEG241	SEG242	SEG243	SEG244	SEG245	SEG246	SEG247	SEG248	SEG249	SEG250	SEG251	SEG252	SEG253	SEG254	SEG255	SEG256	SEG257	SEG258	SEG259	SEG260	SEG261	SEG262	SEG263	SEG264	SEG265	SEG266	SEG267	SEG268	SEG269	SEG270	SEG271	SEG272	SEG273	SEG274	SEG275	SEG276	SEG277	SEG278	SEG279	SEG280	SEG281	SEG282	SEG283	SEG284	SEG285	SEG286	SEG287	SEG288	SEG289	SEG290	SEG291	SEG292	SEG293	SEG294	SEG295	SEG296	SEG297	SEG298	SEG299	SEG300	SEG301	SEG302	SEG303	SEG304	SEG305	SEG306	SEG307	SEG308	SEG309	SEG310	SEG311	SEG312	SEG313	SEG314	SEG315	SEG316	SEG317	SEG318	SEG319	SEG320	SEG321	SEG322	SEG323	SEG324	SEG325	SEG326	SEG327	SEG328	SEG329	SEG330	SEG331	SEG332	SEG333	SEG334	SEG335	SEG336	SEG337	SEG338	SEG339	SEG340	SEG341	SEG342	SEG343	SEG344	SEG345	SEG346	SEG347	SEG348	SEG349	SEG350	SEG351	SEG352	SEG353	SEG354	SEG355	SEG356	SEG357	SEG358	SEG359	SEG360	SEG361	SEG362	SEG363	SEG364	SEG365	SEG366	SEG367	SEG368	SEG369	SEG370	SEG371	SEG372	SEG373	SEG374	SEG375	SEG376	SEG377	SEG378	SEG379	SEG380	SEG381	SEG382	SEG383	SEG384	SEG385	SEG386	SEG387	SEG388	SEG389	SEG390	SEG391	SEG392	SEG393	SEG394	SEG395	SEG396	SEG397	SEG398	SEG399	SEG400	SEG401	SEG402	SEG403	SEG404	SEG405	SEG406	SEG407	SEG408	SEG409	SEG410	SEG411	SEG412	SEG413	SEG414	SEG415	SEG416	SEG417	SEG418	SEG41

[illegible]

003655 OPEN(2,52,3843)
003655 OPEN(3,52,3843)
003655 ADD(2,11,SIZE(13))
003655 ADD(2,21,SIZE(13))
003655 ADD(2,41,SIZE(17))
003655 ADD(2,51,SIZE(17,7))
003655 ADD(2,61,SIZE(17,11))
003655 ADD(2,71,SIZE(17,7))
003655 ADD(2,91,SIZE(17,11))
003655 ADD(2,151,SIZE(17))
003655 ADD(2,161,SIZE(13,11))
003655 ADD(2,171,SIZE(17))
003655 ADD(2,181,SIZE(17))
003655 ADD(2,241,SIZE(17,7,11))
003655 ADD(2,251,SIZE(17,11))
003655 ADD(2,261,SIZE(17,11))
003655 ADD(2,271,SIZE(17,7,11))
003655 ADD(2,281,SIZE(17,7,11))
003655 ADD(2,291,SIZE(11))
003655 ADD(2,301,SIZE(17,7,11))
003655 ADD(2,311,SIZE(13,11))
003655 ADD(2,321,SIZE(11))
003655 ADD(2,331,SIZE(11))
003655 ADD(2,341,SIZE(11))
003655 ADD(2,351,SIZE(12,11))
003655 ADD(2,361,SIZE(11))
003655 ADD(2,371,SIZE(11,11))
003655 ADD(2,381,SIZE(12,11))
003655 ADD(2,391,SIZE(17,7,11))
003655 ADD(2,401,SIZE(11))
003655 ADD(2,411,SIZE(11))
003655 ADD(2,421,SIZE(11))
003655 ADD(2,431,SIZE(11))
003655 ADD(2,441,SIZE(11))
003655 ADD(2,451,SIZE(13,11))
003655 ADD(2,461,SIZE(11))
003655 ADD(2,471,SIZE(11))
003655 ADD(2,481,SIZE(11))
003655 ADD(2,491,SIZE(11))
003655 ADD(2,501,SIZE(11))
003655 ADD(2,511,SIZE(11))
003655 ADD(2,521,SIZE(11))
003655 ADD(3,11,SIZE(13))
003655 ADD(3,21,SIZE(11))
003655 ADD(3,41,SIZE(17))
003655 ADD(3,61,SIZE(17,11))
003655 ADD(3,71,SIZE(17,7))
003655 ADD(3,91,SIZE(17,11))
003655 ADD(3,151,SIZE(17))
003655 ADD(3,161,SIZE(13,11))
003655 ADD(3,171,SIZE(17))
003655 ADD(3,181,SIZE(17))
003655 ADD(3,241,SIZE(17,7,11))
003655 ADD(3,251,SIZE(17,11))
003655 ADD(3,261,SIZE(17,11))

```

003655      A0013,27),SIZE(7,7,11)
003655      A0013,28),SIZE(7,7,11)
003655      A0013,29),SIZE(11)
003655      A0013,30),SIZE(7,7,11)
003655      A0013,31),SIZE(3,11)
003655      A0013,32),SIZE(11)
003655      A0013,33),SIZE(11)
003655      A0013,34),SIZE(11)
003655      A0013,35),SIZE(2,11)
003655      A0013,36),SIZE(11)
003655      A0013,37),SIZE(1,11)
003655      A0013,38),SIZE(2,11)
003655      A0013,39),SIZE(7,7,11)
003655      A0013,40),SIZE(11)
003655      A0013,41),SIZE(11)
003655      A0013,42),SIZE(11)
003655      A0013,43),SIZE(11)
003655      A0013,44),SIZE(11)
003655      A0013,45),SIZE(3,11)
003655      A0013,46),SIZE(11)
003655      A0013,47),SIZE(11)
003655      A0013,48),SIZE(11)
003655      A0013,49),SIZE(11)
003655      A0013,50),SIZE(11)
003655      A0013,51),SIZE(11)
003655      A0013,52),SIZE(11)
003655      *
003655      *
003655      MODEL BUILD
003655      IF SENSE EQ 2
003677      SYSTEM DB3=2
003703      END IF
003704      IF SENSE EQ 3
003723      SYSTEM DB3=3
003724      END IF
003725      WRITE AA,IMS(3,30,BASE(1,1,TVAR))
003746      WRITE ADRN,IMS(3,4,BASE(1,TVAR))
003765      WRITE ATP,IMS(3,52,BASE(TVAR))
003800      WRITE ATPDN,IMS(3,6,TRANSPOSE(,YR1),BASE(1,1))
004025      WRITE AURN,IMS(3,5,BASE(1,1,TVAR))
004046      WRITE CP,IMS(3,32,BASE(TVAR))
004063      WRITE CPCH,IMS(3,6,TRANSPOSE(,YR1),BASE(2,1))
004106      WRITE DAUR,IMS(3,7,BASE(1,1,TVAR))
004127      WRITE DPLYR,IMS(3,33)
004142      WRITE FCINP,IMS(3,25,TRANSPOSE(TYPE,*))
004160      WRITE FCGNP,IMS(3,18)
004173      WRITE FGINF,IMS(3,34)
004206      WRITE FHRF,IMS(3,2)
004221      WRITE FTXYR,IMS(3,38)
004234      WRITE GMPYR,IMS(3,36)
004247      WRITE HP,IMS(3,41,BASE(TVAR))
004264      WRITE HR,IMS(3,42,BASE(TVAR))
004301      WRITE IP,IMS(3,4),BASE(TVAR))
004316      WRITE IPCH,IMS(3,6,TRANSPOSE(,YR1),BASE(3,1))
004341      WRITE PCIN,IMS(3,6,TRANSPOSE(,YR1),BASE(4,1))
004364      WRITE POPYR,IMS(3,45)

```

```

004377 WRITE PP,TMS(3,46,BASE(TVARI))
004414 WRITE PPDN,TMS(3,6,TRANSPPOSE(*,YR1),BASE(5,11))
004437 WRITE RADYR,TMS(3,1)
004452 WRITE SCIX,TMS(3,16)
004465 WRITE SFC,TMS(3,17)
004500 WRITE SP,TMS(3,48,BASE(TVARI))
004515 WRITE SPDN,TMS(3,6,TRANSPPOSE(*,YR1),BASE(6,11))
004541 WRITE URIPN,TMS(3,6,TRANSPPOSE(*,YR1),BASE(7,11))
004563 WRITE VCINP,TMS(3,15)
004576 WRITE VCINP,TMS(3,26,TRANSPPOSE(TYPE,*))
004614 WRITE VCINF,TMS(3,51)
004627 IF IBASE NE 0
004643 VCVR=VCINP*VCINF
004667 WRITE VCVR,TMS(3,26)
004702 FCVR=FCINP*FCINF
004726 WRITE FCVR,TMS(3,25)
004741 END IF IBASE
004742 END MODEL BUILD
004743 SAVE START,DATA=SEG1

```

(SEGMENT CHNGE)

001557	DATA SEG1
001573	INDEX
001573	DISP7(7)
001615	DISP3(3)
001633	END INDEX
001633	READ CATEG(1,25)
DATA	BUSINESS
DATA	CORPORATE
DATA	PERSONAL
DATA	AERIAL
DATA	INSTRUCTIONAL
DATA	ATR TAXI
DATA	OTHER
001677	READ CATEG(1,6,6)
DATA	RUSH
DATA	CORP
DATA	PERSNL
DATA	AERIAL
DATA	INSTR
DATA	AIP TX
DATA	OTHER
001716	READ DISP7(1,5,5)
DATA	ATPON
DATA	CPON
DATA	IPON
DATA	PCIN
DATA	PPON
DATA	SPON
DATA	URIPN
001726	READ DISP3(1,3,3)
DATA	DPI
DATA	GNP
DATA	RAD
001732	READ FUEL(1,15)
DATA	AV GAS
DATA	JET FUEL
001741	READ FUEL (1,8,8)
DATA	AV GAS
DATA	JET FUEL
001746	READ TYPE(1,35)
DATA	SNG-ENG. P.-NON-AER
DATA	SNG-ENG. P.-AER.
DATA	MULTI-ENG. P.
DATA	TURROPROP
DATA	TURROJET
DATA	P. ENG.- HELICOPTER
DATA	F. ENG.- HELICOPTER
002030	READ TYPE (1,14,7)
DATA	SINGL-INON-AER
DATA	SINGL-F AER
DATA	MULTI- PISTON
DATA	TURBOC PROP
DATA	TURBOC JET
DATA	PISTON HELIC
DATA	TURBINE HELIC
002065	READ AGE(1,5)

DATA 16-24
DATA 25-34
DATA 35
002071 READ AGE(11,5,5)
DATA 16-24
DATA 25-34
DATA 35

DATA

002075 ADM(CATEG), (AIRCRAFT DESTRUCTION RATE, NORMALIZED (AC/YR)), TMS(3,4)

002075 AITYPE), (INITIAL ANNUALIZED INVESTMENT), VALUE=0

002127 ALPHA(TYPE), (ALPHA), VALUE=0

002156 ATON(YR1), (AIRLINE TRANSPORT PILOT DEPARTURE RATE, NORMALIZED), TMS(3,6,

TRANSP(, YR1), BASE(1,1))

002200 CPON(YR1), (COMMERCIAL PILOT DEPARTURE RATE, NORMALIZED), TMS(3,6,

TRANSP(, YR1), BASE(2,1))

002243 DATI(TYPE), (DELTA AT), VALUE=0

002305 DEF1, (DEFL), VALUE=0.7037

002330 DEF2(YR1), (DEFL72), VALUE=0

002343 DELVC(TYPE, YR1), (VC INCREMENT), VALUE=0

002372 DEPREC(TYPE), (DEPRECIATION PERIOD, IN YEARS), VALUE=0

002525 OPTI(YR1), (DISPOSABLE PERSONAL INCOME (1972 \$, 1972=1)), TMS(3,33)

002554 DUM, (A DUMMY VARIABLE)

002606 DUM7(TYPE), (DUMMY VARIABLE FOR READ)

002623 DUMYR(YR1), (A DUMMY INDEXED DATA SET)

002642 ETAS(01SP3, YR1), (ECONOMIC DATA)

002661 FAD(TYPE, YR1), (FUEL AND OIL COST)

002677 FAD7(TYPE), (FUEL AND OIL COST FOR 1977), VALUE=0

002716 FC, (FIXED COST INDEX (1972 \$, 1972=1)). ONE COMPONENT OF FC IS THE ANNUALIZED INV

ESTIMET)

002745 FCIMP(TYPE), (INITIAL ASSUMPTIONS FOR FIXED COST), TMS(3,18)

002777 FCINF(YR1), (FIXED COST INFLATION FACTOR), TMS(3,34)

003027 FCYR(TYPE, YR1), (FIXED COST), TMS(3,25)

003056 FHPR, (FLYING HOURS REQUIRED FACTOR), TMS(3,2)

003102 F172(TYPE), (1972 FIXED COST), VALUE=0

003130 F172(FUEL, YR1), (FEDERAL FUEL TAX (\$/GAL)), TMS(3,30)

003154 F172(FUEL), (INITIAL ASSUMPTIONS FOR FEDERAL FUEL TAX (\$/GAL)), VALUE=0.07

003203 GNPI(YR1), (GROSS NATIONAL PRODUCT (1972 \$, 1972=1)), TMS(3,36)

003232 HPOPX(TYPE), (TEMPORARY HOURS PER OPERATION), VALUE=0

003263 INDIR*, (AN INDIRECT DATA SET)

003312 IPON(YR1), (INSTRUMENT PILOT DEPARTURE RATE, NORMALIZED), TMS(3,6,

TRANSP(, YR1), BASE(3,1))

003371 ISFC, (SFC CHANGE FLAG), VALUE=0

003406 LFEI(TYPE, YR1), (LANDING FEE (\$/LANDING)), TMS(3,9)

003435 NEWLF(TYPE), (SCRATCH DATASET FOR NEW VALUES FOR LANDING FEE)

003461 NEWXF(FUEL), (SCRATCH DATASET FOR NEW VALUES FOR FUEL TAX)

003504 NEWVC(TYPE), (SCRATCH DATASET FOR NEW VALUES FOR VARIABLE COST)

003530 ASIM1, (ONE LESS THAN ENDING YEAR)

003546 MYEAR, (1 YEAR TO BEGIN CHANGES IN VARIABLE COST DATA)

003570 OSFC(TYPE), (OLD SFC), VALUE=0

003613 PCINI(YR1), (PRIVATE CERTIFICATES ISSUED RATE, NORMALIZED), TMS(3,6,

TRANSP(, YR1), BASE(4,1))

003655 PPOH(YR1), (PRIVATE PILOTS DEPARTURE RATE, NORMALIZED), TMS(3,6,

TRANSP(, YR1), BASE(5,1))

003717 PRATE(DISP7, YR1), (INITIAL PILOT RATES), TMS(3,6)

003745 RAD(YR1), (REVENUE AIRCRAFT DEPARTURES (1972=1)), TMS(3,1, BASE(3))

004001 RESID(TYPE), (DEPRECIATION RESIDUAL), VALUE=0

004027 SCX(AGE1, YR1), (STUDENT CERTIFICATES ISSUED MULTIPLIER, TMS(3,16)

SFCI(TYPE), (SPECIFIC FUEL CONSUMPTION), TMS(3,17)

004061 SPONI(YR1), (STUDENT PILOTS DEPARTURE RATE, NORMALIZED), TMS(3,6,

TRANSP(, YR1), BASE(6,1))

004107 UPGRAD(YR1), (UPGRADE TO INSTRUMENT PILOT RATE, NORMALIZED), TMS(3,6,

TRANSP(, YR1), BASE(7,1))

004213 VC, (VARIABLE COST INDEX (\$/HR), (1972 \$, 1972=1)). COMPONENTS OF VC ARE FTAX (FED


```

004507 MODEL PRCH
004515 DROP*
004517 ASKENTER NAME OF PILOT VARIABLE TO BE CHANGED OR LIST OR NONE) NONE
004543 ELSE LIST
004547 TELL DATA(ATPON,CPON,IPON,PCIN,PPON,SPON,URIPN)
004563 PRCH
004563 ELSE DATA=INDIR(ATPON,CPON,IPON,PCIN,PPON,SPON,URIPN)
004577 TELL CENTER VALUES FOR EACH YEAR )
004607 READ INDIR
004616 PUT THS(INDIR)
004623 PRCH
004626 END ASK
004627 END MODEL PRCH
*
COMMENT
COMMENT
004630 MODEL CHVAR
004636 ASK(should you like the values to change yearly at a constant rate) YES
004663 TELL CENTER RATE OF CHANGE)
004671 READ DUM
004700 DUM=DUM+1
004714 TELL CENTER 1977 VALUE FOR DATA)
004723 SET YR(1)
004731 READ DUMYR
004740 SET YR(1)=DUMYR(Y)*DUM
004771 SET YR(1)=NSIM)
005002 ELSE NO
005006 TELL CENTER THE VALUES FOR EACH SIMULATION YEAR INDIVIDUALLY)
005123 READ DUMYR
005132 END ASK
005033 END MODEL CHVAR
*
COMMENT
COMMENT
005034 MODEL ECH
005042 ASKENTER NAME OF ECONOMIC VARIABLE TO BE CHANGED OR LIST OR NONE) NONE
005067 ELSE LIST
005073 TELL DATA(GPI,GNP,RAD)
005100 ECH
005103 ELSE GNP
005107 CHVAR
005112 GNP=DUMYR
005126 PUT THS(GNP)
005133 ECH
005136 ELSE DPI
005142 CHVAR
005145 DPI=DUMYR
005161 PUT THS(DPI)
005166 ECH
005171 ELSE RAD
005175 CHVAR
005200 RAD=DUMYR
005214 PUT THS(RAD)
005221 ECH
005224 END ASK
005225 END MODEL ECH
*
COMMENT

```



```

COMMENT
005226 MODEL YRCHK
005234 READ MYEAR
005243 IF (MYEAR LT 1977) OR (MYEAR GT 1987)
005275 TELL (THE YEAR MUST BE BETWEEN 1977 AND 1987. ENTER NEW VALUE)
005312 YRCHK
005315 END IF
005316 MYEAR=MYEAR-1976
005332 END MODEL YRCHK
COMMENT
COMMENT
COMMENT
MODEL FCOST
005333 ASK (DO YOU WISH TO CHANGE THE VALUES OF THE ANNUALIZED INVESTMENT FOR 1977) NO
005341 ELSE YES
005367 TELL (ENTER VALUES TO BE ADDED TO THE ANNUALIZED INVESTMENT (1972 $) FOR EACH TYP
005371 E.)
005415 TELL (THESE VALUES WILL BE INFLATED THROUGH TIME AND WILL BE ADDED TO THE FIXED C
OST.)
005437 READ DUM7
005446 DUM7=1-RESID*DUM7/OEPREC
005475 OAI=DUM7*DEFL72/FIX72
005521 END ASK
005522 END MODEL FCOST

```

```

005523 MODEL VCOST
005531 ASK100 YOU WISH TO CHANGE THE VALUES OF FTAX, LFEE OR NONE) NONE
005554 ELSE FTAX
005560 TELLWHAT YEAR WOULD YOU LIKE THE NEW FUEL TAX TO BEGIN)
005574 YRCHK
005577 YRFT=NYEAR
005610 ASK100 YOU LIKE THE FUEL TAX TO REMAIN CONSTANT FOR ALL SUBSEQUENT YEARS) YES
005637 TELLENTER THE FUEL TAX VALUES IN DOLLARS, FIRST FOR AVIATION GAS, THEN FOR JET
      FUEL.)
005661 READ NEMTX
005670 SET YR1(NYEAR-NSIM)
005701 FTAX=NEMTX
005720 ELSE NO
005724 ASK100 YOU LIKE THE FUEL TAX TO CHANGE AT A CONSTANT RATE FOR ALL SUBSEQUENT
      YEARS) YES
005755 TELLENTER RATE OF CHANGE)
005763 READ DUM
005772 DUM=DUM+1
006006 TELLENTER THE FUEL TAX VALUES FOR THE FIRST YEAR OF THE CHANGED FUEL TAX, IN DO
      LLARS, FIRST FOR AVIATION GAS, THEN FOR JET FUEL.)
006041 SET YR1(NYEAR)
006047 READ FTAX
006056 SET YP1(NYEAR-NSIM)
006067 FTAXIF(Y1)=FTAX(F,Y)*DUM
006113 ELSE NO
006117 TELLENTER FUEL TAX VALUES, IN DOLLARS, FIRST FOR AVIATION GAS, THEN FOR JET FUE
      L.) ENTER VALUES FOR EACH YEAR.)
006147 SET YR1(NYEAR-NSIM)
006160 READ(YR1) FTAX
006170 END ASK
006171 END ASK
006172 SET YR11(NSIM)
006203 TELL THE NEW VALUES FOR THE FUEL TAX ARE)
006215 SHOW(10,10) FTAX.2.ORDER(YR1,FUEL)
006230 PUT THS(FTAX)
006235 VCOST
006240 ELSE LFEE
006244 TELLWHAT YEAR WOULD YOU LIKE THE LANDING FEE TO BEGIN)
006260 YRCHK
006263 YRLF=NYEAR
006274 ASK100 YOU LIKE THE LANDING FEE TO REMAIN CONSTANT FOR ALL SUBSEQUENT YEARS).
      YES
006323 TELLENTER THE LANDING FEE VALUES BY TYPE, IN DOLLARS)
006337 READ NEMLF
006346 SET YR1(NYEAR-NSIM)
006357 LFEE=NEMLF
006376 ELSE NO
006402 ASK100 YOU LIKE THE LANDING FEE TO CHANGE AT A CONSTANT RATE FOR ALL SUBSEQU
      NT YEARS) YES
006433 TELLENTER RATE OF CHANGE)
006441 READ DUM
006450 DUM=DUM+1
006464 TELLENTER LANDING FEE VALUES, BY TYPE, FOR THE FIRST YEAR OF THE IMPOSED FEE.)
006505 SET YR1(NYEAR)
006513 READ LFEE
006522 SET YR1(NYEAR-NSIM)

```

```

006533 LFEE(T,Y+1)=LFEE(T,Y)*DUM
006557 ELSE NO
006563 TELL(ENTER THE LANDING FEE VALUES, BY TYPE, FOR EACH YEAR, IN DOLLARS)
006602 SET YR1(NYEAR-NSIH)
006613 READ(YR1) LFEE
006623 END ASK
006624 END ASK
006625 SET YR1(1-NSIH)
006636 TELL(ENTER THE NEW VALUES FOR THE LANDING FEE ARE)
006650 SHOW(6,9) LFEE,2,ORDER(YR1,TYPE)
006663 PUT THIS(LFEE)
006670 VCOST
006673 END ASK
006674 END MODEL VCOST

```

```

006675 MODEL CHANGE
006703 DROP*
006705 ASIMI=NSIM-1
006721 ASK(ENTER NAME OF VARIABLE TO BE CHANGED, OR LIST, OR NONE)LIST
006744 SHOW TEXT=LEFT(1, AIRCRAFT VARIABLES)/)
006761 TELL DATA(ADMN,FC,FCINF,VC,VCINF,FHRF)
006771 SHOW TEXT=LEFT(1, ECONOMIC VARIABLES)/)
007006 TELL DATA(DPI,GNP,RAD)
007013 SHOW TEXT=LEFT(1,ECONOMIC DPI, GNP, RAD)
007026 SHOW TEXT=LEFT(1, FUEL VARIABLES)/)
007042 TELL DATA(SFC)
007045 SHOW TEXT=LEFT(1, PILOT VARIABLES)/)
007061 TELL DATA(ATPON,CPON,IPON,PCIN,PPON,SPON,URIPN,SCIX)
007073 SHOW TEXT=LEFT(1,PILOT ATPDN,CPDN,IPDN,PCIN,PPDN,SPDN,URIPN)/)
007115 CHANGE
007120 ELSE NONE
007124 ELSE PILOT
007130 TELL(1,THE CURRENT VALUES FOR THE PILOT RATES ARE)
007143 SHOW(7,7) PRATE.3,ORDER(YR1,DISP7)
007156 PPCM
007161 TELL(1,THE NEW VALUES FOR THE PILOT RATES ARE)
007173 SHOW(7,7) PRATE.3, ORDER(YR1,DISP7)
007206 CHANGE
007211 ELSE DATA(INDIRIATPON,CPON,IPON,PCIN,PPON,SPON,URIPN,FCINF,VCINF,FHRF)
007230 TELL(1,THE CURRENT VALUES FOR THE SELECTED VARIABLE ARE)
007244 SHOW(10,10) INDIR.3
007253 ASK(100 YOU STILL WISH TO CHANGE THE VALUES) NO
007273 ELSE YES
007277 TELL(ENTER NEW VALUES)
007305 READ INDIR
007314 PUT THS(INDIR)
007321 TELL(1,THE NEW VALUES FOR THE SELECTED VARIABLE ARE)
007334 SHOW(10,10) INDIR.3
007343 END ASK
007344 CHANGE
007347 ELSE ECONOMIC
007353 TELL(1,THE CURRENT VALUES FOR THE ECONOMIC DATA ARE)
007366 ETAB(1,Y)=DPI(Y)
007402 ETAB(2,Y)=GNP(Y)
007416 ETAB(3,Y)=RAD(Y)
007432 SHOW(10,10) ETAB.4,ORDER(YR1,DISP3)
007445 ECH
007450 TELL(1,THE NEW VALUES FOR THE ECONOMIC DATA ARE)
007462 ETAB(1,Y)=DPI(Y)
007476 ETAB(2,Y)=GNP(Y)
007512 ETAB(3,Y)=RAD(Y)
007526 SHOW(10,10) ETAB.4,ORDER(YR1,DISP3)
007541 CHANGE
007544 ELSE DPI
007550 TELL(1,THE CURRENT VALUES FOR THE SELECTED VARIABLE ARE)
007564 SHOW(10,10) DPI.4
007573 ASK(100 YOU STILL WISH TO CHANGE THE VALUES) NO
007613 ELSE YES
007617 CHVAR
007622 DPI=DUMYR
007636 .PUT THS(DPI)

```

```

007643 TELL (THE NEW VALUES FOR THE SELECTED VARIABLE ARE)
007656 SHOW (10,10) OPT.4
007665 END ASK
007666 CHANGE
007671 ELSE GNP
007675 TELL (THE CURRENT VALUES FOR THE SELECTED VARIABLE ARE)
007711 SHOW (10,10) GNP.4
007720 ASK (DO YOU STILL WISH TO CHANGE THE VALUES) NO
007740 ELSE YES
007744 CHVAR
007747 GNP=DUMYR
007763 PUT TMS(GNP)
007770 TELL (THE NEW VALUES FOR THE SELECTED VARIABLE ARE)
008003 SHOW (10,10) GNP.4
008012 END ASK
008013 CHANGE
008016 ELSE RAD
008022 TELL (THE CURRENT VALUES FOR THE SELECTED VARIABLE ARE)
008036 SHOW (10,10) RAD.4
008045 ASK (DO YOU STILL WISH TO CHANGE THE VALUES) NO
008065 ELSE YES
008071 CHVAR
008074 RAD=DUMYR
008110 PUT TMS(RAD)
008115 TELL (THE NEW VALUES FOR THE SELECTED VARIABLE ARE)
008133 SHOW (10,10) RAD.4
008137 END ASK
008140 CHANGE
008143 ELSE SCIX
008147 TELL (THE CURRENT VALUES FOR THE SCI MULTIPLIER ARE)
008162 SHOW (10,10) SCIX.3
008171 ASK (DO YOU STILL WISH TO CHANGE THE VALUES) NO
008211 ELSE YES
008215 TELL (ENTER NEW VALUES BY AGE GROUP FOR EACH YEAR)
008233 READ (AGE) SCIX
008240 PUT TMS(SCIX)
008245 TELL (THE NEW VALUES FOR THE SCI MULTIPLIER ARE)
008260 SHOW (10,10) SCIX.3
008267 END ASK
008270 CHANGE
008273 ELSE VC
008277 TELL (THE COMPONENTS OF THE VARIABLE COST INDEX ARE FTAX, THE FEDERAL FUEL TAX, A
MD LFEE, THE LANDING FEE)
008325 TELL (THE CURRENT VALUES FOR THE VARIABLE COST INDEX COMPONENTS ARE)
008344 SHOW (10,10) FTAX.2, ORDER (YR1, FUEL)
008357 SHOW (6,9) LFEE.2, ORDER (YR1, TYPE)
008372 VCOST
008375 CHANGE
008400 ELSE SFC
008404 TELL (THE CURRENT VALUES FOR THE SELECTED VARIABLE ARE)
008420 SHOW (12,10) SFC.2
008427 ASK (DO YOU STILL WISH TO CHANGE THE VALUES) NO
008447 ELSE YES
008453 TELL (ENTER NEW VALUES)
008461 JSFC=SFC
008475 READ SFC

```

```

010504 TSFC=1
010515 PUT INSI(SFC)
010522 TELL (THE NEW VALUES FOR THE SELECTED VARIABLE ARE)
010535 SHOW (12,10) SFC.2
010544 END ASK
010545 CHANGE
010550 ELSE ADRN
010554 TELL (THE CURRENT VALUES FOR THE SELECTED VARIABLE ARE)
010570 SHOW (12,10) ADRN.5
010577 ASK (DO YOU STILL WISH TO CHANGE THE VALUES) NO
010617 ELSE YES
010623 TELL (ENTER NEW VALUES)
010631 READ ADRN
010640 PUT INSI(ADRN)
010645 TELL (THE NEW VALUES FOR THE SELECTED VARIABLE ARE)
010660 SHOW (12,10) ADRN.5
010667 END ASK
010670 CHANGE
010673 ELSE FC
010677 TELL (THE CURRENT VALUES OF THE FIXED COST ARE)
010711 SHOW (10,10) FCINP.3
010720 TELL (ONE COMPONENT OF FC IS THE ANNUALIZED INVESTMENT)
010734 SHOW (10,10) AI.1
010743 FCOST
010746 CHANGE
010751 END ASK
010752 DELVC=0
010767 IF (YRLF*YRFT) NE 0
011012 SET TYPE (1-3,6)
011024 SET FUEL (1)
011032 FADIT,Y)= (FAX (F,Y) - FTAX77(F)) * OSFC (T) + IF TAX (F,Y) * (SFC (T) - OSFC (T)))
011124 SET TYPE (4,5,7)
011136 SET FUEL (2)
011144 FADIT,Y)= (FAX (F,Y) - FTAX77(F)) * OSFC (T) + IF TAX (F,Y) * (SFC (T) - OSFC (T)))
011236 SET TYPE
011242 SET FUEL
011246 DELVC (T,Y) = FADIT,Y) * (YRFT NE 0) * (YVAL (Y) GE YRFT)
011313 DELVC (T,Y) = (YRLF NE 0) * LFEE (T,Y) * 0.325 / (2 * HPOPX (T)) * (YVAL (Y) GE YRLF) + DELVC (T,Y)
011404 END IF
011435 IF TSFC NE 0
011421 DELVC (T,Y) = ALPHA (T) * (SFC (T) - OSFC (T)) + DELVC (T,Y)
011463 END IF
011464 VCVR = VCINP * VCINF + (DELVC * DEFL72 / VC72)
011533 PUT INSI(VCVR)
011540 FCVR = FCINP * FCINF + DAI
011571 PUT INSI(FCVR)
011576 YRFT=0
011607 YOLF=0
011620 TSFC=0
011631 OSFC=0
011644 DAI=0
011657 END MODEL CHANGE
011660 MODEL CHREQ
011665 OSFC=SFC
011702 SET TYPE (1-3,6)
011714 SET FUEL (1)

```

011722 ALPHA(T)=FA077(T)/SFC(T)-FTAX(F)
011750 SET TYPE(4,5,7)
011762 SET FUEL(2)
011770 ALPHA(T)=FA077(T)/SFC(T)-FTAX(F)
012016 SET TYPE*
012022 SET FUEL*
012026 CHANGE
012031 END MODEL CHREQ
012032 SAVE CHNGE,DATA=SEG1

```

001557 DATA SEG1 (SEGMENT GAO)
001573 DATA
COMMENT
* DEFINE THE FIXED DATA SETS
COMMENT
ATCN1, (COEFFICIENT IN ATCIN EQUATION), VALUE=0.01834
ATCN2, (COEFFICIENT IN ATCIN EQUATION), VALUE=0.0660
CCC, (CONSTANT IN CCIN EQUATION), VALUE=0.0587
CCPA0, (CONSTANT IN CCIN EQUATION), VALUE=0.837
CCVC, (CONSTANT IN CCIN EQUATION), VALUE=-2.024
FED75, (FEDERAL REGISTRATION COST FOR 1975), VALUE=0
FIFR, (TYPE, CATEGORY, (FRACTION OF FLIGHTS FILING IFR), VALUE=0
FIX72, (TYPE, (1972 FIXED COST), VALUE=0
FPH, (TYPE, CATEGORY, (FLIGHTS PER HOUR), VALUE=0
HC11, (CONSTANT IN HCI EQUATION), VALUE=2936
HC12, (CONSTANT IN HCI EQUATION), VALUE=-6.264
HFLT, (TYPE, CATEGORY, (HOURS FLOWN), VALUE=0
HPDN, (HPDN), VALUE=0.272
HRIN, (HRIN), VALUE=0.013
IOPPH, (TYPE, CATEGORY, (ITINERANT OPERATIONS PER HOUR), VALUE=0
LOPPH, (TYPE, CATEGORY, (LOCAL OPERATIONS PER HOUR), VALUE=0
SCC, (AC), (CONSTANT IN SCIN EQUATION), VALUE=0
SCCP, (AGE1), (CONSTANT IN SCIN EQUATION), VALUE=0
SCVC, (AGE1), (CONSTANT IN SCIN EQUATION), VALUE=0
TC72, (TYPE, CATEGORY, (1972 TOTAL COST), VALUE=0
TC75, (TYPE, CATEGORY, (1975 TOTAL COST), VALUE=0
TC76, (TYPE, CATEGORY, (1976 TOTAL COST), VALUE=0
VC72, (TYPE, (1972 VARIABLE COST), VALUE=0
* DEFINE THE SCRAP/H DATA SETS
COMMENT
*
AANUM, (TYPE, (NUMBER OF AIRPLANES)
AART, (TYPE, CATEGORY, (AIRCRAFT ACTIVATION RATE (AC/YR))
ADRT, (TYPE, CATEGORY, (AIRCRAFT DESTRUCTION RATE (AC/YR))
AT, (TYPE, CATEGORY, (ADJUSTED TIME)
ATC1, (AIRLINE TRANSPORT CERTIFICATES ISSUED RATE (P/YR))
ATCIN, (AIRLINE TRANSPORT CERTIFICATES ISSUED, NORMALIZED)
ATP, (AIRLINE TRANSPORT PILOT DEPARTURE RATE (P/YR))
CCI, (COMMERCIAL CERTIFICATES ISSUED)
CCIN, (COMMERCIAL CERTIFICATES ISSUED, NORMALIZED)
CCN75, (1975 CCIN VALUE HALVED)
CPD, (COMMERCIAL PILOT DEPARTURE RATE (P/YR))
DAA, (TYPE, CATEGORY, (DAA, (AC))
OPPA, (TYPE, CATEGORY, (DESIRED PILOTS PER AIRCRAFT (P/AC))
DUM1, (A TEMPORARY STORAGE AREA)
DUM2, (A TEMPORARY STORAGE AREA)
FAFUEL, (FUEL AVAILABLE (GALLONS))
FP, (FUEL TAX REVENUE)
HCT, (HELICOPTER CERTIFICATES ISSUED (P/YR))
HFA, (HFA (HRS/YR))
HFA1, (HFA1)
HFD, (HELICOPTER PILOT DEPARTURE RATE (P/YR))
HPO, (HELICOPTER RATINGS DEPARTURE RATE (P/YR))
HPI, (HELICOPTER RATINGS ISSUED RATE (P/YR))
IFR, (TYPE, CATEGORY, (IFR FLIGHT PLANS FILED, THOUSANDS)
IOPS, (TYPE, CATEGORY, (ITINERANT OPERATIONS, THOUSANDS)

```



```

004001 IPD, (INSTRUMENT PILOT DEPARTURE RATE (P/YR))
004002 IQI, (INSTRUMENT RATINGS ISSUED RATE (P/YR))
004003 LR, (LANDING FEE REVENUE)
004004 LOPS (TYPE, CATEGORY), (LOCAL OPERATIONS, THOUSANDS)
004005 PCT, (PRIVATE CERTIFICATES ISSUED RATE (P/YR))
004006 PPD, (PRIVATE PILOT DEPARTURE RATE (P/YR))
004007 RR, (REGISTRATION REVENUE)
004008 SCI, (STUDENT CERTIFICATES ISSUED RATE (P/YR))
004009 SCINAGE, (STUDENT CERTIFICATES ISSUED RATE, NORMALIZED (P/YR))
004010 SCN7STAGE1, (1975 SCIN VALUES HALVED)
004011 SPD, (STUDENT PILOT DEPARTURE RATE (P/YR))
004012 TCP, (TYPE, CATEGORY), (TOTAL COST, PREVIOUS YEAR)
004013 URIP, (UPGRADE RATE TO INSTRUMENT FROM PRIVATE (P/YR))
004014
004015 * DEFINE THE ENDOGENOUS DATA SETS
004016
004017 AA (TYPE, CATEGORY), (ACTIVE AIRCRAFT BY PRIMARY USE), TMS(3,30,BASE(1,1,TVAR))
004018 AASUM, (TOTAL AIRCRAFT), TMS(3,29,BASE(TVAR))
004019 APOT, (AOPS), (LOCAL AND ITINERANT OPERATIONS + IFR FLIGHT PLANS FILED),
004020 TMS(3,31,BASE(1,TVAR))
004021 ATP, (AIRLINE TRANSPORT PILOTS), TMS(3,52,BASE(TVAR))
004022 AUR (TYPE, CATEGORY), (AIRCRAFT UTILIZATION RATE (HRS/AC/YR)), TMS(3,28,BASE(1,1,TVAR))
004023 CP, (COMMERCIAL PILOTS), TMS(3,32,BASE(TVAR))
004024 FC (FUEL), (FUEL CONSUMED (MILLION GALLONS)), TMS(3,35,BASE(1,TVAR))
004025 FTP, (FEDERAL TAX REVENUE (MILLION DOLLARS)), TMS(3,37,BASE(1,TVAR))
004026 WFM (TYPE, CATEGORY), (HOURS FLOWN (THOUSANDS)), TMS(3,39,BASE(1,1,TVAR))
004027 WFSUM, (TOTAL HOURS FLOWN (THOUSANDS)), TMS(3,40,BASE(TVAR))
004028 HP, (HELICOPTER PILOTS), TMS(3,41,BASE(TVAR))
004029 MR, (HELICOPTER RATINGS), TMS(3,42,BASE(TVAR))
004030 IP, (INSTRUMENT RATINGS), TMS(3,43,BASE(TVAR))
004031 OPSUM, (TOTAL OPERATIONS (THOUSANDS)), TMS(3,27,BASE(1,1,TVAR))
004032 P, (PILOT SUBTOTAL), TMS(3,47,BASE(TVAR))
004033 PP, (PRIVATE PILOTS), TMS(3,46,BASE(TVAR))
004034 SP, (STUDENT PILOTS), TMS(3,48,BASE(TVAR))
004035 TC (TYPE, CATEGORY), (TOTAL COST), TMS(3,24,BASE(1,1,TVAR))
004036 TMP, (TOTAL HELIC RATINGS), TMS(3,49,BASE(TVAR))
004037 TP, (TOTAL PILOTS), TMS(3,50,BASE(TVAR))
004038
004039 * DEFINE THE EXOGENOUS DATA SETS
004040
004041 ADNR (CATEGORY), (AIRCRAFT DESTRUCTION RATE, NORMALIZED (AC/YR)), TMS(3,4)
004042 ATPDN, (AIRLINE TRANSPORT PILOT DEPARTURE RATE, NORMALIZED), TMS(3,6,BASE(1,TIM))
004043 AURN (TYPE, CATEGORY), (AIRCRAFT UTILIZATION RATE, NORMALIZED, (HRS/AC/YR)),
004044 TMS(3,5)
004045 CPON, (COMMERCIAL PILOT DEPARTURE RATE, NORMALIZED), TMS(3,6,BASE(2,TIM))
004046 DAUR (TYPE, CATEGORY), (DESIRED AIRCRAFT UTILIZATION RATE (AC/YR)),
004047 TMS(3,7)
004048 DPI, (DISPOSABLE PERSONAL INCOME (1972 $, 1972=1)), TMS(3,33,BASE(1,TIM))
004049 FCINF, (FIXED COST INFLATION FACTOR), TMS(3,34,BASE(1,TIM))
004050 FURF, (FLYING HOURS REQUIRED FACTOR), TMS(3,2)
004051 FIX (TYPE), (FIXED COST, ($/YR)), TMS(3,25,BASE(1,TIM))
004052 FIAX (FUEL), (FEDERAL FUEL TAX ($/GAL)), TMS(3,30,BASE(1,TIM))
004053 GNP, (GROSS NATIONAL PRODUCT (1972 $, 1972=1)), TMS(3,36,BASE(1,TIM))
004054 IPDN, (INSTRUMENT PILOT DEPARTURE RATE, NORMALIZED), TMS(3,6,BASE(1,TIM))
004055 LFEE (TYPE), (LANDING FEE), TMS(3,9,BASE(1,TIM))

```

```

006066 PCIN, (PRIVATE CERTIFICATES ISSUED RATE, NORMALIZED), TMS(3,6,BASE(4,TIM))
006123 POP(AGE1), (U.S. POPULATION), TMS(3,45,BASE(1,TIM))
006153 PPON, (PRIVATE PILOTS DEPARTURE RATE, NORMALIZED), TMS(3,6,BASE(5,TIM))
006210 RAD(RAD), TMS(3,1,BASE(1,TIM))
006233 RAD1, (RAD FOR YEAR - 1), TMS(3,1,BASE(1,TIM))
006263 RAD2, (RAD FOR YEAR - 2), TMS(3,1,BASE(1,TIM))
006306 SCIX(AGE1), (SCI MULTIPLIER), TMS(3,16,BASE(1,TIM))
006336 SFC(TYPE), (SPECIFIC FUEL CONSUMPTION (GALLONS/HOUR)), TMS(3,17)
006367 SPON, (STUDENT PILOTS DEPARTURE RATE, NORMALIZED), TMS(3,6,BASE(6,TIM))
006424 URIPH, (UPGRADE RATE TO INSTRUMENT PILOT, NORMALIZED), TMS(3,6,BASE(7,TIM))
006461 VCITYPE1, (VARIABLE COST INDEX ($/HR), (1972 $, 1972-1)), TMS(3,26,BASE(1,TIM))
006517 VCITYPE1, (VARIABLE COST INDEX FOR FIRST YEAR), TMS(3,15)
006547 END DATA
COMMENT .
COMMENT .
006547 READ FPM
DATA 0.909 0.0 0.069 0.0 0.0 0.0 0.941 0.0 0.0
DATA 0.663 0.0 0.876 0.720 0.804 0.0 0.0 1.684
DATA 0.960 0.0 0.791 0.0 0.0 0.0 1.379 0.0
DATA 0.0 0.709 0.350 0.0 0.0 0.0 0.809 0.0
DATA 0.998 0.0 0.891 0.0 0.0 0.0 0.855 0.0
DATA 1.067 0.0 0.905 1.509 0.693 1.148 0.0
DATA 1.120 0.0 1.139 0.940 0.955 0.653 0.0
006547 READ (CATEC) HF (1,70,10)
DATA 4.664 0 1664 0 0 90 0
DATA 3.37 0 1510 0.50 7.35 0 215
DATA 9074 0 516 0 0 32 0
DATA 0 2002 64 0 0 161 0
DATA 4.005 0 165 0 0 72 0
DATA 945 0 1384 425 174 77 561
DATA 4.317 0 299 44 31 354 207
006547 READ FE075
DATA 54 100 100 293 443 30 96
006547 READ FIX72 (1,70,10)
DATA 6561 22216 22216 06605 123724 14020 30320
006547 READ VC72
DATA 12.40 36.99 36.99 77.22 234.40 20.90 45.70
006547 READ TC72 (1,70,10)
DATA 8371 0 29429 0 124214 237601 19409 0
DATA 9107 0 35310 0 27210 0 15434 0
DATA 7009 0 28134 0 0 21495 0
DATA 0 32203 28134 0 0 21244 0
DATA 11790 0 39194 197002 272053 24526 63702
DATA 11291 0 30420 117650 174041 23502 49098
DATA 10667 0 35906 0 130474 253776 17710 0
006547 READ TC75 (1,70,10)
DATA 9424 0 41206 130474 253776 0 50705
DATA 10311 0 33796 0 0 15603 0
DATA 6647 0 38740 33755 0 18496 0
DATA 0 38740 38623 0 19274 0
DATA 12022 0 47361 179231 392636 20951 65771
DATA 11014 0 30290 119021 215442 22799 55572
DATA 1191 0 38256 0 17695 0
006547 READ TC76 (1,70,10)
DATA 110066 0 38256 0 17695 0

```

DATA	18708	0	44157	128412	260452	0	54090
DATA	9321	0	36726	0	0	14104	0
DATA	0	42452	37557	0	0	19365	0
DATA	13317	0	42321	0	0	18808	0
DATA	13471	0	50058	194461	401399	23032	70745
DATA	12824	0	40223	114551	272152	23158	59571
806547	READ FIFR	0.0	0.374	0.0	0.0	0.0	0.0
DATA	0.117	0.0	0.528	0.772	0.979	0.0	0.0
DATA	0.220	0.0	0.259	0.0	0.0	0.0	0.0
DATA	0.650	0.0	0.0	0.0	0.0	0.0	0.0
DATA	0.0	0.016	0.0	0.0	0.0	0.0	0.0
DATA	0.046	0.0	0.092	0.0	0.0	0.2	0.0
DATA	0.185	0.0	0.587	0.634	0.8	0.0	0.0
DATA	0.032	0.3	0.128	0.493	0.667	0.023	0.023
806547	READ ICPP4	0.0	1.49	0.0	0.0	1.18	0.0
DATA	1.37	0.0	1.69	1.4	1.6	0.0	3.03
DATA	1.16	0.0	1.11	0.0	0.0	0.02	0.0
DATA	0.92	0.0	0.72	0.0	0.0	0.54	0.0
DATA	0.0	0.81	0.4	0.0	0.0	0.17	0.0
DATA	0.5	0.0	1.9	3.1	1.39	2.12	2.12
DATA	1.68	0.0	0.72	1.3	1.91	0.35	0.35
DATA	0.79	0.0	0.0	0.0	0.0	1.29	0.0
806547	READ LOPPH	0.0	0.38	0.0	0.02	0.0	0.19
DATA	0.78	0.0	0.06	0.05	0.0	4.42	0.0
DATA	0.65	0.0	0.91	0.0	0.0	4.32	0.0
DATA	2.28	0.0	0.0	0.0	0.0	15.86	0.0
DATA	0.0	1.6	0.0	0.0	0.0	0.16	0.16
DATA	5.21	0.0	4.23	0.0	0.0	2.72	2.72
DATA	0.7	0.0	0.16	0.08	0.0	0.0	0.0
DATA	2.36	0.0	2.04	1.84	0.0	0.0	0.0
806547	READ SCC	0.00141	0.00158	0.000426			
DATA	0.00141	0.00158	0.000426				
806547	READ SCVC	-1.294	-1.455	-1.884			
DATA	-1.294	-1.455	-1.884				
806547	READ SCDP	0.208	0.193	1.019			
DATA	0.208	0.193	1.019				

```

006547  MODEL GADYN
006555  TVAR=1
006566  TIM=TVAR
006577  TIM1=TVAR+1
006613  TIM2=TVAR+2
006613  *
006613  * INITIAL EQUATIONS
006613  *
006627  JAA=0
006644  P = SP + PP + CP + ATP
006666  THP = HP + MR
006702  IP = P + HP
006716  CCN75=CC*(VC(11)**CCVC)*(RAD**CCRAD)/2
006756  SCN75=SCC*(VC(11)*FHRF**SCVC)*(DP1**SCDP)/2
007031  HFN=HF
007053  HF = HF * 1000
007072  LOPS=LOPPH*HFM
007116  OPS=LOPPH*HFM
007142  OPS=LOPS*1OPS
007166  IFRF=FIFR*FPH*HFM
007217  APORT(11) = SUM(IT,C1)(LOPS(IT,C1))
007251  APORT(12) = SUM(IT,C1)(OPS(IT,C1))
007303  APORT(13)=SUM(IT,C1)(IFRF(IT,C1))
007335  OPSUM=SUM(IT,C1)(OPS(IT,C1))
007367  AASUM = SUM(IT,C1)(AAT(IT,C1))
007421  +FSUM=SUM(IT,C1)(HFM(IT,C1))
007453  TC=TC76/TC72
007477  TCP=TC75/TC72
007523  AT=1
007540  AT(1,1) = 2
007551  AT(1,1) = 3
007562  AT(1,2) = 2
007573  AT(1,2) = 3
007604  AUR=0
007604  *
007604  * RATES SECTION
007604  *
007621  RATE
007621  TIME=TIME-1976
007623  TIME=TIME+1
007637  TIME=TIME+2
007653  PUT CORE(ATPON,CPON,DPI,FIK ,FTAX,GNP,IPON,LFEE,PCIN,POP,PPDN,
007667  RAD,RAO1,RAD2,SCIX,SPON,JURIP,VC)
007667  PUT CORE(IFCINF)
007667  * RATES SECTION FOR PILOT DATA
007667  *
007715  SCIN=SCC*(VC(11)*FHRF**SCVC)*(LOPI**SCDP)*SCIX
007722  * COMPUTE NEW COMMERCIAL PILOTS RATE, NORMALIZED ---- #CCINF#
007722  *
007776  CCIN=CC*(VC(11)**CCVC)*(RAD**CCRAD)
008033  SCIN=CCIN*(SCIN GE SCN75)+(SCN75*(SCIN LT SCN75))
008107  CCIN=CCIN*(CCIN GE CCN75)+(CCN75*(CCIN LT CCN75))
008107  * COMPUTE PILOT RATES
008107  *
008153  SCI = (SUM(14)(SCIN(A) * POP(A)))
008153  CPO = CPON * CP
008213  IPO = IPDH * IP
008227  PPD = PPDN * PP
008243  SPD = SPDH * SP
008257

```

```

010273 PCI = PCIN * SP
010307 JRIIP = URIPN * PP
010323 CCI = CCIN * PP
010337 IRI = URIP * CCI
010353 YCI=HCI1*(VC(6)*HCI2) (
010376 WPD=MPDN*HP
010412 MPD=CPDN*4R
010426 HRI=HRI*CP
010442 ATCIN=ATCIN1*(ATCN2*(RAD1-RAD2)/RAD2)
010477 ATCIN=ATCIN*(ATCIN GE 0)
010522 ATCI=ATCIN*CP
010536 ATPO=ATPDN*ATP
* RATES SECTION FOR AIRCRAFT DATA
COMMENT *
* COMPUTE AIRCRAFT UTILIZATION RATE ---- #AUR#
010552 AUR(1,1)=AURN(1,1)
010563 AUR(1,2)=AURN(1,2)
010574 AUR(1,3)=AURN(1,3)
010605 AUR(3,3)=AURN(3,3)
010616 AUR(6,3)=AURN(6,3)
010627 AUR(3,1)=AURN(3,1)*(VC(3)*(-0.268))
010656 AUR(3,2)=AURN(3,2)*(VC(3)*(-0.466))
010705 AUR(4,2)=AURN(4,2)*(VC(4)*(-2.192))*(RA)*(-1.5))
010752 AUR(5,2)=AURN(5,2)*(VC(5)*(-0.207))
011001 SET CATEG(7)
011007 SET TYPE(4,5)
011117 AUR=AURN
011136 SET TYPE(6,7)
011046 AUP=AURN*(VC*0.415)
011101 SET TYPE*
011105 SET CATES*
011111 QUM1 = TIME - 1970
011125 QUM2=EXP(1.233*(0.921*QUM1))
011155 HFA=400600/QUM2
011171 AUP(2,4)=0.9*HFA/AA(2,4)
011210 AUR(3,4)=0.026*HFA/AA(3,4)
011227 AUR(4,4)=0.074*HFA/AA(4,4)
011246 AUR(6,5)=(HCI1*HRI)*23.8/AA(6,5)
011274 QUM1=134.1*PCI*(63.6*IRI)+(14.9*SCI)
011340 AUP(1,5)=0.967*QUM1/AA(1,5)
011357 AUP(3,5)=0.033*QUM1/AA(3,5)
011376 HFAI=271000*(GNP**3.0)
011421 AUP(1,6)=0.28*HFA/AA(1,6)
011448 AUP(3,6)=0.40*HFA/AA(3,6)
011457 AUR(4,6)=0.14*HFA/AA(4,6)
011476 AUR(5,6)=0.03*HFA/AA(5,6)
011515 AUR(6,6)=0.03*HFA/AA(6,6)
011534 AUP(7,6)=0.12*HFA/AA(7,6)
011553 AUR(6,1)=AUR(6,1)*(GNP**(-2.048))
011602 AUP(7,2)=AURN(7,2)*(GNP**(-0.409))*(VC(7)*(-0.436))
011647 AUR(1,7)=(PP*CP*ATP)*(DPI**2.69)*6.5*0.917/AA(1,7)
011715 AUP(3,7)=(PP*CP*ATP)*(DPI**2.69)*6.5*0.083/AA(3,7)
COMMENT * COMPUTE AIRCRAFT DESTRUCTION RATE ---- #ADR#
011743 ADR = ADPN * AUR * AA * JT
011763 CPMEYF * AVOID NEGATIVE VALUE FOR DESTRUCTION RATE
012017 .ADP = ADR * (ADR GE 0)

```

```

* . COMPUTE DESIRED PILOTS PER AIRCRAFT --- #DPPA#
012052 TC=VC*AUR*VG72*(FIX*FIX72)/TC72
012126 DPPA=0.0
012163 OPPA(1,1)=21.5*(GNP**(-2.80))
012172 OPPA(3,1)=66.1*(GNP**(-3.90))*(FIX(3)*0.23)
012233 OPPA(6,1)=93.1*(TCP(6,1)*1.91)*(GNP**(-2.65))
012274 OPPA(1,3)=7.34*(OPT**(-1.03))
012323 OPPA(3,3)=199*(OPT**(-1.47))
012352 OPPA(6,3)=99.5*(OPT**(-0.70))
012401 SET CATEG(1,3)
012411 SET TYPE(1,3,6)
* . LOWER LIMIT FOR #DPPA# IS #L#
012423 DPPA=DPPA*(OPPA GE 1) + (OPPA LT 1)
* . COMPUTE DESIRED ACTIVE AIRCRAFT --- #DAA#
012472 SET TYPE(1,3)
012502 OAA = CP + PP + ATP / DPPA
012540 SET TYPE(6)
012546 OAA = THP / OPPA
012572 SET TYPE*
012575 SET CATEG*
012602 JAA(1,2)=-3199*(GNP*4357)
012625 JAA(3,2)=-5242*(GNP*10146)-(TCP(3,2)*732)
012662 JAA(4,2)=-4431*(5046*GNP)
012705 JAA(5,2)=-5504*(6049*GNP)
012730 JAA(7,2)=-1733*(GNP*2036)
* . COMPUTE DESIRED AIRCRAFT UTILIZATION RATE --- #DAUR#
012753 DAUR(3,5)=-303*(445*TCP(3,5))
* . COMPUTE THE REMAINING #DAA#
012776 SET CATEG(4-6)
013006 OAA = HF / DAUR
013032 SET CATEG(7)
013040 SET TYPE(1,3)
013050 OAA = HF / DAUR
013074 SET TYPE*
013100 SET CATEG*
013104 JAA(4,7)=169
013115 JAA(5,7)=-521*(756*GNP)-(102*FIX(5))
013152 JAA(6,7)=-230*(1007*FIX(6))+(1667*GNP)
013207 JAA(7,7)=-1363*(1645*GNP)
* . COMPUTE AIRCRAFT ACTIVATION RATES --- #AAR#
013232 AAR = OAA - AA / AT
013263 TCP=IC
* . LEVELS
* . EQUATIONS FOR TIME = T+1
PUT THSIAA,AASUM,APORT,ATP,AUR,CP,FC, FTR,HFM,HFSUM,HP,HR,IP,OPS,OPSUM,P,PP,
SP,TC,THP,TP)
LEVEL
013333 TVAR=TVAR+1
013335 SET CATEG*
013351 SET FUEL*
013355 SET TYPE*
* . LEVELS SECTION FOR AIRCRAFT SUPPLY DATA
* . COMPUTE HOURS FLOWN --- #HF#

```

```

013365 SET CATEG(14-6)
013375 HF=AUR*AA*DT
013424 SET CATEG(7)
013432 SET TYPE(1,3)
013442 HF=AUR*AA*DT
013452 * COMPUTE ACTIVE AIRCRAFT --- #AA#
013471 SET TYPE*
013475 SET CATEG*
013501 AA = AAR - ADR * DT + AA
013535 * COMPUTE REMAINING HOURS FLOWN
013535 SET CATEG(1-3)
013545 HF=AUR*AA*DT
013574 SET CATEG(7)
013602 SET TYPE(14-7)
013612 HF=AUR*AA*DT
013641 SET TYPE*
013645 SET CATEG*
013655 * COMPUTE HOURS FLOWN IN THOUSANDS FOR TABLE OUTPUT --- #HF1000#
013655 HFM=HF/1000
013673 LOPS=LOPH*HFM
013717 LOPS=LOPH*HFM
013743 OPS=LOFS*OPS
013767 TFR=FIFR*FPH*HFM
014023 APORT(1)=SUM(IT,C) (LOPS(IT,C))
014032 APORT(2)=SUM(IT,C) (LOPS(IT,C))
014104 APORT(3)=SUM(IT,C) (TFR(IT,C))
014136 OPSUM=SUM(IT,C) (OPS(IT,C))
014170 AASUM = SUM(IT,C) (AAT(IT,C))
014222 HFSUM=SUM(IT,C) (HFM(IT,C))
014232 * LEVELS SECTION FOR FUEL DATA
014254 * COMPUTE FUEL CONSUMED --- #FC#
014254 SET TYPE(1-3,6)
014266 SET FUEL(1)
014274 FC(F)=SUM(IT) (SFC(IT)*SUM(IC) (HF(IT,C)))
014351 SET TYPE(14,5,7)
014365 SET FUEL(2)
014373 FC(F)=SUM(IT) (SFC(IT)*SUM(IC) (HF(IT,C)))
014432 SET TYPE*
014456 SET FUEL*
014456 * COMPUTE FUEL TAX REVENUE
014462 FR=SUM(FI) (FTAX(F)*FC(F))
014516 AANUM(IT) = SUM(IC) (AA(IT,C))
014531 SET TYPE(1-6)
014561 LR=SUM(IT) (LFEE(IT)*SUM(IC) (OPS(IT,C))) * 0.325/2
014642 SET TYPE*
014646 RP = SUM(IT) (FE075(IT) * FCINF) * AANUM(IT)
014712 FTP=FR*RP*LR
014731 FC=FC/1000000
014753 FT=FTR/1000000
014753 * LEVELS SECTION FOR PILOT DATA
014764 (P=GP*(OT*(CCI-CPD-ATCI)))
015021 ATP=ATP*(OT*(ATCI-ATPO))
015053 IP = IP + (OT * (IRI - IPD))
015105 PP = PP + (OT * (PCI - CCI - PPD))
015142 SP = SP + (OT * (SCI - PCI - SPD))
015177 HR = HRI - HRD * DT + HR

```

```

015221  HR = HCI - MPD + QT + HP
015243  P = SP + PP + CP + ATP
015265  TP = P + HP
015301  THP = HP + HQ
015315  END MODEL GADYN
COMMENT
COMMENT
COMMENT
015316  MODEL DYNAM
015324  COMPUTE TIME=1977
015335  ENDING=ENSIN
015346  GADYN
015351  COMPUTE TIME=0
015362  END MODEL DYNAM
015363  SAVE DYN.DATA=SEG1

```


002040		DATA
0002040	AUTR(TYPE,CATEG,YR1), (AIRCRAFT UTILIZATION RATE (HRS/AC/YR)),TMS(3,28)	
0002073	DPYR(YR1),(DPI (1972 \$, 1972=1)),TMS(3,33)	
0002120	FCLYR(YR1),(FIXED COST INFLATION FACTOR),TMS(3,36)	
0002147	FXFR(FUEL,YR1),(FEDERAL FUEL TAX (DOLLARS/GALLON)),TMS(3,38)	
0002200	GMPYR(YR1),(GNP (1972 \$, 1972=1)),TMS(3,36)	
0002225	RADYR(YR1),(RAO (1972 \$, 1972=1)),TMS(3,1,RASE(3))	
0002255	ASGYR(YR1),(TOTAL AIRCRAFT,TMS(3,29))	
0002301	AVATYR(TYPE,CATEG,YR1),(ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR),TMS(3,30)	
0002350	APYR(AOPS,YR1),(LOCAL AND ITINERANT OPERATIONS PLUS FLIGHT PLANS FILED A S REPORTED ON JANUARY 1 OF DESIGNATED YEAR),TMS(3,31)	
0002416	ATPYR(YR1),(AIR TRANSPORT PILOTS),TMS(3,52)	
0002443	CPYR(YR1),(COMMERCIAL PILOTS),TMS(3,32)	
0002470	FCGY(FUEL,YR1),(FUEL CONSUMED (MILLION GALLONS) DURING PREVIOUS YEAR, AS REPORTE D ON JANUARY 1 OF DESIGNATED YEAR),TMS(3,35)	
0002536	FHYR(TYPE,YR1),(FIXED CJSI,(YR1)),TMS(3,25)	
0002564	FYR(CUMIND,YR1),(FEDERAL TAX REVENUE DJRING PREVIOUS YEAR AS REPORTED ON JANUA RY 1 OF DESIGNATED YEAR),TMS(3,37)	
0002627	HMYR(TYPE,CATEG,YR1),(HOURS FLOWN (THOUSANDS) DURING PREVIOUS YEAR AS REPORT E D ON JANUARY 1 OF DESIGNATED YEAR),TMS(3,39)	
0002674	HFSYR(YR1),(TOTAL HOURS FLOWN (THOUSANDS)),TMS(3,40)	
0002723	HPYR(YR1),(HELICOPTER PILOTS),TMS(3,41)	
0002750	HRPYR(YR1),(HELICOPTER RATINGS),TMS(3,42)	
0002775	IPYR(YR1),(INSTUMENT RATINGS),TMS(3,43)	
0003022	OPSGYR(YR1),(TOTAL OPERATIONS),TMS(3,44)	
0003047	OPCYR(TYPE,CATEG,YR1),(OPERATIONS (THOUSANDS) DURING PREVIOUS YEAR AS REPORTED O N JANUARY 1 OF DESIGNATED YEAR),TMS(3,27)	
0003114	PPYR(ASEL,YR1),(U.S. POPULATION),TMS(3,45)	
0003141	PPYR(YR1),(PRIVATE PILOTS),TMS(3,46)	
0003165	PYR(YR1),(PILOT SUBTOTAL),TMS(3,47)	
0003211	SPYR(YR1),(STUDENT PILOTS),TMS(3,48)	
0003235	TNPVP(YR1),(TOTAL HELIC RATINGS),TMS(3,49)	
0003262	TPYR(YR1),(TOTAL PILOTS),TMS(3,50)	
0003306	VCIYR(YR1),(VARIABLE COST INFLATION FACTOR),TMS(3,51)	
0003335	VCYR(TYPE,YR1),(VARIABLE COST (B/YR), (1972 \$, 1972=1)),TMS(3,26)	
0003367	END	
0003367	*	
0003367	*	
0003367	* COMMENT	
0003367	* COMMENT	
0003367	TABLE	
0003367	PTAB(YR1),DATA1 /SPYR/PPYR/CPYR/ATPYR/PYR//HPYR//TPYR//IPYR/HRYR/TNPYR///),	
0003367	SIZE(20,9),TITLE(PILOT DATA)	
0003451	TOTS(YR1),DATA1 ASGYR/HFSYR/OPSGYR),SIZE(30,10),	
0003451	TITLE(TOTALS FOR AIRCRAFT, HOURS FLOWN AND OPERATIONS)	
0003451	ETAR(YR1),DATA1(DPYR,4/GMPYR,4/RADYR,4)/SIZE(23,8),	
0003451	TITLE(ECONOMIC DATA)	
0003451	END	

```

003534 MODEL PRINT
003542 000P*
003544 ASKWHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.),NONE
003570 ELSE LIST
003574 SHOW TEXT=LEFT(Identifier DESCRIPTION 1/,
(AIRCRAFT1 ACTIVE AIRCRAFT BY YEAR),
(AIRCRAFT2 ACTIVE AIRCRAFT BY USER CATEGORY),
(AIRPORTS LOCAL AND ITINERANT OPERATIONS PLUS IFR FLIGHT PLANS FILED),
(AIRUTIL AIRCRAFT UTILIZATION RATES 1/,
(ECONOMIC DPI-GNP,RAD)),
(FIXEDCOST FIXED COST 1/,
(VARCOST VARIABLE COST 1/,
(FUEL FUEL CONSUMED IN MILLIONS OF GALLONS),
(HOURSFLOWN HOURS FLOWN IN THOUSANDS),
(OPERATIONS TOTAL OPERATIONS, IN THOUSANDS),
(PILOTS SP,PP,CP,ATP,P,HP,TP,IP,HR,THP),
(REVENUE FEDERAL TAX REVENUE),
(TOTALS TOTAL AIRCRAFT, TOTAL HOURS FLOWN, TOTAL OPERATIONS),
PRINT
004040 ELSE AIRCRAFT2
004043 SHOW(15,12) AAYR,TOTAL(CATEG,TYPE),ORDER(CATEG,TYPE,YR1)
004047 PRINT
004067 ELSE AIRCRAFT1
004072 SHOW(6,12) AAYR, TOTAL(CATEG,TYPE), ORDER(YR1,TYPE,CATEG)
004076 PRINT
004116 ELSE AIRPORTS
004121 SHOW(6,14) APTYR, ORDER(YR1,ADPS)
004125 PRINT
004140 ELSE AIRUTIL
004141 SHOW(15,12) AUYR,ORDER(CATEG,TYPE,YR1)
004147 PRINT
004163 ELSE ECONOMIC
004166 ETAR
004172 PRINT
004174 ELSE FIXEDCOST
004177 SHOW(7,9) FIXYR,3,ORDER(YR1,TYPE)
004203 PRINT
004216 ELSE VARCOST
004221 SHOW(7,9) VCYR,3,ORDER(YR1,TYPE)
004225 PRINT
004240 ELSE FUEL
004243 SHOW(10,15) FCGY,ORDER(YR1,FUEL),TOTAL(FUEL)
004247 PRINT
004265 ELSE HOURSFLOWN
004270 SHOW(15,12) HFMYR,TOTAL(CATEG,TYPE),ORDER(CATEG,TYPE,YR1)
004274 PRINT
004314 ELSE OPERATIONS
004317 SHOW(15,12) OPSYR,TOTAL(CATEG,TYPE),ORDER(CATEG,TYPE,YR1)
004321 PRINT
004343 ELSE PILOTS
004346 PTAR
004352 PRINT
004354 ELSE REVENUE
004357 FIPYR=ETYR*1000000
004363 SHOW(10,16) FIRYR,ORDER(YR1,DUHIND)
004405 PRINT
004420

```

004423
004427
004431
004434
004435
004436

ESSE TOTALS
TOTL
PRINT
END ASK
END PRINT
SAVE SHOW, DATA=SE61

COMMENT
 COMMENT *OPEN THE SCATTER PLOT SECTION
 COMMENT
 001557 DATA SEGI (SCATTER PLOTS)
 COMMENT
 COMMENT *READ IN THE STUB HEADINGS FOR THE INDEXES
 COMMENT
 001573 READ CATEG(1,25)
 DATA BUSINESS TRANSPORTATION
 DATA CORPORATE TRANSPORTATION
 DATA PERSONAL FLYING
 DATA AERIAL APPLICATION
 DATA INSTRUCTIONAL FLYING
 DATA ATP TAXI
 DATA OTHER
 001637 READ FUEL (1,15)
 DATA AVIATION GAS
 DATA JET FUEL
 001646 READ TYPE(1,35)
 DATA SINGLE-ENGINE PISTON (NONAERIAL)
 DATA SINGLE-ENGINE PISTON (AERIAL)
 DATA MULTI-ENGINE PISTON
 DATA TURBOPROP
 DATA TURBOJET
 DATA PISTON-ENGINE HELICOPTER
 DATA TURBINE-ENGINE HELICOPTER

```

COMMENT
COMMENT *DEFINE THE DATASETS TO BE PLOTTED
COMMENT
001730 DATA
001731 AA (TYPE,CATEG,YR1) 1 (NUMBER OF ACTIVE AIRCRAFT) TMS(3,30)
001732 AASUM(YR1) (TOTAL NUMBER OF AIRCRAFT) TMS(3,29)
001733 ATP (YR1) (AIRLINE/TRANSPORT PILOTS) TMS(3,52)
001734 AUP (TYPE,CATEG,YR1) (AIRCRAFT UTILIZATION RATE (HRS/AC/YR)) TMS(3,28)
001735 CP (YR1) (COMMERCIAL PILOTS) TMS(3,32)
001736 DPI (YR1) (DISPOSABLE PERSONAL INCOME (1972 $,1972=1)) TMS(3,31)
001737 FC (FUEL,YR1) (FUEL CONSUMED (MILLION GALLONS)) TMS(3,35)
001738 FIX (TYPE,YR1) (FIXED COST INDEX ($/HRI),(1972 $,1972=1)) TMS(3,25)
001739 FTPI(YR1),(FEDERAL TAX REVENUE (MILLION DOLLARS)),TMS(3,37) TMS(3,36)
001740 GNP (YR1) (GROSS NATIONAL PRODUCT (1972 $,1972=1)) TMS(3,34)
001741 HFSUM(YR1) (HOURS FLOWN (THOUSANDS)) TMS(3,40)
001742 HP (YR1) (HELICOPTER PILOTS) TMS(3,41)
001743 HR (YR1) (HELICOPTER RATINGS) TMS(3,42)
001744 IP (YR1) (INSTRUMENT RATINGS) TMS(3,43)
001745 OPS (TYPE,CATEG,YR1) (OPERATIONS (THOUSANDS)) TMS(3,27)
001746 OPSUM(YR1) (TOTAL OPERATIONS (THOUSANDS)) TMS(3,44)
001747 P (YR1) (PILOT SUBTOTAL) TMS(3,47)
001748 PP (YR1) (PRIVATE PILOTS) TMS(3,46)
001749 RAD (YR1) (REVENUE AIRCRAFT DEPARTURES (1972 $,1972=1)) TMS(3,45)
001750 TMS(3,1)
001751 BASE(3)
001752 TMS(3,48)
001753 TMS(3,24)
001754 TP (YR1) (TOTAL HELICOPTER RATINGS) TMS(3,49)
001755 TMS(3,50)
001756 VC (TYPE,YR1) (VARIABLE COST INDEX ($/HRI),(1972 $,1972=1)) TMS(3,26)
001757 DUMH(YR1),(A DUMMY DATA SET FOR THE FIRST VARIABLE OF A PLOT)
001758 INDIR*,(AN INDIRECT DATA SET FOR THE FIRST VARIABLE OF A PLOT)
001759 IND2*,(AN INDIRECT DATA SET FOR THE SECOND VARIABLE OF A PLOT)
001760 END DATA

```

```

COMMENT *
COMMENT * BUILD THE MODELS THAT ALLOW THE USER TO SEE PLOTS OF HIS RESULTS
COMMENT *
COMMENT * MODEL DESCRIPTION
COMMENT * SELC SELECTS THE USER CATEGORY FOR A PLOT
COMMENT * SELT SELECTS THE AIRCRAFT TYPE FOR A PLOT
COMMENT * PLOT LISTS ALL POSSIBLE PLOTS AND SHOWS THE REQUESTED ONES
COMMENT *
COMMENT *
COMMENT *
COMMENT * MODEL SELC
003127 ASK(SELECT USER CATEGORY, OR LIST) LIST
003135 SHOW TEXT=LEFT(USER CATEGORIES, . . 1),
003153 11 BUSINESS TRANSPORTATION),
12 CORPORATE TRANSPORTATION),
13 PERSONAL FLYING),
14 AERIAL APPLICATION),
15 INSTRUCTIONAL FLYING),
16 AIR TAXI),
17 OTHER)
003251 SELC
003254 ELSE INDEX=CATEG
003260 END ASK
003261 END MODEL SELC
COMMENT *
COMMENT *
COMMENT * MODEL SELT
003279 ASK(SELECT AIRCRAFT TYPE, OR LIST) LIST
003306 SHOW TEXT=LEFT(AIRCRAFT TYPES, . . 1),
11 SINGLE-ENGINE PISTON, NONAERIAL),
12 SINGLE-ENGINE PISTON, AERIAL),
13 MULTI-ENGINE PISTON),
14 TURBOPROP),
15 TURBOJET),
16 PISTON-ENGINE HELICOPTER),
17 TURBINE-ENGINE HELICOPTER)
003413 SELT
003416 ELSE INDEX=TYPE
003422 END ASK
003423 END MODEL SELT
COMMENT *
COMMENT *

```

```

003424      MODEL XAXIS
003432      IVAR=1
003443      ASK(PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OR LIST), LIST
003467      TELL DATA(AA,ASUM,ATP,AUR,CP,DPI,FC,FIX,FTR,GNP,HF,HFSUM,HP,HR,IP,OPS,OPSUM,P,
           PP,RAD,SP,TC,THP,TP,VC)
003522      XAXIS
003525      ELSE TIME
003531      IVAR=0
003542      ELSE DATA=IND2(AASUM,CP,HFSUM,HP,OPSUM,P,PP,SP,TP,ATP,DPI,GNP,RAD,FTR,HP,IP,THP,
           HP)
           ELSE DATA=IND2(FC)
003571      ASK(PLEASE ENTER EITHER 1 FOR AVIATION GAS, OR 2 FOR JET FUEL),END
003623      ELSE INDEX=FUEL
003627      END ASK
003630      ELSE DATA=IND2(VC,FIX)
003637      SELT
003642      ELSE DATA=IND2(AA,HF,OPS,AUR,TC)
003654      SELT
003657      SELC
003662      END ASK
003663      END MODEL XAXIS

```



```

003664 MODEL PLOT
003672 DPOP*
003674 SET TYPE*
003676 SET CATEG*
003700 SET FUEL*
003704 ASKINWHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE, NONE
003710 ELSE LIST
003735 TELL DATA(AA,AASUN,ATP,AUR,CP,OPT,FC,FIX,FTR,GNP,HF,HFSUM,HP,HR,IP,OPS,OPSUN,P,
003741 PP,RAD,SP,TC,THP,TP,VC)

003774 PLOT
003777 ELSE AA
004003 SELT
004006 SELC
004011 DUM(Y)=AA(T,C,Y)
004031 XAXIS
004034 IF IVAR EQ 0
004050 SHOW SCATTER AA, XLABEL(Y E A R), YLABEL(NUMBER)
004071 END IF IVAR
004072 IF IVAR EQ 1
004106 SHOW SCATTER(IND2,DUM), XLABEL(IND2.L), YLABEL(NUMBER), TITLE(IND2.L,(VS),AA.L),
SUBTITLE( )
END IF IVAR

004142 PLOT
004143 ELSE HF
004146 SELT
004152 SELC
004155 DUM(Y)=HF(T,C,Y)
004160 XAXIS
004203 IF IVAR EQ 0
004203 SHOW SCATTER HF, XLABEL(Y E A R), YLABEL(NUMBER)
004217 END IF IVAR
004240 IF IVAR EQ 1
004241 SHOW SCATTER(IND2,DUM), XLABEL(IND2.L), YLABEL(NUMBER), TITLE(IND2.L,(VS),HF.L),
004255 SUBTITLE( )
END IF IVAR

004311 PLOT
004312 ELSE OPS
004315 SELT
004321 SELC
004327 DUM(Y)=OPS(T,C,Y)
004347 XAXIS
004352 IF IVAR EQ 0
004366 SHOW SCATTER OPS, XLABEL(Y E A R), YLABEL(NUMBER)
004407 END IF IVAR
004410 IF IVAR EQ 1
004424 SHOW SCATTER(IND2,DUM), XLABEL(IND2.L), YLABEL(NUMBER), TITLE(IND2.L,(VS),OPS.L),
SUBTITLE( )
END IF IVAR

004460 PLOT
004461 ELSE DATA=INDIR(AASUN,ATP,CP,HFSUM,HP,OPSUN,P,PP,SP,TP,HR,IP,THP)
004464 XAXIS
004506 IF IVAR EQ 0
004511 SHOW SCATTER INDIR, XLABEL(Y E A R), YLABEL(NUMBER), TITLE(INDIR.L)
004525 END IF IVAR
004552 IF IVAR EQ 1
004553 SHOW SCATTER(IND2,INDIR), XLABEL(IND2.L), YLABEL(NUMBER), TITLE(IND2.L,(VS),
004567

```

```

INDIP.L)
004621 END IF IVAR
004622 PLOT
004623 ELSE AUR
004624 SELT
004625
004626 SELC
004627 DUM(Y)=AUR(T,G,Y)
004628 XAXIS
004629 IF IVAR EQ 0
004630 SHOW SCATTER AUR, XLABEL(Y E A R), YLABEL(HRS/AIRCRAFT/YR)
004631 END IF IVAR
004632 IF IVAR EQ 1
004633 SHOW SCATTER(IND2,DUM), XLABEL(IND2.L), YLABEL(HRS/AIRCRAFT/YR),
004634 TITLE(IND2.L,(VS),AUR.L),SUBTITLE( )
004635 END IF IVAR
004636
004637 ELSE DATA=INDIR(PI,GMP,RAD)
004638 PLOT
004639 ELSE DATA=INDIR(PI,GMP,RAD)
004640 XAXIS
004641 IF IVAR EQ 0
004642 SHOW SCATTER INDIR.2, XLABEL(Y E A R), YLABEL(1972 $, 1972=1), TITLE(INDIR.L)
004643 END IF IVAR
004644 IF IVAR EQ 1
004645 SHOW SCATTER(IND2,INDIR.2), XLABEL(IND2.L), YLABEL(1972 $,1972=1),
004646 TITLE(IND2.L,(VS),INDIR.L)
004647 END IF IVAR
004648
004649 PLOT
004650 ELSE FC
004651 ASK(PLEASE ENTER EITHER 1 FOR AVIATION GAS, OR 2 FOR JET FUEL FOR THE PLOT),
004652 END
004653 ELSE INDEX=FUEL
004654 END ASK
004655 DUM(Y)=FC(F,Y)
004656 XAXIS
004657 IF IVAR EQ 0
004658 SHOW SCATTER FC, XLABEL(Y E A R), YLABEL(MILLIONS OF GALLONS)
004659 END IF IVAR
004660 IF IVAR EQ 1
004661 SHOW SCATTER(IND2,DUM), XLABEL(IND2.L), YLABEL(MILLIONS OF GALLONS),
004662 TITLE(IND2.L,(VS),FC.L),SUBTITLE( )
004663 END IF IVAR
004664
004665 PLOT
004666 ELSE FTR
004667 XAXIS
004668 IF IVAR EQ 0
004669 SHOW SCATTER FTR, XLABEL(Y E A R), YLABEL(MILLIONS OF DOLLARS)
004670 END IF IVAR
004671 IF IVAR EQ 1
004672 SHOW SCATTER(IND2,FTR), XLABEL(IND2.L), YLABEL(MILLIONS OF DOLLARS),
004673 TITLE(IND2.L,(VS),FTR.L)
004674 END IF IVAR
004675
004676 PLOT
004677 ELSE TC
004678 SELT
004679 SELC
004680 DUM(Y)=TC(T,G,Y)
004681 XAXIS

```

```

005505 IF IVAR EQ 0
005521 SHOW SCATTER TC.2, XLABEL(Y E A R), YLABEL(DOLLARS)
005542 END IF IVAR
005543 IF IVAR EQ 1
005557 SHOW SCATTER(IND2,DUM.2,XLABEL(IND2.L),YLABEL(DOLLARS),TITLE(IND2.L,( VS),TC.L,
      1, SUBTITLE( )
005611 END IF IVAR
005614 PLOT
005617 ELSE VC
005623 SELT
005626 DUM(Y)=VC(T,Y)
005644 XAXIS
005647 IF IVAR EQ 0
005663 SHOW SCATTER VC.2, XLABEL(Y E A R), YLABEL(DOLLARS PER HOUR)
005706 END IF IVAR
005707 IF IVAR EQ 1
005723 SHOW SCATTER(IND2,DUM.2), XLABEL(IND2.L), YLABEL(DOLLARS PER HOUR),
      TITLE(IND2.L,( VS),VC.L),SUBTITLE( )
005761 END IF IVAR
005762 PLOT
005765 ELSE FIX
005771 SELT
005774 DUM(Y)=FIX(T,Y)
006012 XAXIS
006015 IF IVAR EQ 0
006031 SHOW SCATTER FIX.2, XLABEL(Y E A R), YLABEL(DOLLARS PER HOUR)
006054 END IF IVAR
006055 IF IVAR EQ 1
006071 SHOW SCATTER(IND2,DUM.2), XLABEL(IND2.L), YLABEL(DOLLARS PER HOUR),
      TITLE(IND2.L,( VS),FIX.L),SUBTITLE( )
006127 END IF IVAR
006130 PLOT
006133 END ASK
006134 END PLOT
006135 COMMENT
006135 SAVE SHMPL, DATA=SEGI

```

```

001557 DATA SEG1 (SEGMENT $NTVT)
001573 READ DUMIND(1,2)
DATA
001575 READ DUMIND(1,2,2)
DATA
COMMENT
COMMENT *DEFINE THE BASELINE DATASETS ON DB3
DATA
001577 AA1 (TYPE,CATEG,YR1) TMS(13,30)
001577 AA2 (TYPE,CATEG,YR1) TMS(13,29)
001621 AASG1(YR1) TMS(13,31)
001641 APT1 (AOPS,YR1) TMS(13,52)
001662 ATP1 (YR1) TMS(13,24)
001702 AUR1 (TYPE,CATEG,YR1) TMS(13,32)
001724 CPl (YR1) TMS(13,35)
001744 FC1 (FUEL,YR1) TMS(13,37)
001765 FTP1 (DUMIND,YR1) TMS(13,39)
002006 HFM1 (TYPE,CATEG,YR1) TMS(13,40)
002030 HFS1 (YR1) TMS(13,41)
002050 HP1 (YR1) TMS(13,42)
002370 HPI (YR1) TMS(13,43)
002110 IPI (YR1) TMS(13,27)
002130 OPS1 (TYPE,CATEG,YR1) TMS(13,44)
002152 OPSG1(YR1) TMS(13,47)
002172 P1 (YR1) TMS(13,46)
002212 PP1 (YR1) TMS(13,43)
002232 SP1 (YR1) TMS(13,24)
002252 TC1 (TYPE,CATEG,YR1) TMS(13,49)
002274 THP1 (YR1) TMS(13,50)
002314 TP1 (YR1) TMS(12,30)
COMMENT
COMMENT *DEFINE THE COMPARABLE DATASETS ON DB2
DATA
002334 AA2 (TYPE,CATEG,YR1) TMS(12,29)
002356 AASG2(YR1) TMS(12,31)
002376 APT2 (AOPS,YR1) TMS(12,52)
002417 ATP2 (YR1) TMS(12,28)
002437 AUR2 (TYPE,CATEG,YR1) TMS(12,32)
002461 CP2 (YR1) TMS(12,35)
002501 FC2 (FUEL,YR1) TMS(12,37)
002522 FTP2 (DUMIND,YR1) TMS(12,39)
002543 HFM2 (TYPE,CATEG,YR1) TMS(12,40)
002565 HFS2 (YR1) TMS(12,41)
002605 HP2 (YR1) TMS(12,42)
002625 HPI2 (YR1) TMS(12,43)
002645 TP2 (YR1) TMS(12,44)
002665 OPS2 (TYPE,CATEG,YR1) TMS(12,47)
002707 OPSG2(YR1) TMS(12,46)
002727 P2 (YR1) TMS(12,43)
002747 PP2 (YR1) TMS(12,40)
002767 SP2 (YR1) TMS(12,24)
003007 TC2 (TYPE,CATEG,YR1) TMS(12,49)
003231 THP2 (YR1) TMS(12,50)
003351 TP2 (YR1)
COMMENT
COMMENT *DEFINE THE SCRATCH SENSITIVITY DATASETS

```

```

COMMENT
003071 AA (TYPE,CATEG,YR1),(ACTIVE AIRCRAFT BY PRIMARY USE DURING PREVIOUS YEAR, AS REP
003140 ORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE)
003140 APTS(OPS,YR1),(LOCAL AND ITINERANT OPERATIONS PLUS IFR FLIGHT PLANS FILED, AS P
003176 SRCENT DEVIATION FROM BASELINE)
003214 ATP (YR1),(AIR TRANSPORT PILOTS)
003214 AUR (TYPE,CATEG,YR1),(AIRCRAFT UTILIZATION RATE, AS PERCENT DEVIATION FROM BASE
LINE)
003245 CP (YR1),(COMMERCIAL PILOTS)
003263 FC (FUEL,YR1),(FUEL CONSUMED DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF 0
003325 ESIENATED YEAR, AS PERCENT DEVIATION FROM BASELINE)
003325 FTR (DUMIND,YR1),(FEDERAL TAX REVENUE DURING PREVIOUS YEAR, AS REPORTED ON JANUA
003371 RY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE)
003436 HF (TYPE,CATEG,YR1),(HOURS FLOWN (THOUSANDS) DURING PREVIOUS YEAR, AS REPORTED
003454 ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE)
003472 HP (YR1),(HELICOPTER PILOTS)
003502 HR (YR1),(HELICOPTER PILOTS)
003520 INDIR
003565 IP (YR1),(INSTRUMENT RATINGS)
003602 OPS (TYPE,CATEG,YR1),(OPERATIONS (THOUSANDS) DURING PREVIOUS YEAR, AS REPORTED 0
003617 N JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE)
003634 P (YR1),(PILOT SUBTOTAL)
003652 PP (YR1),(PRIVATE PILOTS)
003667 SP (YR1),(STUDENT PILOTS)
003705 THP (YR1),(TOTAL HELIC RATINGS)
003705 TP (YR1),(TOTAL PILOTS)
003723 OPSUM(YR1),(TOTAL OPERATIONS)
003740 HFSUM(YR1),(TOTAL HOURS FLOWN)
003766 AASUM(YR1),(TOTAL AIRCRAFT)
003766 IC (TYPE,CATEG,YR1),(TOTAL COST, AS PERCENT DEVIATION FROM BASELINE)
003766 END
COMMENT
COMMENT
COMMENT
COMMENT
003766 MODEL LOADD
003774 SYSTEM DR3=2,DR2=3
004002 IF SENS1 EQ 3
004316 SYSTEM DR3=3,DR2=2
004024 END IF
004025 IF NSIMB LT NSIM
004041 SET YR11=NSIMB
004052 END IF
004053 IF ITECH EQ 1
004067 SHOW TEXT=LEFT/(STEP 7 -- PRINT TABLES FOR SENSITIVITY ANALYSIS-1/)
004111 END IF
004112 ASK(00 YOU WANT TO SEE TABLES OF THE SENSITIVITY ANALYSIS RESULTS)NO
004137 ELSE YES
004143 LOAD SHMSA, XEO=PRINS
004147 END ASK
004150 IF ITECH EQ 1
004164 SHOW TEXT=LEFT/(STEP 8 -- PLOT THE RESULTS OF SENSITIVITY ANALYSIS-1/)
004207 END IF
004210 ASK(00 YOU WANT TO SEE PLOTS OF THE SENSITIVITY ANALYSIS RESULTS)NO
004234 ELSE YES
004240 LOAD SHPLS, XEO=PLOTS
004244 END ASK

```

0

084245 EMD MODEL LOAD

C

C

(SEGMENT SHMSA SHOWS TABLES OF SENSITIVITY RESULTS)

004246	DATA SEG2	
004271	READ ADPS(1,30,18)	
DATA	LOCAL OPERATIONS THOUSANDS	
DATA	ITINERANT OPERATIONS THOUSANDS	
DATA	IFR FLIGHT PLANS THOUSANDS	
004314	READ CATEG(1,25)	
DATA	BUSINESS	
DATA	CORPORATE	
DATA	PERSONAL	
DATA	AERIAL	
DATA	INSTRUCTIONAL	
DATA	AIR TAXI	
DATA	OTHER	
004368	READ CATEG(1,6,6)	
DATA	BUSH	
DATA	COPP	
DATA	PERSHL	
DATA	AERIAL	
DATA	INSTR	
DATA	AIR TX	
DATA	OTHER	
004377	READ FUEL(1,15)	
DATA	AV GAS	
DATA	JET FUEL	
004406	READ FUEL (1,8,8)	
DATA	AV GAS	
DATA	JET FUEL	
004413	READ TYPE(1,35)	
DATA	SNG-ENG. P..NON-AER	
DATA	SNG-ENG. P..AER.	
DATA	MULTI-ENG. P.	
DATA	TURBOPROP	
DATA	TURBOJET	
DATA	P. ENG.. HELICOPTER	
DATA	T. ENG.. HELICOPTER	
004475	READ TYPE(1,14,7)	
DATA	SINGL-FNON-AER	
DATA	SINGL-P AER	
DATA	MULTI- PISTON	
DATA	TURBO PROP	
DATA	TURBO JET	
DATA	PISTON HELIC	
DATA	TURBINE HELIC	

```

COMMENT      **
COMMENT      *DEFINE OUTPUT TABLES FOR SENSITIVITY RESULTS
COMMENT      *
004532      TABLE
004532      PLTID(YR1),DATA1//SP-2/RP-2/CP-2/ATP-2/P-2//HP-2//TP-2//IP-2/HR-2/,
004625      THP-2//1),SIZE(20,9),TITLE(PILCT DATA. PERCENT DEVIATION FROM BASELINE)
004664      TILT(YR1),DATA1//AASUM-2//HFSUM-2//OPSUM-2),SIZE(19,8),TITLE(TOTALS, AS PERCENT
          DEVIATION FROM BASELINE)
          ENO TABLE

```



```

*CHOOSE OUTPUT TABLES DESIRED FOR SENSITIVITY RESULTS
COMMENT
COMMENT
MODEL PRINS
DROP
ASKINWHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE) NONE
ELSE LIST
SHOW TEXT=LEFT(TABLE VARIABLES IN TABLE)/,
(AIRCRAFT1 ACTIVE AIRCRAFT BY YEAR)/,
(AIRCRAFT2 ACTIVE AIRCRAFT BY USER CATEGORY)/,
(AIRPORTS LOCAL AND ITINERANT OPERATIONS PLUS IFR FLIGHT PLANS FILED)/,
(AIRUTIL AIRCRAFT UTILIZATION RATES)/,
(FUEL FUEL CONSUMED)/,
(HOURSFLOWN HOURS FLOWN)/,
(OPERATIONS TOTAL OPERATIONS)/,
(PILOTS SP,PP,CP,ATP,P,HP,TP,IP,HR,IHP)/,
(REVENUE FEDERAL TAX REVENUE)/,
(TOTALS TOTAL AIRCRAFT, TOTAL HOURS FLOWN, TOTAL OPERATIONS)/
PRINS
005121 ELSE AIRUTIL
005124 AUR =AUR2-AUR1/AUR1*100
005130 SHOW(15,12) AUR.2,ORDEPCATEG,TYPE,YR1)
005171 PRINS
005205 ELSE PILOTS
005210 TP =TP2-TP1/TP1*100
005214 TWP =TWP2-TWP1/TWP1*100
005243 SP =SP2-SP1/SP1*100
005272 PP =PP2-PP1/PP1*100
005321 P =P2-P1/P1*100
005350 IP =IP2-IP1/IP1*100
005377 HP =HP2-HP1/HP1*100
005426 MP =MP2-MP1/MP1*100
005455 CP =CP2-CP1/CP1*100
005504 ATP =ATP2-ATP1/ATP1*100
005533 PRINS
005562 ELSE FUEL
005564 FC =FC2-FC1/FC1*100
005567 SHOW(10,15) FC.2,ORDER(YR1,FUEL)
005573 PRINS
005627 ELSE AIRCRAFT2
005642 AA =AA2-AA1/AA1*100
005645 SHOW(20,9) AA.2,ORDER(CATEG,TYPE,YR1)
005651 PRINS
005712 AA =AA2-AA1/AA1*100
005726 SHOW(20,9) AA.2,ORDER(YR1,TYPE,CATEG)
005731 PRINS
005731 AA =AA2-AA1/AA1*100
005735 SHOW(16,9) AA.2,ORDER(YR1,TYPE,CATEG)
005776 PRINS
006012 ELSE OPERATIONS
006015 OPS =OPS2-OPS1/OPS1*100
006021 SHOW(20,9) OPS.2,ORDER(CATEG,TYPE,YR1)
006362 PRINS
006376 ELSE HOURSFLOWN
006101 HF =HFM2-HFM1/HFM1*100
006105 SHOW(20,9) HF.2,ORDER(CATEG,TYPE,YR1)
006146 PRINS
006162

```

```

006165 ELSE AIRPORTS
006171 APTS=APT2-APT1/APT1*100
006225 SHOW(10,12) APTS-2,ORDER(YR1,A(PS)
006240 PRINS
006243 ELSE REVENUE
006247 FTR =FTR2-FTR1/FTR1*100
006303 SHOW(10,9) FTR-2,ORDER(YR1,DUMIND)
006316 PRINS
006321 ELSE TOTALS
006325 AASUM=AASG2-AASG1/AASG1*100
006354 OPSUM=OPSG2-OPSG1/OPSG1*100
006403 MFSUM=MFS2-MFS1/MFS1*100
006432 TLTB
006434 PRINS
006437 END ASK
006440 END MODEL PRINS
006441 SAVE SHMSA,DATA=SEG2

```

(SEGMENT SHPLS SHOWS PLOTS OF SENSITIVITY RESULTS)

```

004246 DATA SEG2
COMMENT *
COMMENT *READ IN THE STUB HEADINGS FOR THE INDEXES
COMMENT *
004271 READ CATEG(1,23)
DATA BUSINESS TRANSPORTATION
DATA CORPORATE TRANSPORTATION
DATA PERSONAL FLYING
DATA AERIAL APPLICATION
DATA INSTRUCTIONAL FLYING
DATA AIR TAXI
DATA OTHER
004335 READ FUEL (1,15)
DATA AVIATION GAS
DATA JET FUEL
004344 READ TYPE(1,35)
DATA SINGLE-ENGINE PISTON (NONAERIAL)
DATA SINGLE-ENGINE PISTON (AERIAL)
DATA MULTI-ENGINE PISTON
DATA TURBOPOP
DATA TURBOJET
DATA PISTON-ENGINE HELICOPTER
DATA TURBINE-ENGINE HELICOPTER
COMMENT *
COMMENT *BUILD THE MODELS THAT ALLOW THE USER TO SEE PLOTS OF HIS RESULTS
COMMENT *
COMMENT *MODEL DESCRIPTION
COMMENT *SELC SELECTS THE USER CATEGORY FOR A PLOT
COMMENT *SELT SELECTS THE AIRCRAFT TYPE FOR A PLOT
COMMENT *PLOTS LISTS ALL POSSIBLE PLOTS AND SHOWS THE REQUESTED ONES
COMMENT *
COMMENT *
COMMENT *
MODEL SELC
004426 ASK(SELCT USER CATEGORY, OR LIST) LIST
004434 SHOW TEXT=LEFT(1,USER CATEGORIES. . .)/,
004452 11 BUSINESS TRANSPORTATION)/,
12 CORPORATE TRANSPORTATION)/,
13 PERSONAL FLYING)/,
14 AERIAL APPLICATION)/,
15 INSTRUCTIONAL FLYING)/,
16 AIR TAXI)/,
17 OTHER))
004550 SELC
004553 ELSE IMCK=CATEG
004557 END ASK
004559 END MODEL SELC
COMMENT *
COMMENT *
MODEL SELT
004561 ASK(SELCT AIRCRAFT TYPE, OR LIST) LIST
004567 SHOW TEXT=LEFT(1,AIRCRAFT TYPES. . .)/,
004605 11 SINGLE-ENGINE PISTON, NONAERIAL)/,
12 SINGLE-ENGINE PISTON, AERIAL)/,
13 MULTI-ENGINE PISTON)/,

```

14 TURBOPROP//
15 TURBOJET//
16 PISTON-ENGINE HELICOPTER//
17 TURBINE-ENGINE HELICOPTER//
SEL
ELSE INDEX=TYPE
END ASK
END MODEL SELT

004712
004715
004721
004722

```

COMMENT
COMMENT
*MODEL PLOTS LISTS ALL POSSIBLE SENSITIVITY RESULTS AND SHOWS THE SELECTED ONES
COMMENT
MODEL PLOTS
004721 DROP
004723 ASKINHAT SENSITIVITY PLOT (WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
004733 I, NONE)
ELSE LIST
004763 YFULL DATA(AA, AASUM, ATP, CP, FC, FTR, HF, HFSUM, HP, HR, IP, OPS, OPSUM, PP, P, SP, TC, THP, TP)
004767 PLOTS
005014 ELSE DATA=INDIR(AA, HF, OPS, TC)
005017 SHOW SCATTER INDIR.2, XLABEL(Y E A R), YLABEL(PERCENT)
005033 SFLT
005033 SELC
005036 AA = AA2-AA1/AA1*100
005036 HF = HF2-HF1/HF1*100
005036 OPS = OPS2-OPS1/OPS1*100
005140 TC = TC2-TC1/TC1*100
005201 SHOW SCATTER INDIR.2, XLABEL(Y E A R), YLABEL(PERCENT)
005242 PLOTS
005263 ELSE FC
005266 ASKPLEASE ENTER EITHER 1 FOR AVIATION GAS OR 2 FOR JET FUEL FOR THE PLOT1, END
005272 ELSE INDEX=FUEL
005323 END ASK
005324 FC = FC2-FC1/FC1*100
005325 SHOW SCATTER FC.2, XLABEL(Y E A R), YLABEL(PERCENT)
005361 PLOTS
005402 ELSE DATA=INDIR(CP, FTR, HP, HR, IP, P, PP, SP, THP, TP, OPSUM, HFSUM, AASUM, ATP)
005405 ATP = ATP2-ATP1/ATP1*100
005439 CP = CP2-CP1/CP1*100
005457 FTR = FTR2-FTR1/FTR1*100
005506 HP = HP2-HP1/HP1*100
005542 HR = HR2-HR1/HR1*100
005571 IP = IP2-IP1/IP1*100
005620 P = P2-P1/P1*100
005647 PP = PP2-PP1/PP1*100
005676 SP = SP2-SP1/SP1*100
005725 THP = THP2-THP1/THP1*100
005754 TP = TP2-TP1/TP1*100
006003 OPSUM=OPS2-OPS1/OPS1*100
006032 HFSUM=HF2-HF1/HF1*100
006061 AASUM=AA2-AA1/AA1*100
006110 SHOW SCATTER INDIR.2, XLABEL(Y E A R), YLABEL(PERCENT)
006137 PLOTS
006160 END ASK
006163 END MODEL PLOTS
006164 COMMENT
006165 SAVE SHPLS, DATA=SEG2

```

001557 DATA SEG1
001567 SAVE GAO,REQ=MAIN

APPENDIX B. A SAMPLE BATCH RUN OF GAD

READY/
000017 LOAD 640

YOU ARE ENTERING THE GENERAL AVIATION DYNAMICS MODEL
CREATED AT BATTELLE COLUMBUS LABORATORIES.
WRITTEN IN THE MODELING LANGUAGE N U C L E U S .
IN THIS SESSION YOU WILL PROJECT CERTAIN LEVELS OF
GENERAL AVIATION ACTIVITY FOR THE YEARS 1977 TO 1987.

ENTER ENDING YEAR FOR SIMULATION

DATA 1987

STEP 1 --- WOULD YOU LIKE TO COMPUTE THE FORECAST WITH THE INITIAL ASSUMPTIONS UNCHANGED (YES OR NO) OR VIEW THE STEPS OF THIS

MODEL (TEACH)

DATA YES

DO YOU WANT TO SEE TABLES OF RESULTS OF THE FORECAST

DATA YES

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.

DATA LIST

IDENTIFIER DESCRIPTION

AIRCRAFT1 ACTIVE AIRCRAFT BY YEAR

AIRCRAFT2 ACTIVE AIRCRAFT BY USER CATEGORY

AIRPORTS LOCAL AND ITINERANT OPERATIONS PLUS IFR FLIGHT PLANS FILED

AIRUTIL AIRCRAFT UTILIZATION RATES

ECONOMIC DPI,GMP,RAD

FIXEDCOST FIXED COST

VARIABLECOST VARIABLE COST

FUEL FUEL CONSUMED IN MILLIONS OF GALLONS

HOURSFLOWN HOURS FLOWN IN THOUSANDS

OPERATIONS TOTAL OPERATIONS, IN THOUSANDS

PILCTS SP,PP,CP,ATP,P,HP,TP,IP,MR,THP

REVENUE FEDERAL TAX REVENUE

TOTALS TOTAL AIRCRAFT, TOTAL HOURS FLOWN, TOTAL OPERATIONS

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.

DATA TOTALS

READY/
000017 LOAD GAO

YOU ARE ENTERING THE GENERAL AVIATION DYNAMICS MODEL
CREATED AT BATTELLE COLUMBUS LABORATORIES.
WRITTEN IN THE MODELING LANGUAGE N U C L E U S .
IN THIS SESSION YOU WILL PROJECT CERTAIN LEVELS OF
GENERAL AVIATION ACTIVITY FOR THE YEARS 1977 TO 1987.

ENTER ENDING YEAR FOR SIMULATION

DATA 1987
STEP 1 -- WOULD YOU LIKE TO COMPUTE THE FORECAST WITH THE INITIAL ASSUMPTIONS UNCHANGED (YES OR NO) OR VIEW THE STEPS OF THIS
MODEL (TEACH)

DATA YES

DO YOU WANT TO SEE TABLES OF RESULTS OF THE FORECAST

DATA YES

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.

DATA LIST

IDENTIFIER DESCRIPTION

AIRCRAFT1 ACTIVE AIRCRAFT BY YEAR

AIRCRAFT2 ACTIVE AIRCRAFT BY USER CATEGORY

AIRPORTS LOCAL AND ITINERANT OPERATIONS PLUS IFR FLIGHT PLANS FILED

AIRUTIL AIRCRAFT UTILIZATION RATES

ECONOMIC MPI,GNP,RAD

FIXEDCOST FIXED COST

VARCOST VARIABLE COST

FUEL FUEL CONSUMED IN MILLIONS OF GALLONS

HOURSFLOWN HOURS FLOWN IN THOUSANDS

OPERATIONS TOTAL OPERATIONS, IN THOUSANDS

PILGTS SP,PP,CP,ATP,P,HP,TP,IP,HR,TMP

REVENUE FEDERAL TAX REVENUE

TOTALS TOTAL AIRCRAFT, TOTAL HOURS FLOWN, TOTAL OPERATIONS

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.

DATA TOTALS

TOTALS FOR AIRCRAFT, HOURS FLOWN AND OPERATIONS, 1977 TO 1987

	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986
TOTAL AIRCRAFT	175,130	184,633	201,114	215,740	228,036	242,947	266,532	294,001	327,164	357,622
TOTAL HOURS FLOWN (THOUSANDS)	35,850	36,932	40,560	42,896	45,162	48,238	53,646	59,025	66,312	71,641
TOTAL OPERATIONS	109,616	111,581	121,260	127,404	133,376	141,621	155,785	169,530	188,129	201,560

1987

TOTAL AIRCRAFT	379,702
TOTAL HOURS FLOWN (THOUSANDS)	75,337
TOTAL OPERATIONS	210,775

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA PILOTS

GENERAL AVIATION DYNAMICS MODEL PAGE 2

PILOT DATA, 1977 TO 1987

	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987
STUDENT PILOTS	188,801	183,794	183,654	183,176	182,384	182,471	183,122	181,626	180,463	178,248	174,891
PRIVATE PILOTS	309,005	323,821	335,104	344,608	352,437	359,682	363,763	368,310	371,781	374,399	375,960
COMMERCIAL PILOTS	187,801	189,699	192,068	195,342	199,209	203,591	208,388	213,191	217,972	222,700	227,353
AIR TRANSPORT PILOTS	45,072	47,784	50,879	53,766	56,656	59,549	62,466	65,414	68,401	71,413	74,447
PILOT SUBTOTAL	730,679	745,098	761,705	776,891	790,685	804,292	817,739	828,541	838,617	846,759	852,650
HELICOPTER PILOTS	4,804	4,333	3,940	3,608	3,322	3,074	2,855	2,628	2,403	2,187	1,983
TOTAL PILOTS	735,483	749,431	765,645	780,499	794,007	807,366	820,595	831,169	841,020	848,946	854,634
INSTRUMENT RATINGS	211,364	221,497	232,437	243,969	255,978	268,368	281,046	293,596	306,000	318,202	330,169
HELICOPTER RATINGS	23,012	24,395	25,739	27,052	28,347	29,633	30,916	32,203	33,493	34,786	36,081
TOTAL HELIC RATINGS	27,816	28,728	29,679	30,659	31,669	32,706	33,771	34,831	35,896	36,973	38,064

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA VARCOST

VARIABLE COST (\$/YR), (1972 \$, 1972=1)

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURRO PROP	TURRO JET	PISTON HELIC	TURBINE HELIC
1977	1.138	1.194	1.194	1.092	1.172	1.222	1.174
1978	1.150	1.206	1.206	1.103	1.183	1.234	1.185
1979	1.161	1.217	1.217	1.114	1.195	1.246	1.197
1980	1.172	1.229	1.229	1.125	1.206	1.258	1.208
1981	1.183	1.241	1.241	1.135	1.218	1.270	1.220
1982	1.195	1.253	1.253	1.146	1.230	1.283	1.232
1983	1.217	1.277	1.277	1.168	1.253	1.307	1.255
1984	1.240	1.300	1.300	1.189	1.276	1.331	1.278
1985	1.262	1.324	1.324	1.211	1.299	1.355	1.301
1986	1.285	1.347	1.347	1.233	1.322	1.379	1.325
1987	1.307	1.371	1.371	1.254	1.346	1.404	1.348

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA AIRCRAFT2

ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1977

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	175,130	138,057	6,824	21,320	2,486	1,938	2,753	1,752
BUSINESS	37,349	28,494	0	8,441	0	0	424	0
CORPORATE	10,283	1,612	0	4,570	1,975	1,582	0	544
PERSONAL	85,110	81,462	0	3,184	0	0	460	0
AERIAL	7,757	0	6,824	354	0	0	579	0
INSTRUCTIONAL	13,026	12,177	0	564	0	0	281	0
AIR TAXI	6,789	2,327	0	2,964	347	177	182	792
OTHER	14,816	11,995	0	1,235	164	179	827	416

1978

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	184,633	144,716	7,576	22,528	2,696	2,292	2,740	2,086
BUSINESS	41,898	32,159	0	9,207	0	0	502	0
CORPORATE	11,276	1,747	0	4,961	2,196	1,794	0	577
PERSONAL	87,017	83,434	0	3,244	0	0	339	0
AERIAL	8,560	0	7,576	396	0	0	588	0
INSTRUCTIONAL	12,449	11,636	0	531	0	0	282	0
AIR TAXI	7,167	2,323	0	3,057	334	279	160	1,014
OTHER	16,266	13,388	0	1,130	165	219	869	495

1979

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	281,114	157,615	7,841	25,003	3,051	2,469	3,046	2,088
BUSINESS	47,524	36,693	0	10,255	0	0	576	0
CORPORATE	12,370	1,931	0	5,303	2,444	2,030	0	663
PERSONAL	94,843	90,870	0	3,611	0	0	362	0
AERIAL	8,834	0	7,841	370	0	0	623	0
INSTRUCTIONAL	12,059	11,371	0	362	0	0	306	0
AIR TAXI	8,018	2,747	0	3,527	442	191	249	863
OTHER	17,465	14,003	0	1,555	165	249	930	562

ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1980

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	215,740	168,882	8,283	27,186	3,219	2,704	3,206	2,260
BUSINESS	52,483	40,573	0	11,277	0	0	634	0
CORPORATE	13,077	2,016	0	5,568	2,561	2,229	0	703
PERSONAL	99,364	95,177	0	3,809	0	0	378	0
AERIAL	9,332	0	8,283	391	0	0	658	0
INSTRUCTIONAL	12,162	11,493	0	365	0	0	304	0
AIR TAXI	8,954	3,067	0	3,938	494	213	278	964
OTHER	20,368	16,556	0	1,838	165	262	954	593

1981

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	228,036	178,403	8,712	28,928	3,363	2,904	3,344	2,382
BUSINESS	57,164	44,191	0	12,290	0	0	683	0
CORPORATE	13,725	2,103	0	5,795	2,679	2,405	0	743
PERSONAL	103,822	99,419	0	4,008	0	0	395	0
AERIAL	9,816	0	8,712	412	0	0	692	0
INSTRUCTIONAL	12,236	11,631	0	302	0	0	302	0
AIR TAXI	9,418	3,226	0	4,142	519	224	293	1,014
OTHER	21,857	17,833	0	1,979	165	276	979	625

1982

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	242,947	190,085	9,127	30,985	1,565	1,120	3,523	2,542
BUSINESS	62,492	48,313	0	13,429	0	0	750	0
CORPORATE	14,535	2,233	0	6,055	2,855	2,588	0	804
PERSONAL	110,007	105,295	0	4,305	0	0	418	0
AERIAL	10,283	0	9,127	431	0	0	725	0
INSTRUCTIONAL	12,335	11,741	0	291	0	0	302	0
AIR TAXI	9,994	3,359	0	4,351	545	235	308	1,065
OTHER	23,402	19,124	0	2,123	165	297	1,020	673

ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1983

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	266,532	208,519	9,526	34,261	4,015	3,468	3,866	2,877
BUSINESS	71,508	55,393	0	15,214	0	0	900	0
CORPORATE	16,149	2,537	0	6,533	3,265	2,867	0	946
PERSONAL	118,966	113,755	0	4,763	0	0	448	0
AERIAL	10,733	0	9,526	450	0	0	757	3
INSTRUCTIONAL	12,426	11,867	0	257	0	0	303	0
AIR TAXI	10,631	3,643	0	4,676	585	253	331	1,144
OTHER	26,120	21,324	0	2,368	165	348	1,128	787

1984

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	294,001	229,549	9,909	38,659	4,473	3,875	4,244	3,296
BUSINESS	82,265	63,804	0	17,384	0	0	1,077	0
CORPORATE	17,735	2,797	0	7,069	3,615	3,187	0	1,067
PERSONAL	127,104	121,444	0	5,183	0	0	477	0
AERIAL	11,165	0	9,909	468	0	0	787	0
INSTRUCTIONAL	12,565	12,013	0	248	0	0	305	0
AIR TAXI	12,523	4,291	0	5,508	689	298	389	1,348
OTHER	30,644	25,201	0	2,798	165	390	1,209	881

1985

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	327,164	255,096	10,276	43,922	5,035	4,365	4,691	3,778
BUSINESS	96,551	75,011	0	20,249	0	0	1,291	0
CORPORATE	19,763	3,143	0	7,729	4,083	3,578	0	1,229
PERSONAL	137,345	131,100	0	5,734	0	0	511	0
AERIAL	11,578	0	10,276	485	0	0	816	0
INSTRUCTIONAL	12,554	12,028	0	222	0	0	304	0
AIR TAXI	14,304	4,901	0	6,292	787	340	445	1,539
OTHER	35,069	28,912	0	3,211	165	447	1,324	1,010

ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1986

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	357,622	278,217	10,626	49,331	5,416	4,802	5,025	4,205
BUSINESS	109,210	84,758	0	22,983	0	0	1,439	0
CORPORATE	21,115	3,315	0	8,248	4,317	3,925	0	1,309
PERSONAL	144,661	137,997	0	6,124	0	0	539	0
AERIAL	11,972	0	10,626	502	0	0	844	0
INSTRUCTIONAL	12,544	12,028	0	211	0	0	304	0
AIR TAXI	16,955	5,809	0	7,457	934	403	527	1,825
OTHER	41,166	34,280	0	3,805	165	474	1,371	1,071

1987

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	379,702	295,055	10,958	53,313	5,614	5,121	5,209	4,431
BUSINESS	119,002	92,209	0	25,282	0	0	1,511	0
CORPORATE	21,980	3,401	0	8,597	4,435	4,197	0	1,349
PERSONAL	149,947	142,985	0	6,399	0	0	563	0
AERIAL	12,346	0	10,958	518	0	0	870	0
INSTRUCTIONAL	12,482	11,986	0	191	0	0	305	0
AIR TAXI	18,395	6,301	0	8,090	1,014	438	572	1,900
OTHER	45,549	38,172	0	4,237	165	486	1,387	1,102

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA FUEL

FUEL CONSUMED (MILLION GALLONS) DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

	TOTAL	AV GAS	JET FUEL
1977	0	J	0
1978	1,043	5,44	519
1979	1,154	576	578
1980	1,230	610	619
1981	1,300	644	656
1982	1,389	689	699
1983	1,550	758	792
1984	1,712	844	865
1985	1,929	956	973
1986	2,047	1,037	1,051
1987	2,197	1,091	1,103

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA REVENUE

FEDERAL TAX REVENUE DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1977	0
1978	86,311,587
1979	95,429,267
1980	101,973,351
1981	108,003,171
1982	115,494,205
1983	128,765,796
1984	142,703,905
1985	160,960,100
1986	175,016,900
1987	185,001,799

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.

DATA NONE

DO YOU WANT TO SEE PLOTS OF RESULTS OF THE FORECAST

DATA YES

WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE

DATA LIST

IDENTIFIER DESCRIPTION

AA NUMBER OF ACTIVE AIRCRAFT

AASUM TOTAL NUMBER OF AIRCRAFT

ATP AIRLINE TRANSPORT PILOTS

AUR AIRCRAFT UTILIZATION RATE (HRS/AC/YR)

CP COMMERCIAL PILOTS

DPI DISPOSABLE PERSONAL INCOME (1972 \$, 1972=1)

FC FUEL CONSUMED (MILLION GALLONS)

FIX FIXED COST INDEX (\$/HR), (1972 \$, 1972=1)

FTR FEDERAL TAX REVENUE (MILLION DOLLARS)

GMP GROSS NATIONAL PRODUCT (1972 \$, 1972=1)

HF HOURS FLOWN (THOUSANDS)

HFSUM TOTAL HOURS FLOWN (THOUSANDS)

HP HELICOPTER PILOTS

HR HELICOPTER RATINGS

IP INSTRUMENT RATINGS

OPS OPERATIONS (THOUSANDS)

OPSLM TOTAL OPERATIONS (THOUSANDS)

P PILOT SURTOTAL

PP PRIVATE PILOTS

RAD REVENUE AIRCRAFT DEPARTURES (1972 \$, 1972=1)

SP STUDENT PILOTS

TC TOTAL COST

THP TOTAL HELICOPTER RATINGS

TP TOTAL PILOTS

VC VARIABLE COST INDEX (\$/HR), (1972 \$, 1972=1)

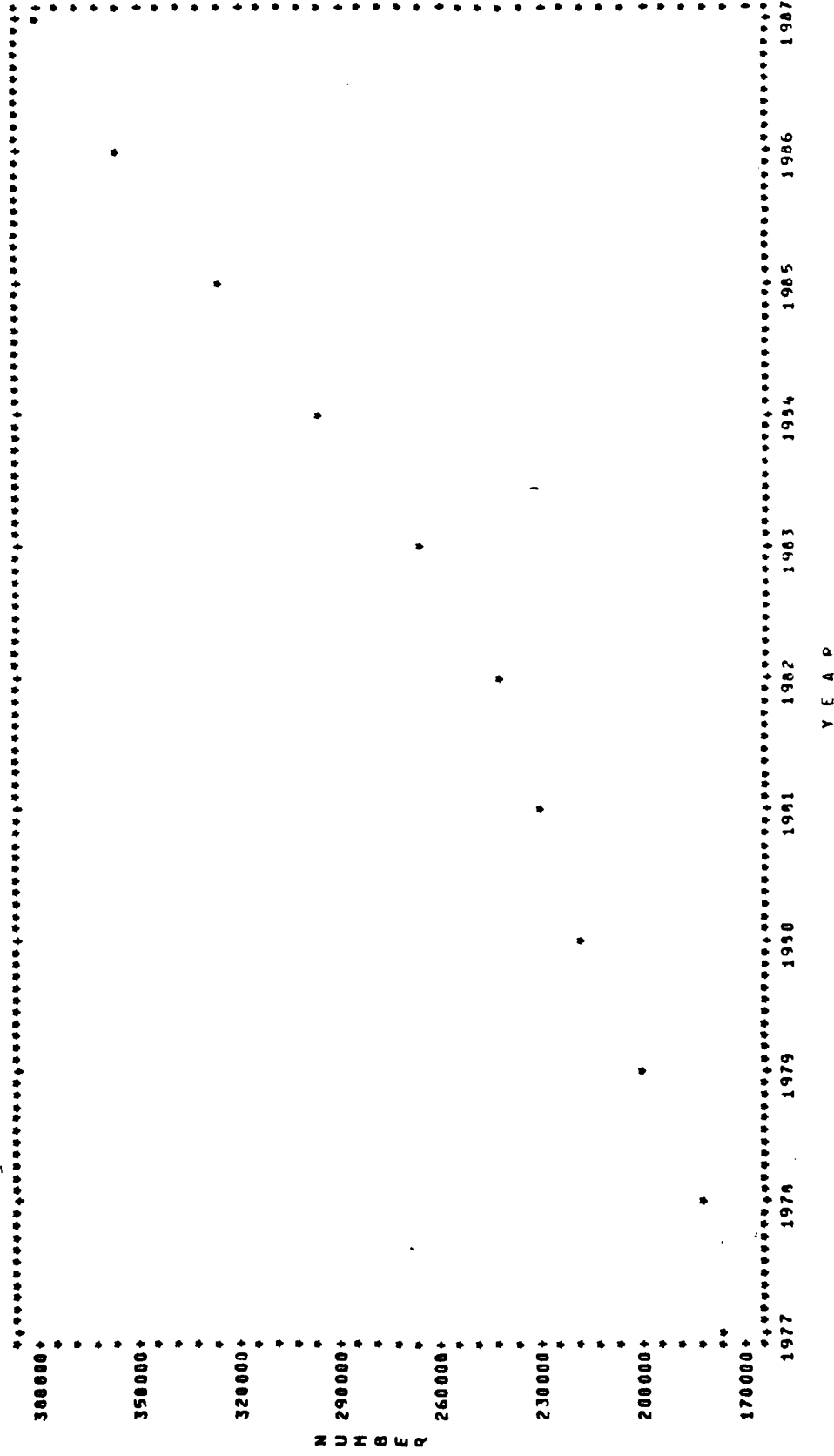
WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE

DATA AASUM

PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OR LIST

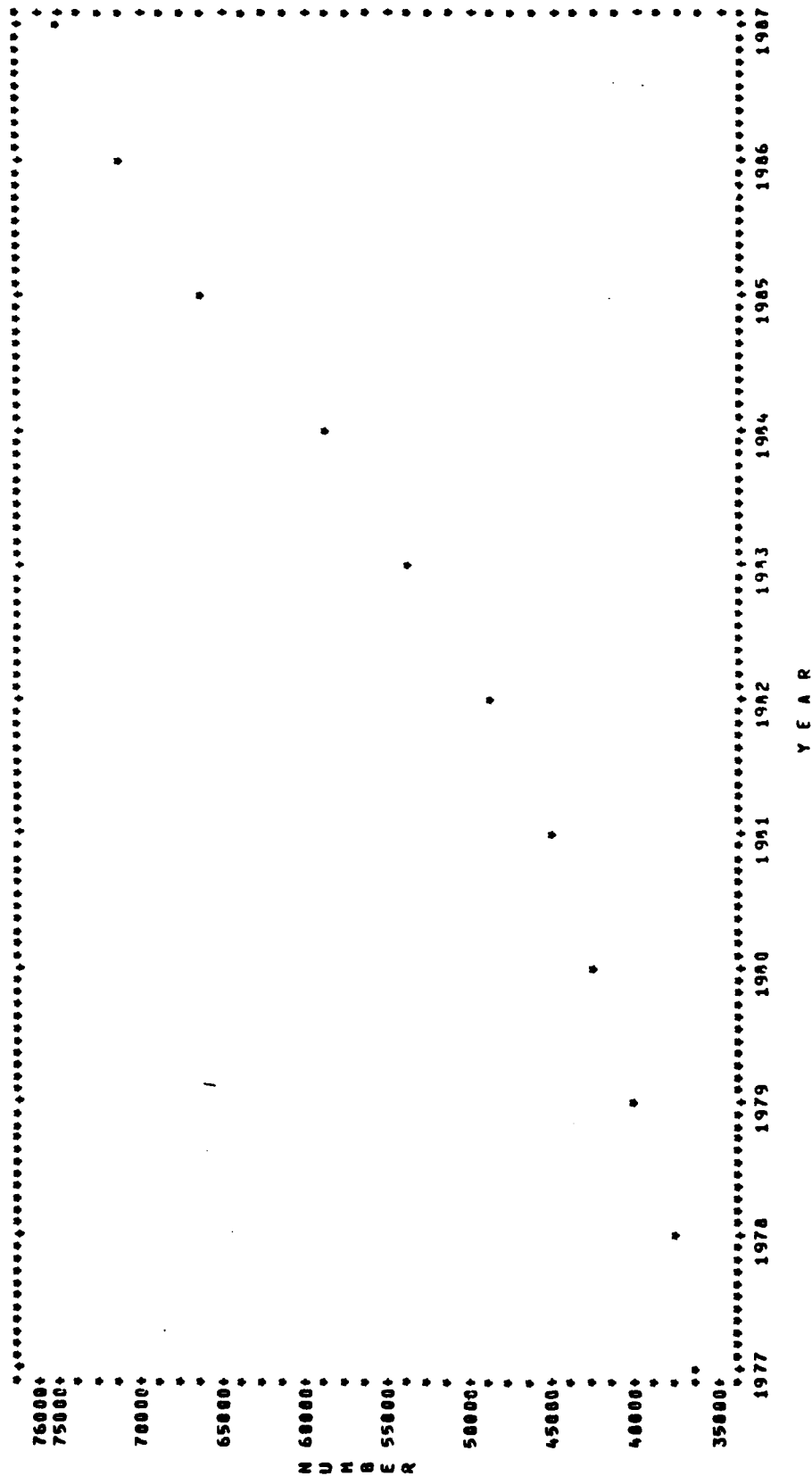
DATA TIME

TOTAL NUMBER OF AIRCRAFT, 1977 TO 1987



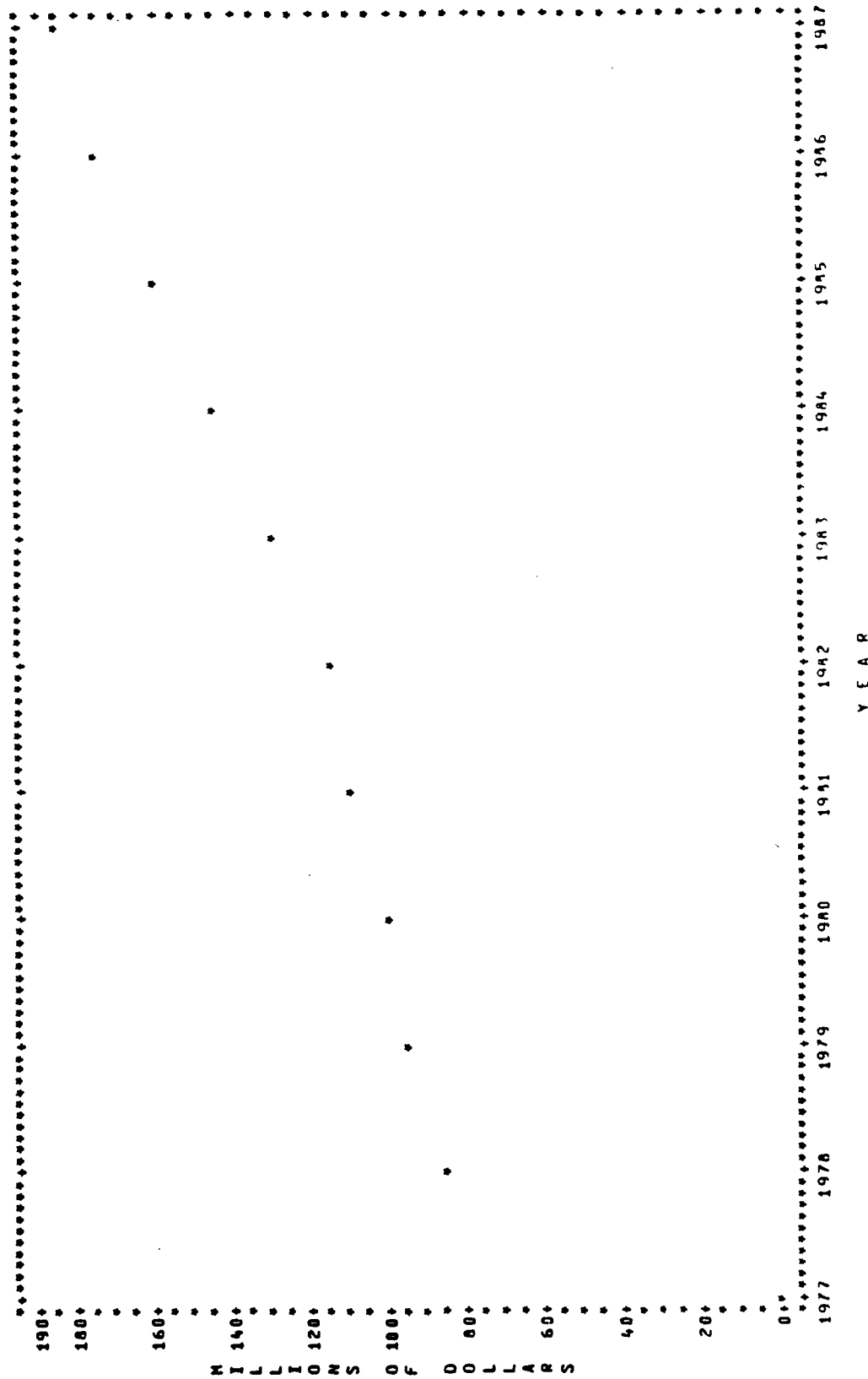
WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
 DATA *FSUM
 PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OR LIST
 DATA TIME

TOTAL HOURS FLOWN (THOUSANDS), 1977 TO 1987



WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
 DATA FTP
 PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OR LIST
 DATA TIME

FEDERAL TAX REVENUE (MILLION DOLLARS). 1977 TO 1987



WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
 DATA NONE
 WOULD YOU LIKE TO SAVE THE RESULTS OF THIS SESSION FOR LATER SENSITIVITY ANALYSIS?
 DATA YES
 WOULD YOU LIKE TO CONTINUE WITH ANOTHER FORECAST

DATA YES
 ENTER ENDING YEAR FOR SIMULATION
 DATA 1987
 STEP 1 -- WOULD YOU LIKE TO COMPUTE THE FORECAST WITH THE INITIAL ASSUMPTIONS UNCHANGED (YES OR NO) OR VIEW THE STEPS OF THIS
 MODEL (TEACH)
 DATA NO
 ENTER NAME OF VARIABLE TO BE CHANGED, OR LIST, OR NONE
 DATA VC
 THE COMPONENTS OF THE VARIABLE COST INDEX ARE FTAX, THE FEDERAL FUEL TAX, AND LFEE, THE LANDING FEE
 THE CURRENT VALUES FOR THE VARIABLE COST INDEX COMPONENTS ARE
 FEDERAL FUEL TAX (\$/GAL)

	AV GAS	JET FUEL
1977	0.07	0.07
1978	0.07	0.07
1979	0.07	0.07
1980	0.07	0.07
1981	0.07	0.07
1982	0.07	0.07
1983	0.07	0.07
1984	0.07	0.07
1985	0.07	0.07
1986	0.07	0.07
1987	0.07	0.07

[illegible]

DO YOU WISH TO CHANGE THE VALUES OF FTAX, LFEE OR NONE
DATA NONE
ENTER NAME OF VARIABLE TO BE CHANGED, OR LIST, OR NONE
DATA NONE
DO YOU WANT TO SEE TABLES OF RESULTS OF THE FORECAST
DATA YES
WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA TOTALS

GENERAL AVIATION DYNAMICS MODEL PAGE 13

TOTALS FOR AIRCRAFT, HOURS FLOWN AND OPERATIONS, 1977 TO 1987

	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986
TOTAL AIRCRAFT	175,130	184,633	201,114	215,745	227,752	242,071	264,638	290,619	321,737	349,894
TOTAL HOURS FLOWN (THOUSANDS)	35,850	36,932	40,560	42,703	44,768	47,609	52,720	57,749	64,712	69,698
TOTAL OPERATIONS	189,616	111,581	121,260	126,615	131,709	138,988	152,053	164,606	182,293	194,794

1987

TOTAL AIRCRAFT	369,648
TOTAL HOURS FLOWN (THOUSANDS)	73,059
TOTAL OPERATIONS	203,145

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA PILOTS

PILOT DATA, 1977 TO 1987

	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987
STUDENT PILOTS	188,801	183,794	183,654	179,129	173,764	169,641	166,663	162,346	161,208	159,963	157,882
PRIVATE PILOTS	309,095	323,821	335,104	345,428	353,589	359,516	363,737	366,988	368,400	369,079	369,075
COMMERCIAL PILOTS	187,801	189,699	192,068	194,521	196,921	199,283	201,589	203,557	205,771	208,148	210,652
AIR TRANSPORT PILOTS	45,872	47,784	50,879	53,766	56,638	59,482	62,387	65,112	67,898	70,658	73,196
PILOT SUBTOTAL	730,679	745,098	761,705	772,844	780,911	787,922	794,297	798,004	803,278	807,849	811,006
HELICOPTER PILOTS	4,884	4,333	3,940	3,534	3,145	2,789	2,470	2,169	1,917	1,705	1,524
TOTAL PILOTS	735,493	749,431	765,645	776,378	784,056	790,710	796,767	800,172	805,195	809,554	812,530
INSTRUMENT RATINGS	211,364	221,497	232,437	243,148	253,681	264,011	274,100	283,648	293,231	302,741	312,146
HELICOPTER RATINGS	23,812	24,395	25,739	27,052	28,136	29,593	30,822	32,025	33,198	34,346	35,472
TOTAL HELIC RATINGS	27,816	28,728	29,679	30,586	31,481	32,281	33,292	34,194	35,115	36,051	36,996

AD-A085 007

BATTELLE COLUMBUS LABS OH P/B 1/3
THE GENERAL AVIATION DYNAMICS MODEL VOLUME III. SYSTEMS MANUAL.(U)
JUL 79 M A DUFFY: J H MCCREERY DOT-FA77WA-4843

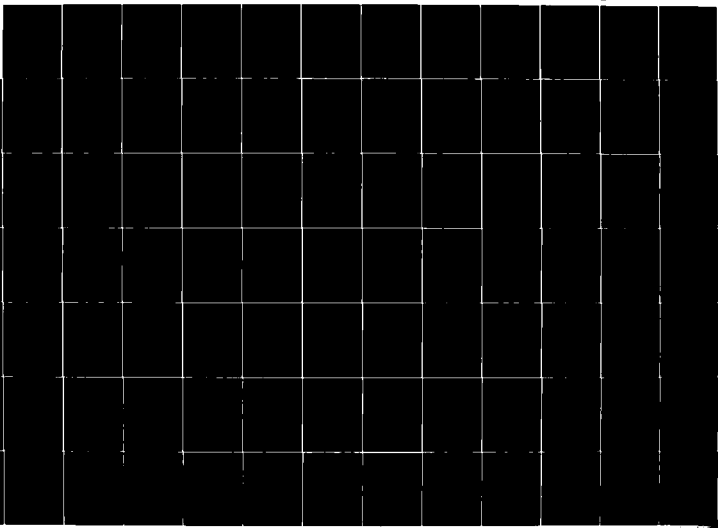
UNCLASSIFIED

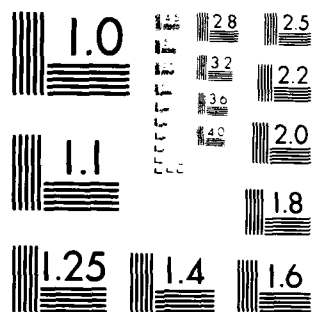
FAA-AVP-79-8-VOL-3

NL

2 of 4

AD
A085007





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA VARCOST

VARIABLE COST (\$/YR), (1972 \$, 1972=1)

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
1977	1.138	1.194	1.194	1.092	1.172	1.222	1.174
1978	1.150	1.206	1.206	1.103	1.183	1.234	1.185
1979	1.191	1.229	1.244	1.138	1.247	1.267	1.215
1980	1.229	1.251	1.280	1.171	1.304	1.298	1.242
1981	1.264	1.272	1.313	1.201	1.357	1.327	1.268
1982	1.296	1.292	1.344	1.228	1.404	1.354	1.292
1983	1.338	1.323	1.385	1.266	1.461	1.392	1.327
1984	1.355	1.344	1.403	1.282	1.474	1.412	1.347
1985	1.372	1.366	1.422	1.300	1.488	1.432	1.367
1986	1.390	1.388	1.441	1.318	1.503	1.453	1.397
1987	1.407	1.409	1.461	1.335	1.518	1.474	1.407

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA AIRCRAFT2

ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1977

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	175,130	138,057	6,824	21,320	2,486	1,938	2,753	1,752
BUSINESS	37,349	28,694	0	8,441	0	0	424	0
CORPORATE	10,243	1,612	0	4,570	1,975	1,562	0	544
PERSONAL	85,110	81,462	0	3,184	0	0	460	0
AERIAL	7,757	0	6,824	354	0	0	579	0
INSTRUCTIONAL	13,026	12,177	0	564	0	0	281	0
AIR TAXI	6,749	2,327	0	2,964	347	177	182	792
OTHER	14,816	11,995	0	1,235	164	179	827	416

1978

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	184,633	144,716	7,576	22,528	2,696	2,292	2,740	2,086
BUSINESS	41,498	32,189	0	9,207	0	0	502	0
CORPORATE	11,276	1,747	0	4,961	2,196	1,794	0	577
PERSONAL	87,317	83,434	0	3,244	0	0	339	0
AERIAL	8,560	0	7,576	396	0	0	588	0
INSTRUCTIONAL	12,449	11,536	0	531	0	0	282	0
AIR TAXI	7,167	2,323	0	3,057	334	279	160	1,014
OTHER	16,266	13,388	0	1,130	165	219	869	495

1979

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	201,114	157,615	7,841	25,003	3,051	2,469	3,046	2,080
BUSINESS	47,524	36,693	0	10,255	0	0	576	0
CORPORATE	12,370	1,931	0	5,303	2,444	2,030	0	663
PERSONAL	94,843	90,270	0	1,611	0	0	362	0
AERIAL	8,834	0	7,841	370	0	0	623	0
INSTRUCTIONAL	12,059	11,371	0	382	0	0	306	0
AIR TAXI	8,018	2,747	0	3,527	442	191	249	863
OTHER	17,465	14,003	0	1,555	165	249	930	562

ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1980

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	215,745	168,085	8,283	27,187	3,221	2,704	3,206	2,260
BUSINESS	52,483	48,573	0	11,277	0	0	634	0
CORPORATE	13,079	2,016	0	5,569	2,562	2,229	0	703
PERSONAL	99,364	95,177	0	3,809	0	0	378	0
AERIAL	9,332	0	8,283	391	0	0	658	0
INSTRUCTIONAL	12,165	11,496	0	365	0	0	304	0
AIR TAXI	8,954	3,067	0	3,934	494	213	278	964
OTHER	20,368	16,556	0	1,830	165	262	954	593

1981

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	227,750	178,144	8,712	28,922	3,365	2,905	3,328	2,382
BUSINESS	57,157	44,191	0	12,291	0	0	676	0
CORPORATE	13,727	2,103	0	5,795	2,680	2,405	0	744
PERSONAL	103,821	99,419	0	4,004	0	0	394	0
AERIAL	9,816	0	8,712	412	0	0	692	0
INSTRUCTIONAL	11,963	11,372	0	296	0	0	295	0
AIR TAXI	9,418	3,226	0	4,142	519	224	293	1,014
OTHER	21,857	17,833	0	1,979	165	276	978	625

1982

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	242,071	189,267	9,127	30,954	3,568	3,122	3,492	2,542
BUSINESS	62,418	48,262	0	13,420	0	0	735	0
CORPORATE	14,537	2,233	0	6,053	2,857	2,589	0	804
PERSONAL	109,795	105,894	0	4,296	0	0	415	0
AERIAL	10,283	0	9,127	431	0	0	725	0
INSTRUCTIONAL	11,742	11,174	0	279	0	0	290	0
AIR TAXI	9,494	3,389	0	4,351	545	235	308	1,065
OTHER	23,402	19,124	0	2,123	165	297	1,020	673

ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1983

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	264,638	206,764	9,526	34,166	4,018	3,471	3,817	2,877
BUSINESS	71,237	55,190	0	15,174	0	0	873	0
CORPORATE	16,151	2,537	0	6,529	3,268	2,870	0	946
PERSONAL	118,283	113,105	0	4,735	0	0	443	0
AERIAL	10,733	0	9,526	450	0	0	757	0
INSTRUCTIONAL	11,527	11,004	0	239	0	0	286	0
AIR TAXI	18,631	3,643	0	4,676	585	253	331	1,144
OTHER	26,076	21,246	0	2,364	165	344	1,127	786

1984

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	290,619	226,438	9,909	38,449	4,473	3,878	4,175	3,296
BUSINESS	81,620	63,303	0	17,274	0	0	1,038	0
CORPORATE	17,736	2,797	0	7,063	3,619	3,190	0	1,067
PERSONAL	125,699	120,103	0	5,126	0	0	470	0
AERIAL	11,165	0	9,909	469	0	0	787	0
INSTRUCTIONAL	11,389	10,884	0	223	0	0	243	0
AIR TAXI	12,523	4,291	0	5,501	649	298	389	1,348
OTHER	30,487	25,861	0	2,782	165	390	1,268	881

1985

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	321,737	250,144	10,276	43,530	5,039	4,369	4,601	3,778
BUSINESS	95,273	74,008	0	20,029	0	0	1,236	0
CORPORATE	19,763	3,143	0	7,722	4,016	3,582	0	1,229
PERSONAL	134,949	128,813	0	5,634	0	0	502	0
AERIAL	11,578	0	10,276	485	0	0	816	0
INSTRUCTIONAL	11,151	10,680	0	102	0	0	279	0
AIR TAXI	14,304	4,901	0	6,292	747	340	445	1,539
OTHER	34,719	28,599	0	3,176	165	447	1,323	1,009

ACTIVE AIRCRAFT BY PRIMARY USE, DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

1986

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	349,494	271,214	10,626	48,697	5,419	4,807	4,926	4,205
BUSINESS	107,110	83,127	0	22,605	0	0	1,378	0
CORPORATE	21,115	3,315	0	8,241	4,320	3,930	0	1,310
PERSONAL	141,120	134,614	0	5,974	0	0	528	0
AERIAL	11,972	0	10,626	502	0	0	844	0
INSTRUCTIONAL	11,114	10,656	0	179	0	0	279	0
AIR TAXI	16,955	5,809	0	7,457	934	403	527	1,825
OTHER	40,508	33,689	0	3,740	165	474	1,370	1,070

1987

	TOTAL	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
TOTAL	369,648	286,001	10,958	52,413	5,617	5,126	5,102	4,431
BUSINESS	116,066	89,417	0	24,723	0	0	1,446	0
CORPORATE	21,980	3,401	0	9,589	4,438	4,202	0	1,350
PERSONAL	145,314	138,567	0	6,201	0	0	549	0
AERIAL	12,346	0	10,958	518	0	0	870	0
INSTRUCTIONAL	11,086	10,649	0	158	0	0	279	0
AIR TAXI	18,395	6,301	0	8,090	1,014	438	572	1,980
OTHER	44,518	37,245	0	4,134	165	486	1,386	1,101

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA FUEL

FUEL CONSUMED (MILLION GALLONS) DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

	TOTAL	AV GAS	JET FUEL
1977	0	0	0
1978	1.043	524	519
1979	1.154	576	578
1980	1.220	609	612
1981	1.282	639	642
1982	1.361	681	680
1983	1.512	756	756
1984	1.663	831	833
1985	1.874	934	940
1986	2.028	1,009	1,018
1987	2.133	1,061	1,071

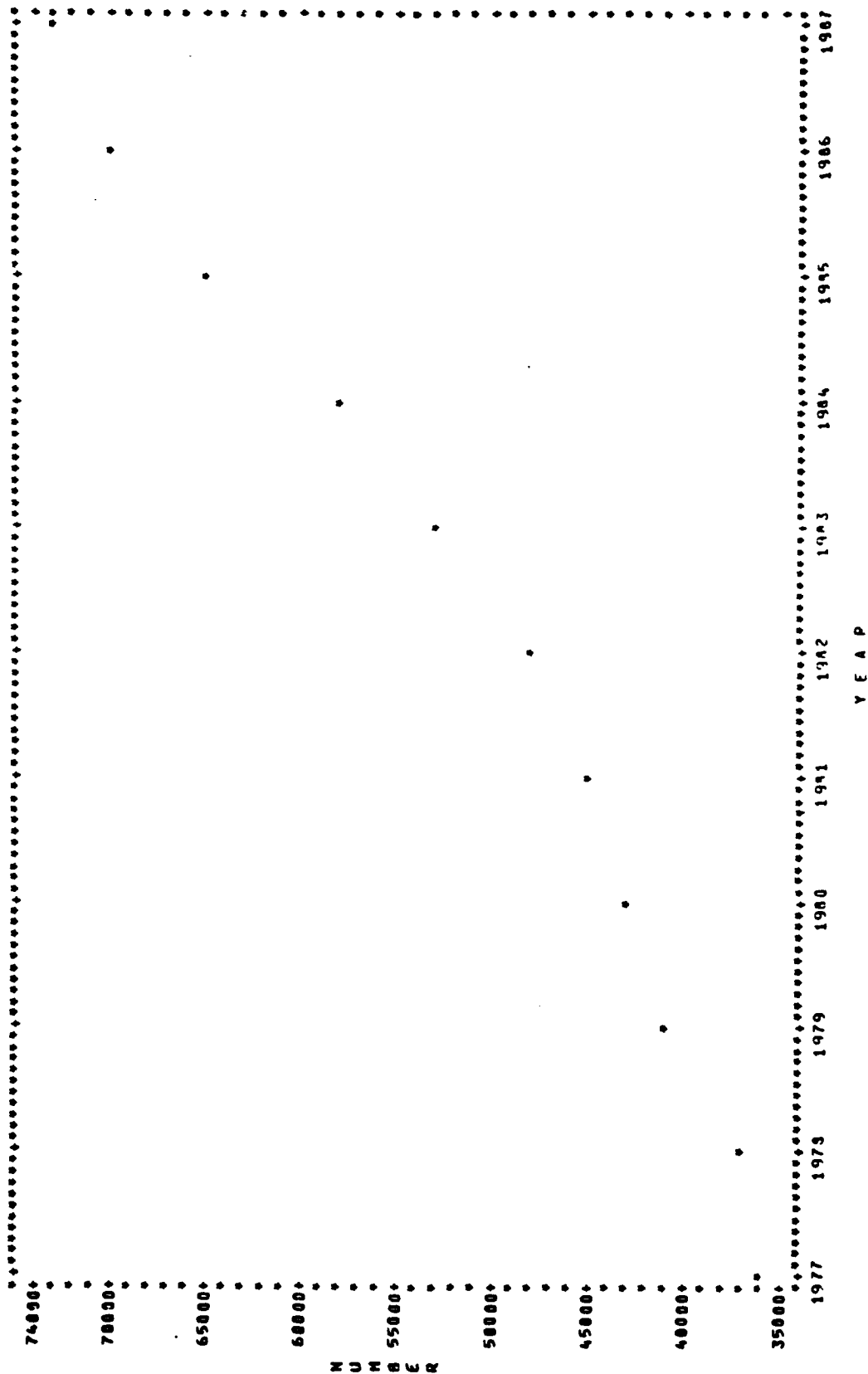
DATA INPUT TABLE WOULD YOU LIKE. OR ENTER LIST OR NONE.
DATA REVENUE

FEDERAL TAX REVENUE DURING PREVIOUS YEAR AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR

	C
1977	86,311,587
1978	95,424,267
1979	162,313,434
1980	234,867,765
1981	317,703,243
1982	428,391,026
1983	554,891,249
1984	625,379,162
1985	677,236,744
1986	713,013,243
1987	

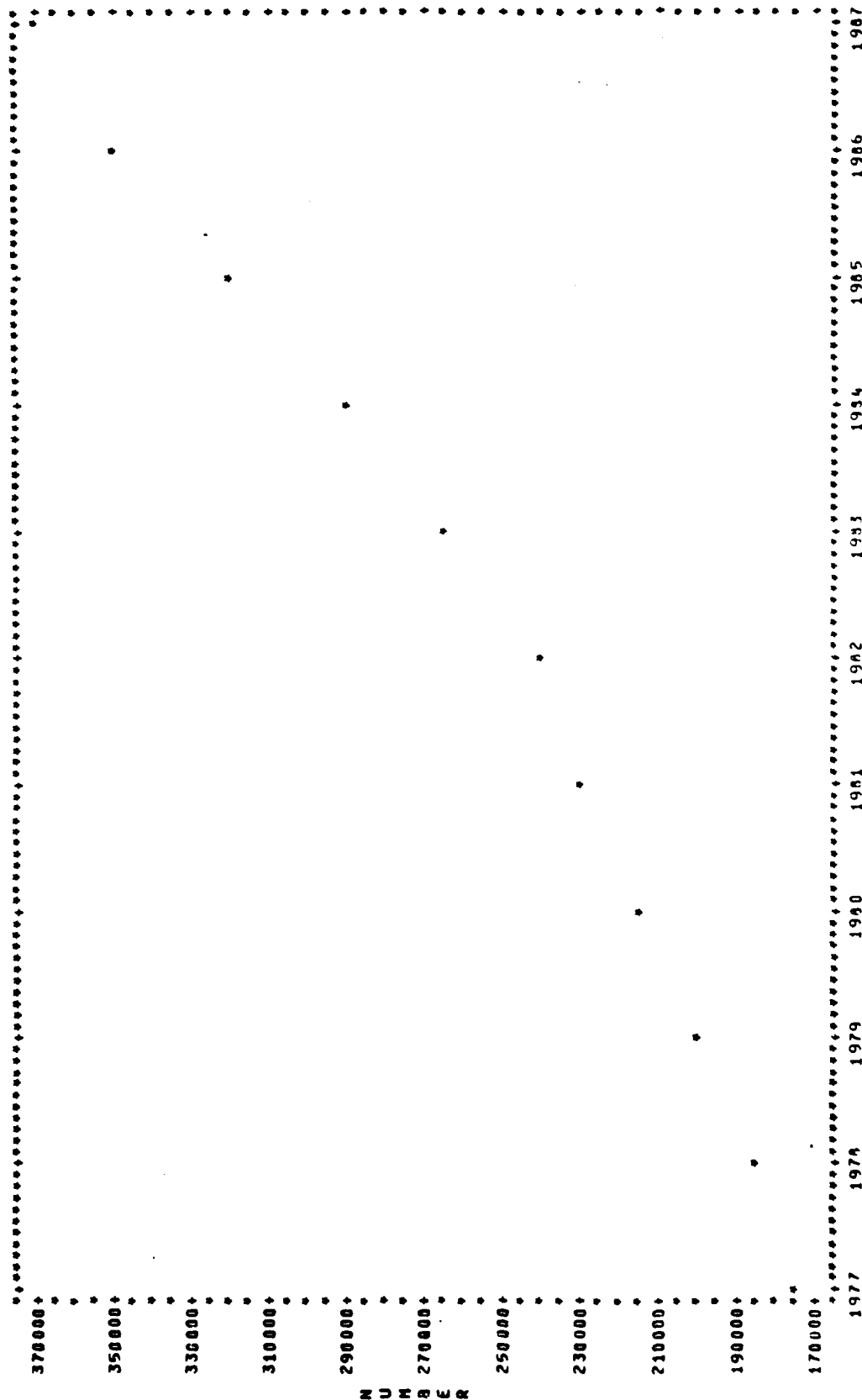
WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
DATA NONE
DO YOU WANT TO SEE PLOTS OF RESULTS OF THE FORECAST
DATA YES
WHAT PLCT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
DATA AASUM
PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OR LIST
DATA TIME

TOTAL HOURS FLOWN (THOUSANDS), 1977 TO 1987



WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
 DATA
 PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OF LIST
 DATA TIME

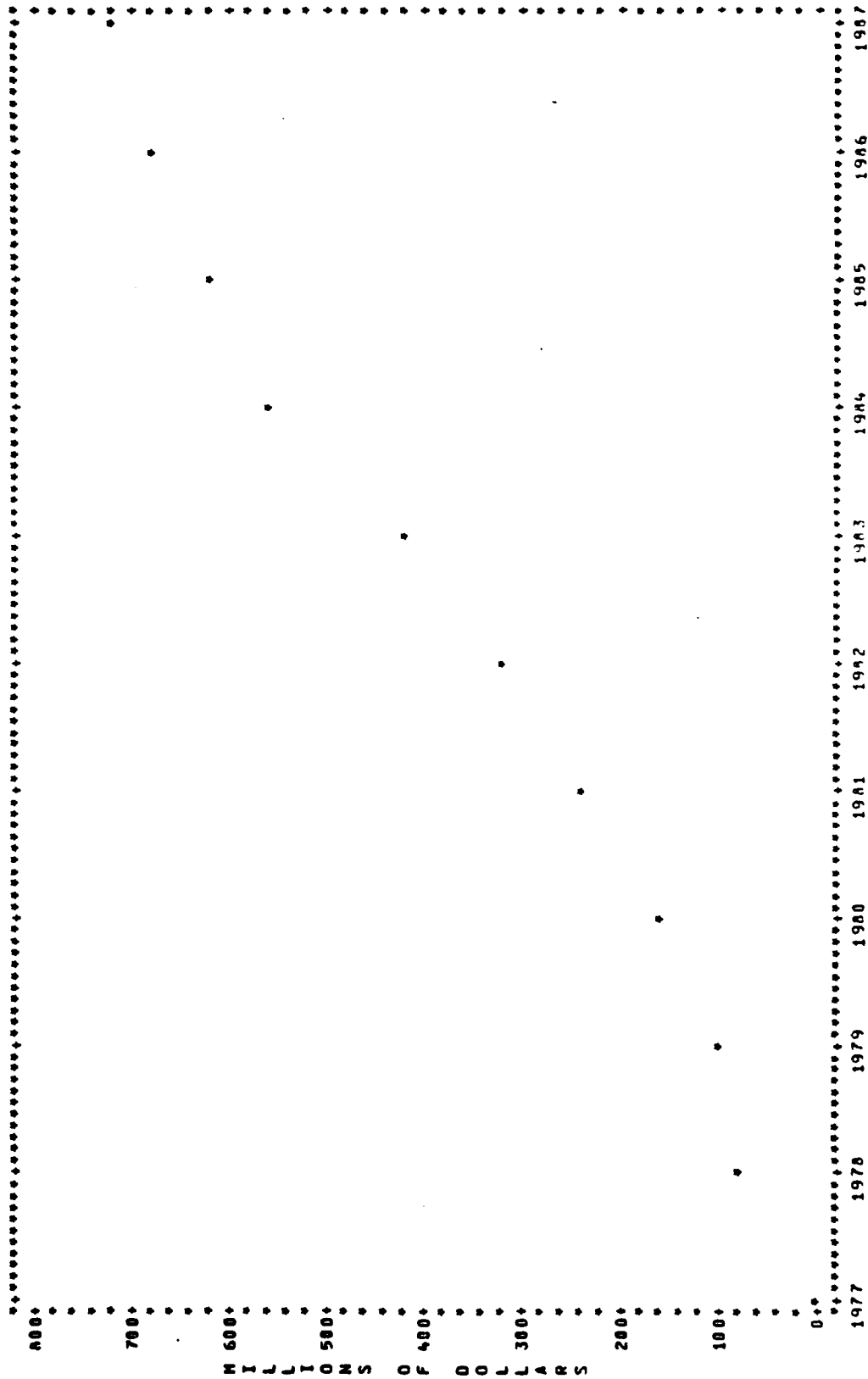
TOTAL NUMBER OF AIRCRAFT, 1977 TO 1987



Y C A R

WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
 DATA HFSUM
 PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OR LIST
 DATA TIME

FEDERAL TAX REVENUE (MILLION DOLLARS), 1977 TO 1987



Y E A R

WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
 DATA MORE
 DO YOU WANT SENSITIVITY ANALYSIS, (THE PREVIOUSLY SAVED FORECAST IS THE BASELINE), (YES OR NO)
 DATA YES
 DO YOU WANT TO SEE TABLES OF THE SENSITIVITY ANALYSIS RESULTS

DATA YES
 WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
 DATA LIST
 TABLE VARIABLES IN TABLE
 AIRCRAFT1 ACTIVE AIRCRAFT BY YEAR
 AIRCRAFT2 ACTIVE AIRCRAFT BY USER CATEGORY
 AIRPORTS LOCAL AND ITINERANT OPERATIONS PLUS IFR FLIGHT PLANS FILED
 AIRUTIL AIRCRAFT UTILIZATION RATES
 FUEL FUEL CONSUMED
 HOURSFLCWN HOURS FLOWN
 OPERATIONS TOTAL OPERATIONS
 PILOTS SP,PR,CP,ATP,P,HP,TP,IP,HR,THP
 REVENUE FEDERAL TAX REVENUE
 TOTALS TOTAL AIRCRAFT, TOTAL HOURS FLOWN, TOTAL OPERATIONS
 WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
 DATA TOTALS

TOTALS, AS PERCENT DEVIATION FROM BASELINE, 1977 TO 1987

	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987
TOTAL AIRCRAFT	0.00	0.00	0.00	0.00	-0.12	-0.16	-0.71	-1.15	-1.66	-2.16	-2.65
TOTAL HOURS FLOWN	0.00	0.00	0.00	-0.45	-0.87	-1.10	-1.73	-2.16	-2.40	-2.71	-3.02
TOTAL OPERATIONS	0.00	0.00	0.00	-0.62	-1.25	-1.16	-2.40	-2.90	-3.10	-3.36	-3.62

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
DATA PILOTS

PILOT DATA. PERCENT DEVIATION FROM BASELINE, 1977 TO 1987

	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987
STUDENT PILOTS	0.00	0.00	0.00	-2.21	-4.73	-7.03	-0.99	-10.62	-10.67	-10.26	-9.73
PRIVATE PILOTS	0.00	0.00	0.00	0.24	0.33	0.23	-0.01	-0.36	-0.91	-1.42	-1.03
COMMERCIAL PILOTS	0.00	0.00	0.00	-0.42	-1.15	-2.12	-3.26	-4.52	-5.60	-6.53	-7.35
AIR TRANSPORT PILOTS	0.00	0.00	0.00	0.00	-0.03	-0.11	-0.25	-0.46	-0.74	-1.06	-1.41
PILOT SUBTOTAL	0.00	0.00	0.00	-0.52	-1.24	-2.04	-2.87	-3.69	-4.21	-4.60	-4.88
HELICOPTER PILOTS	0.00	0.00	0.00	-2.05	-5.34	-9.29	-13.48	-17.47	-20.21	-22.03	-23.17
TOTAL PILOTS	0.00	0.00	0.00	-0.53	-1.25	-2.06	-2.90	-3.73	-4.26	-4.64	-4.93
INSTRUMENT RATINGS	0.00	0.00	0.00	-0.34	-0.90	-1.62	-2.47	-3.39	-4.17	-4.86	-5.46
HELICOPTER RATINGS	0.00	0.00	0.00	0.00	-0.04	-0.13	-0.30	-0.55	-0.88	-1.27	-1.69
TOTAL HELIC RATINGS	0.00	0.00	0.00	-0.24	-0.59	-0.99	-1.42	-1.83	-2.18	-2.49	-2.81

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
DATA AIRCRAFT2

[illegible]

	SNG-L-P NON-AER	SNG-L-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CORPORATE	0.00	0.00	3.00	0.00	0.00	0.00	0.00
PERSONAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AIR TAXI	0.00	0.20	0.00	0.00	0.00	0.00	0.00
OTHER	0.00	0.00	3.00	3.00	0.00	0.00	0.00

[illegible]

ACTIVE AIRCRAFT BY PRIMARY USE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE

1980

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CORPORATE	0.00	0.00	0.01	0.04	0.02	0.00	0.01
PERSONAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	0.03	0.00	0.03	0.00	0.00	0.02	0.00
AIR TAXI	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OTHER	0.00	0.00	0.00	0.00	0.00	-0.02	-0.02

1981

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	0.00	0.00	0.01	0.00	0.00	-1.05	0.00
CORPORATE	0.00	0.00	-0.01	0.07	0.04	0.00	0.01
PERSONAL	0.00	0.00	0.00	0.00	0.00	-0.24	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	-2.23	0.00	-2.16	0.00	0.00	-2.27	0.00
AIR TAXI	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OTHER	0.00	0.00	0.00	0.00	0.00	-0.04	-0.03

1982

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	-0.10	0.00	-0.06	0.00	0.00	-2.04	0.00
CORPORATE	0.00	0.00	-0.04	0.09	0.06	0.00	0.02
PERSONAL	-0.19	0.00	-0.19	0.00	0.00	-0.59	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	-4.03	0.00	-4.20	0.00	0.00	-4.14	0.00
AIR TAXI	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OTHER	0.00	0.00	0.00	0.00	0.00	-0.06	-0.05

ACTIVE AIRCRAFT BY PRIMARY USE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE

1983

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	-0.37	0.00	-0.26	0.00	0.00	-2.93	0.00
CORPORATE	0.00	0.00	-0.06	0.09	0.09	0.00	0.02
PERSONAL	-0.57	0.00	-0.57	0.00	0.00	-1.00	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	-7.27	0.00	-7.41	0.00	0.00	-5.72	0.00
AIR TAXI	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OTHER	-0.18	0.00	-0.18	0.00	0.00	-0.07	-0.05

1984

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	-0.79	0.00	-0.61	0.00	0.00	-3.62	0.00
CORPORATE	0.00	0.00	-0.08	0.09	0.11	0.00	0.02
PERSONAL	-1.10	0.00	-1.11	0.00	0.00	-1.42	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	-9.40	0.00	-10.25	0.00	0.00	-7.14	0.00
AIR TAXI	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OTHER	-0.56	0.00	-0.56	0.00	0.00	-0.08	-0.07

1985

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	-1.34	0.00	-1.09	0.00	0.00	-4.21	0.00
CORPORATE	0.00	0.00	-0.10	0.08	0.11	0.00	0.02
PERSONAL	-1.74	0.00	-1.75	0.00	0.00	-1.43	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	-11.21	0.00	-13.56	0.00	0.00	-8.20	0.00
AIR TAXI	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OTHER	-1.08	0.00	-1.09	0.00	0.00	-0.07	-0.06

ACTIVE AIRCRAFT BY PRIMARY USE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE

1986

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	-1.96	0.00	-1.64	0.00	0.00	-4.21	0.00
CORPORATE	0.00	0.00	-0.10	0.07	0.12	0.00	0.02
PERSONAL	-2.45	0.00	-2.45	0.00	0.00	-2.18	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	-11.41	0.00	-15.19	0.00	0.00	-8.48	0.00
AIR TAXI	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OTHER	-1.72	0.00	-1.73	0.00	0.00	-0.07	-0.06

1987

	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
BUSINESS	-2.57	0.00	-2.21	0.00	0.00	-4.31	0.00
CORPORATE	0.00	0.00	-0.09	0.06	0.12	0.00	0.02
PERSONAL	-3.09	0.00	-3.09	0.00	0.00	-2.49	0.00
AERIAL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
INSTRUCTIONAL	-11.15	0.00	-17.22	0.00	0.00	-8.77	0.00
AIR TAXI	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OTHER	-2.43	0.00	-2.43	0.00	0.00	-0.07	-0.06

0

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE

DATA PAGE

FUEL CONSUMED DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE

	AV GAS	JET FUEL
1977	0.00	0.00
1978	0.00	0.00
1979	0.00	0.00
1980	-0.39	-1.16
1981	-0.77	-2.06
1982	-1.14	-2.77
1983	-1.58	-3.30
1984	-2.02	-3.69
1985	-2.24	-3.37
1986	-2.61	-3.10
1987	-2.95	-2.48

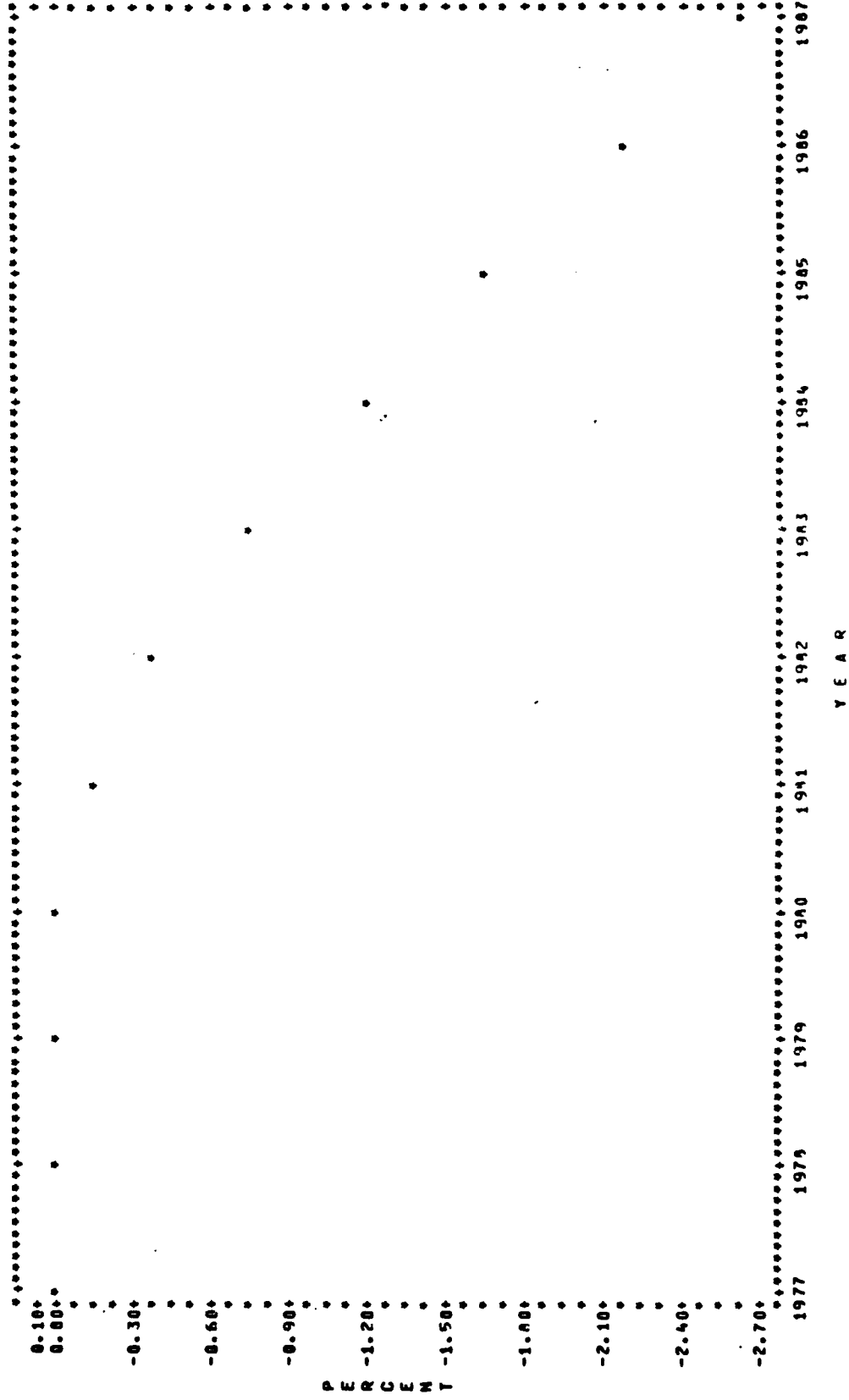
WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
DATA REVENUE

FEDERAL TAX REVENUE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE

1977	0.00
1978	0.00
1979	0.00
1980	59.17
1981	117.47
1982	175.11
1983	232.69
1984	288.84
1985	288.53
1986	286.96
1987	285.19

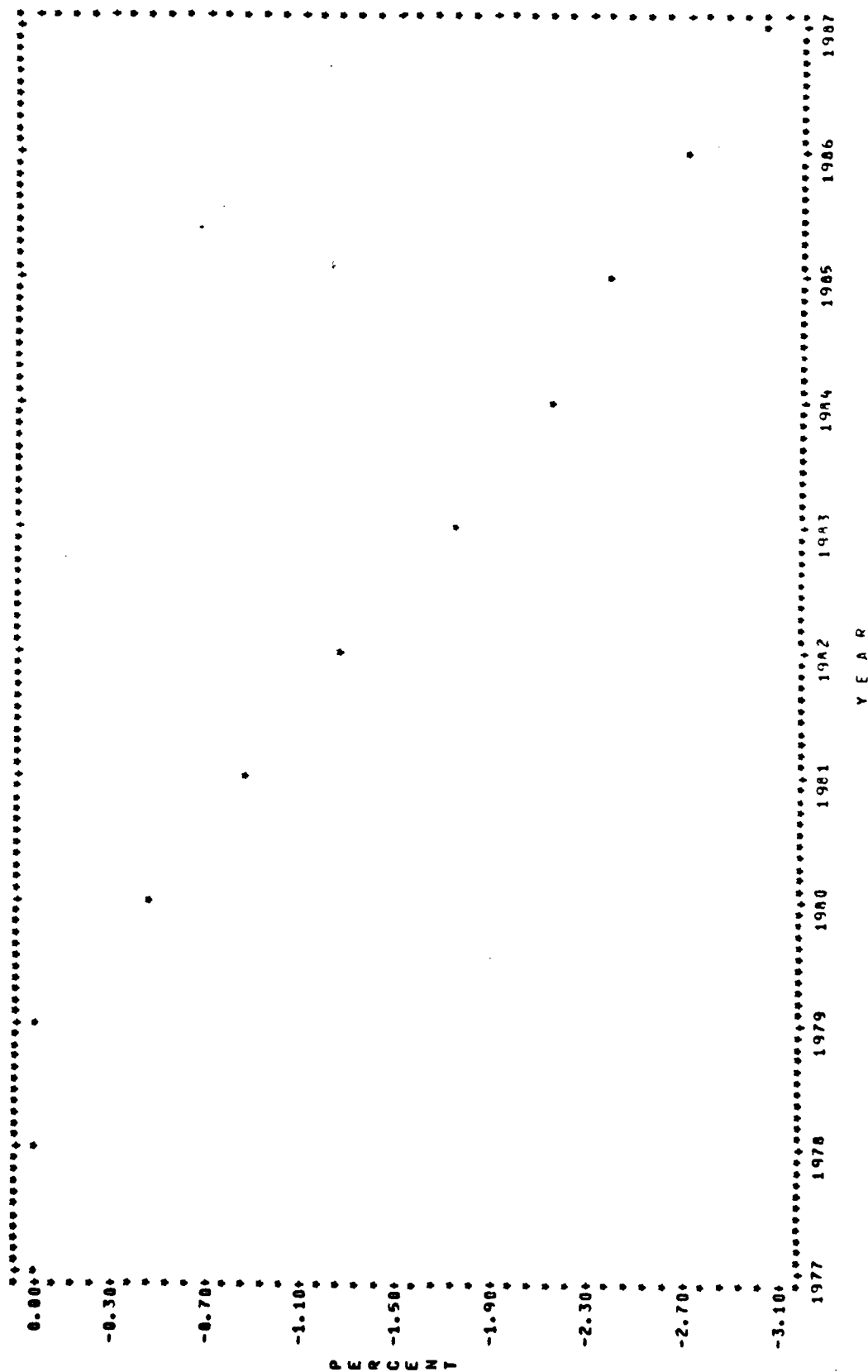
WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
 DATA NONE
 DO YOU WANT TO SEE PLOTS OF THE SENSITIVITY ANALYSIS RESULTS
 DATA YES
 WHAT SENSITIVITY PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
 DATA LIST
 IDENTIFIER DESCRIPTION
 AA ACTIVE AIRCRAFT BY PRIMARY USE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
 AASUM
 ATP TOTAL AIRCRAFT
 CP AIR TRANSPORT PILOTS
 FC COMMERCIAL PILOTS
 FTR FUEL CONSUMED DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
 HF FEDERAL TAX REVENUE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
 HF HOURS FLOWN (THOUSANDS) DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
 HFSUM
 HP TOTAL HOURS FLOWN
 HR HELICOPTER PILOTS
 IP HELICOPTER RATINGS
 OPS INSTRUMENT RATINGS
 OPS OPERATIONS (THOUSANDS) DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
 OPSLH
 PP TOTAL OPERATIONS
 P PRIVATE PILOTS
 P PILOT SURTOTAL
 SP STUDENT PILOTS
 TC TOTAL COST, AS PERCENT DEVIATION FROM BASELINE
 THP TOTAL HELIC RATINGS
 TP TOTAL PILOTS
 WHAT SENSITIVITY PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
 DATA AASUM

TOTAL AIRCRAFT, 1977 TO 1997



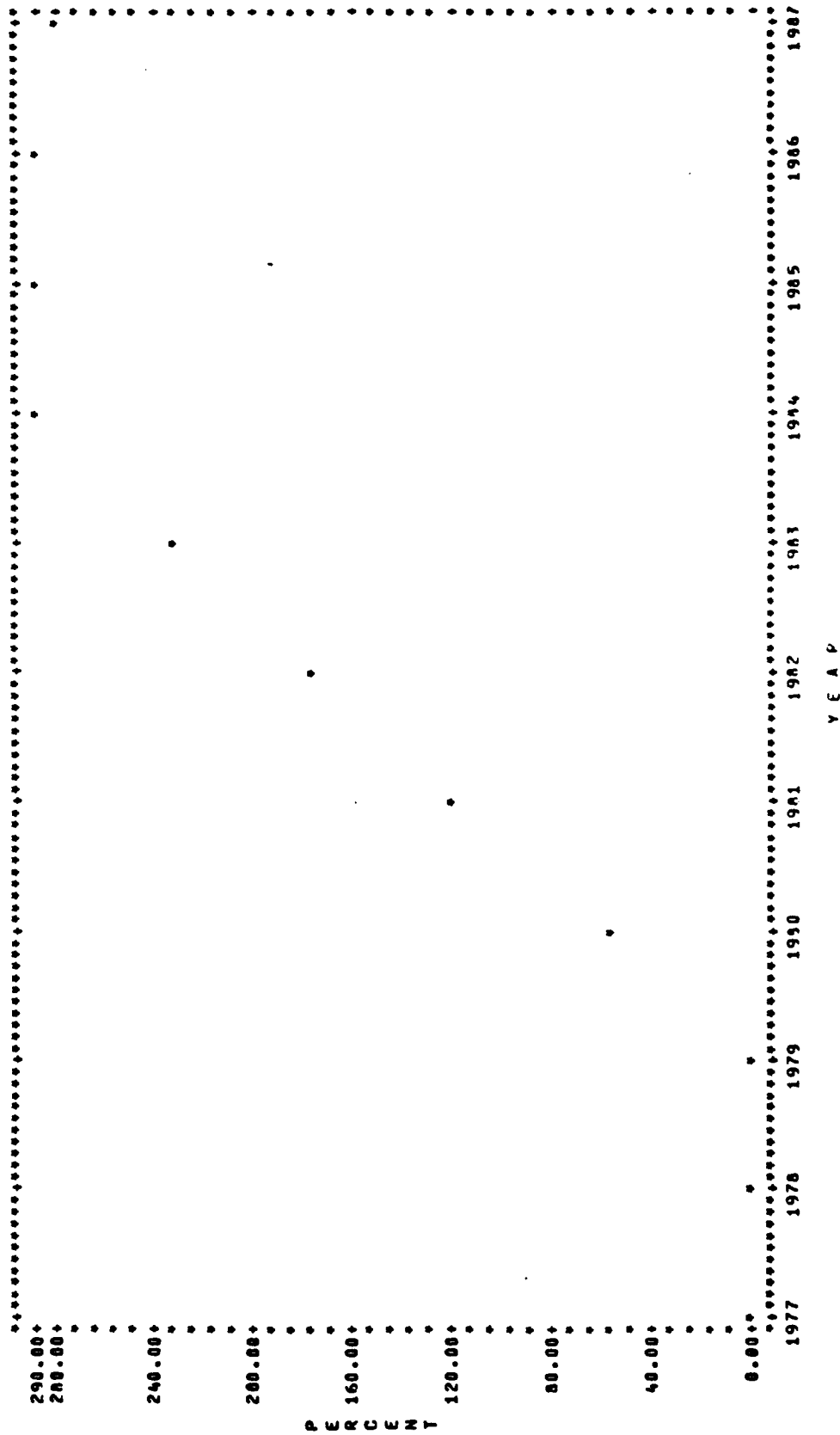
WHAT SENSITIVITY PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
DATA MFSUM

TOTAL HOURS FLOWN, 1977 TO 1987



WHAT SENSITIVITY PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
DATA FIP

FEDERAL TAX REVENUE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE,
1977 TO 1987



WHAT SENSITIVITY PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
DATA NONE

WOULD YOU LIKE TO SAVE THE RESULTS OF THIS SESSION FOR LATER SENSITIVITY ANALYSIS IS

DATA NO

WOULD YOU LIKE TO CONTINUE WITH ANOTHER FORECAST

DATA NO

YOU ARE NOW LEAVING THE GENERAL AVIATION DYNAMICS MODEL.

**APPENDIX C. USER LANGUAGE REFERENCE
MANUAL FOR NUCLEUS**

USER LANGUAGE

REFERENCE MANUAL

for

NUCLEUS, FAA

TABLE OF CONTENTS

	<u>Page</u>
I. USER DOCUMENTATION.	I-1
1.0 Introduction.	I-1
2.0 Overview of the User Language	I-3
2.1 The Structural Elements of the User Language	I-7
2.1.1 Identifier.	I-7
2.1.2 Index	I-8
2.1.3 Dataset	I-9
2.1.4 Model	I-11
2.1.5 Graph	I-12
2.1.6 Table	I-13
2.1.7 Program Module.	I-13
2.1.8 Equation.	I-14
2.1.9 Data File	I-16
2.1.10 Table Management System Array	I-18
2.2 General Syntax Rules	I-19
2.2.1 Character Set	I-19
2.2.2 TIE - Textual Input Element	I-21
2.2.3 Strings and Descriptors	I-23
2.2.4 Line Format	I-24
2.2.5 Continuation.	I-24
2.2.6 Line Length	I-24
2.2.7 Prompting and Echo Printing	I-25
3.0 The Statements of the User Language	I-27
3.1 The SYSTEM Statement	I-30
3.1.1 The USER Parameter.	I-31
3.1.2 The DEBUG Parameter	I-32
3.1.3 The WIDTH Parameter	I-32
3.1.4 The LINES Parameter	I-33
3.1.5 The INPUT Parameter	I-33
3.1.6 The SAVE Parameter.	I-34
3.1.7 The LOAD Parameter.	I-34
3.1.8 The LIST Parameter.	I-35

TABLE OF CONTENTS
(Continued)

	<u>Page</u>
3.1.9 The ZERO Parameter.	I-35
3.1.10 The EPS Parameter	I-36
3.1.11 The SPACE Parameter	I-37
3.1.12 The DEMO Parameter.	I-37
3.1.13 The DB1, DB2, and DB3 Parameters.	I-38
3.1.14 The INITIAL Parameter	I-39
3.1.15 The HEADING Parameter	I-40
3.2 The START Statement	I-41
3.2.1 The HEADING Option	I-41
3.2.2 The AREA Option.	I-41
3.3 The STOP Statement.	I-42
3.4 The OPEN Statement.	I-43
3.5 The SAVE Statement.	I-45
3.5.1 The DATA Option.	I-46
3.5.2 The XEQ Option	I-46
3.6 The LOAD Statement.	I-47
3.6.1 The DATA Option.	I-48
3.6.2 The XEQ Option	I-48
3.7 The TIME Statement.	I-49
3.7.1 The SIZE Option.	I-51
3.8 The ADD Statement	I-52
3.8.1 The SIZE Option.	I-53
3.9 The INDEX Statement	I-54
3.9.1 The LABEL Option	I-54
3.9.2 The TIME Option.	I-55
3.10 The DATA Statement	I-56
3.10.1 The LABEL Option.	I-57
3.10.2 The VALUE Option.	I-57
3.10.3 The TMS Option.	I-58

TABLE OF CONTENTS
(Continued)

	<u>Page</u>
3.10.3.1 The BASE Suboption.	I-59
3.10.3.2 The TRANSPOSE Suboption	I-60
3.10.4 Indirect Data Sets	I-62
3.10.5 Equivalencing Data Sets.	I-63
3.11 The GRAPH Statement	I-63
3.11.1 The X Option	I-64
3.11.2 The Y Option	I-64
3.11.3 Examples of GRAPH Statements	I-65
3.12 The MODEL Statement	I-67
3.12.1 Examples of MODEL Statements	I-68
3.13 The TABLE Statement	I-69
3.13.1 Examples of TABLE Statements	I-71
3.14 The END Statement	I-72
3.15 The READ Statement.	I-73
3.15.1 The READ Index Statement	I-74
3.15.1.1 The STUB Option	I-74
3.15.1.2 The Spanner Option.	I-75
3.15.2 The READ Dataset Statement	I-77
3.15.2.1 Examples of READ Data Statements.	I-78
3.15.3 The READ Graph Statement	I-81
3.15.3.1 Examples of READ Graph Statements	I-83
3.16 The SET Statement	I-84
3.16.1 Examples of the SET Statement.	I-85
3.17 The SHOW Data Statement.	I-88
3.17.1 The ORDER Option	I-89
3.17.2 The TITLE Option	I-90
3.17.3 The TOTAL Option	I-90

TABLE OF CONTENTS
(Continued)

	<u>Page</u>
3.18 Equations.	I-92
3.19 Model Execution Statement.	I-101
3.20 Table Execution Statement.	I-102
3.21 The RATE Statement.	I-103
3.22 The LEVEL Statement.	I-105
3.23 The WRITE Statement.	I-107
3.24 The INPUT Statement.	I-108
3.25 The PUT Statement.	I-109
3.26 The DROP Statement.	I-110
3.27 The IF Statement.	I-111
3.27.1 Boolean Expressions and Variables	I-112
3.28 The DO Statement.	I-114
3.29 SHOW Statement - with XYPLOT, SCATTER, or BAR.	I-115
3.29.1 The POINT Option.	I-116
3.29.2 The TITLE Option.	I-116
3.29.3 The XLABEL Option.	I-117
3.29.4 The X RANGE Option.	I-117
3.29.5 The Y LABEL Option.	I-118
3.29.6 The Y RANGE Option.	I-118
3.29.7 The SUBTITLE Option.	I-119
3.29.8 Examples of SHOW with SCATTER, XYPLOT and BAR.	I-119
3.30 SHOW Statement - with TEXT.	I-123
3.31 TELL Statement - with String.	I-124
3.32 TELL Statement - with DATA.	I-125
3.33 ASK Statement - General.	I-126
3.33.1 ASK Statement - with ELSE DATA = indire indirect.	I-127
3.33.2 ASK Statement - with ELSE INDEX = index.	I-128
4.0 Error Messages.	I-129
5.0 The FORTRAN Interface.	I-135
5.1 When to Use SELNDT.	I-135
5.2 Parameters for SELNDT.	I-136

TABLE OF CONTENTS
(Continued)

	<u>Page</u>
5.2.1 The Parameter IOP.	I-136
5.2.2 The Parameters FILE and TABLE.	I-136
5.2.3 The Parameter THR.	I-137
5.2.4 The Parameter DAT.	I-137
5.2.5 The Parameters DIM and NDIM.	I-137
5.2.6 The Parameter ENT.	I-138
5.2.7 The Parameter BASE	I-138
5.2.8 The Parameter TRANS.	I-138

LIST OF TABLES

	<u>Page</u>
Table I-1. The User Language Statements.	I-5
Table I-2. The Structural Elements of the User Language.	I-6
Table I-3. The Character Set of the System	I-10
Table I-4. Character Groupings	I-10

LIST OF FIGURES

	<u>Page</u>
Figure I-1. Sample Error Messages.	I-133

I. USER DOCUMENTATION

1.0 INTRODUCTION

The NUCLEUS system consists of a user language, an interpreter which interprets and performs the operations specified in the user language, and an interface to FORTRAN. Hereafter, unless otherwise specified, the word "system" will mean the NUCLEUS system.

The user language of the system is a structured programming language according to whose rules and conventions the user writes and organizes his instructions for entry into the system either via keypunched cards or by typing entries on the keyboard of an on-line computer terminal. As a structured programming language it has the following two characteristics: (i) it has no statement labels and thus no GO TO statements, and (ii) major system functions -- such as report generation, data base access, secondary analysis, dialogue control, etc. -- are accessed via single statements. The complete set of user instructions specifying a job to be done by the system is called a source program. The component instructions of a source program are called statements. The user language is especially useful for writing programs which construct, manipulate, and display arrays of numeric data.

The interpreter of the system is a set of ANS* (American National Standard) FORTRAN programs which interpret the statements of a source program and generate interconnected tables of internal instructions which later direct the execution of these statements.

* The single exception to this involves the direct access routines which though they are not in ANS FORTRAN have equivalents in all major computers.

The interface to FORTRAN greatly expands the potential capabilities of the user. The user may use the subroutines of the system in other independently written FORTRAN programs. This is especially important in so far as the data base capabilities of the system are concerned. The call to the FORTRAN subroutine which manipulates the data base, is described in Chapter 5 of this section.

The machine independence of the NUCLEUS system is a very important feature: because the interpreter of the system is machine independent (i.e., written in ANS FORTRAN), any application program written in the user language of the system will also be machine independent, provided the user-supplied FORTRAN subroutines, if any, that are introduced via the FORTRAN interface are machine independent.

An overview of the User Language is given in Chapter 2.0. This chapter includes both a discussion of the structural elements and a description of the general syntax rules of the language.

Chapter 3.0 contains explicit descriptions of each statement of the User Language. For easy reference, the primary word of each statement is placed in capital letters on the upper left-hand corners of all pages describing that statement.

Chapter 4.0 describes the error messages of the User Language, and Chapter 5.0 describes the interface of the system to FORTRAN.

Appendix A contains sample programs - batch and interactive - written in the User Language of the system.

2.0 OVERVIEW OF THE USER LANGUAGE

The statements of a source program are the means by which the user specifies to the system the desired operations of a job or application. A statement may specify one of three types of operations:

- (1) Program control, i.e., it may command the system to perform some overall program control function, such as starting a new program, ending a program, opening a data file, saving a program, loading a previously saved program, or adding an array to a data file.
- (2) Program structure and identifier definition, i.e., it may define the identifier for a structural element of the program and allocate central memory for the associated structure and information for later reference and manipulation within the program.
- (3) Data manipulation, i.e., it may instruct the system to make reference to or manipulate the information associated with previously defined identifiers.

The basic commands, definitions, and instructions of the user language are listed in Table I-1. The first word of any user language statement must be one of the capitalized statement words in this table, except in the following three situations:

- Equation statements
- Model execution statements
- Table execution statements.

In an equation statement the first word of the statement must be the identifier of a previously defined dataset. A model execution statement consists of the identifier of a previously defined model and causes execution of the instructions in the model. A table execution statement consists of the identifier of a previously defined table and causes the printing of the table.

TABLE I-1. THE USER LANGUAGE STATEMENTS

Commands - Program Control

SYSTEM	Sets various system-control parameters
START	Starts a new program
STOP	Stops execution; ends program
OPEN	Opens a data file
SAVE	Saves a program
LOAD	Loads a previously saved program
TIME	Defines the time parameters to control dynamic simulation
ADD	Adds an array to a data file

Definitions - Program Structure

INDEX	Defines an index
DATA	Defines a dataset
GRAPH	Defines a graph
MODEL	Introduces a model
TABLE	Defines an output table
END	Ends the definition of a model, an ASK statement, a conditional DO or IF statement or the definition of multiple indexes, or datasets, or graphs, or tables

Instructions - Data Manipulation

READ	Reads index descriptors, or dataset values, or graph values
SET	Sets index size and ordering of index entries
SHOW	Shows values of a dataset in table or plot form
equation	Computes the values of a dataset or quantity
model	Executes a model
table	Displays a structured tabular report
RATE	Introduces rate equations in a dynamic simulation model
LEVEL	Introduces level equations in a dynamic simulation model
WRITE	Writes values to a table on a direct access file
INPUT	Inputs values from a table on a direct access file
DROP	Drops data set values from working storage
PUT	Puts data set values into working storage
IF	Executes user instructions if some condition is met
DO	Executes user instructions while or until some condition is met
TELL	Tells the user something, i.e., issues a message
ASK	Asks the user a question; based on the response, the system executes one and only one of many possible sets of alternative instructions
ELSE	Defines an execution branch of instructions within an IF or an ASK statement

The basic structural ingredients of elements of the user language and the system are listed in Table I-2. Every application or problem is conceptualized and formulated in terms of these elements before being coded into a program of statements.

TABLE I-2. THE STRUCTURAL ELEMENTS OF
THE USER LANGUAGE

Identifier	A name for an index, a dataset, a model, a graph, a table, or a program module.
Index	A classification (or subscripting) scheme for the values of structured (nonscalar) datasets. Indexes are subscripts of datasets.
Dataset	An n-dimensional storage array containing numeric information, where $n=0,1,\dots,10$. Datasets are the structural components of equations, i.e., the variables of a problem or application.
Model	An ordered sequence of instructions or equations that may be compiled, i.e., defined, as a unit for later execution as a unit.
Graph	An ordered set of pairs of values defining the x- and y-values of the points of a function, $y=f(x)$.
Table	An ordered sequence of datasets defined as a unit for later execution as a titled tabular display of the values of these datasets.
Program Module	An ordered set of user language statements that may be saved for later loading and execution.
Equation	An instruction containing the = sign and setting the value(s) of the dataset at the left-hand side of the = sign equal to the value(s) of the right-hand side.
Data File	A file of information stored in mass storage outside the working space (allocated core) of a program. Data files may contain either program modules or arrays of numeric data.
TMS Array	A n-dimensional storage array containing numeric information and located on a TMS data file, $n=0,1,\dots,10$. The acronym TMS stands for Table Management System and reflects the manner in which the data on the file is brought into working storage.

2.1 The Structural Elements of the User Language

The structural elements of the user language are listed in Table I-2. The semantics, i.e., the meanings, of these elements are further described in this chapter. The syntax of the language, i.e., the manner in which these elements are specified in the statements of the user language, is described in Chapter 3.0.

2.1.1 Identifier

An identifier is a user-supplied name or abbreviation for some structural component of a program - such as an index, or a dataset, or a model, or a graph, or a table, or a program module, or a TMS array. The identifier of a component is used to define and to refer to that component later in the program. All identifiers except those for TMS arrays must contain no more than five alphabetic and/or numeric characters and their first character must be alphabetic. They must contain no blanks or any other special characters. To avoid ambiguities, identifiers must be unique within a given program. TMS arrays are identified via an ordered pair of integer values. The first integer in this pair defines the particular data file within the data file set being used which contains the array. Within this version of the system a given data file set may contain a maximum of three files; thus, the data file identifier must be a number between one and three. The second integer in the TMS array defines the sequence number of the desired array from among the arrays contained on the file; thus, it is an integer value between 1 and n. The value of n is unique for each file and is established by the user when that file is first defined.

The identifier of a structural element does not represent a single machine or file address; rather, it refers to the structured complex of related information contained in the element. When the identifier appears in a statement, the system automatically references the particular portion of the element that is relevant to the context of use of that statement.

2.1.2 Index

The index is one of the two most basic concepts in the system; the other being the dataset concept. The index is the organizational classification scheme in terms of which the internal structure of the datasets in a program is defined. Indexes serve as generalized subscripts of datasets. For example, age and sex may be two indexes classifying the age and sex structure of a population dataset. Each index has two mandatory characteristics. These are:

- Its identifier
- Its size.

The index identifier is used to reference the index later in the program, where it is used primarily as a subscript of the dataset(s) that are classified by it. The identifier of the sex index, for example, may be SEX.

The index size is a non-negative integer which defines the maximum number of entries (or categories) that may occur within the classification scheme of the index. For example, the size of the SEX index is 2 for the two sex classifications, male and female.

The index descriptors are the user-supplied descriptors for the index entries. For example, the two entries of the SEX index may be defined as MALE and FEMALE or as SEX1 and SEX2 or in some other way, as desired by the user. The descriptors of an index are used as stubs (row descriptors) and/or spanners (column headings) of tabular displays of the datasets classified by the index. The descriptors of an index are not mandatory for the definition of the index. However, they are indispensable in defining the rows, columns, or subtitles of the tabular displays of the datasets of the program.

The entries of an index are sequentially ordered from 1 to n, where n is the size of the index, according to the order in which they are read into the program. This ordering may be modified by the SET index statement to allow selective data manipulation or data retrieval and display operations.

2.1.3 Dataset

The concept of the dataset is central in the system. It is in the datasets of the program for an application that the numeric information is stored, and it is through the use of equations and other instructions which manipulate these datasets that the logical structure of the application is defined.

A dataset must have a unique identifier so that it may be referenced in subsequent instructions - particularly equations - within the program.

A dataset may contain only a single numeric value - in which case it is referred to as a scalar - or it may have multi-dimensional structure. A multi-dimensional dataset is an array of storage cells containing numeric information which is classified (i.e., subscripted) by one or more indexes.

The user may use as many (≤ 10) different indexes as are needed to classify the values of a given dataset. A one-dimensional array - sometimes referred to as a vector - is a set of values classified by a single index. For example, the total population values by age for a given state and a given year may be stored in a single one-dimensional dataset that is classified by an age index. The population values by age and sex for a given state and year may be stored in a single two-dimensional dataset whose two dimensions are the age and sex indexes. The population values by age, sex, and state for the entire U.S. in a given year may be stored in a single three-dimensional dataset. The third dimension in this dataset is the state index whose size is 50. A four-dimensional population dataset may have year as the fourth index, and so forth. The almost unlimited dimensionality of datasets is an important property of the system as compared to other systems which allow only three dimensions for subscripted arrays. The sizes of the indexes of multi-dimensional datasets are not restricted by the user language; they are restricted only by the size of central computer memory available.

A dataset has the following characteristics:

- A unique identifier (mandatory)
- Previously defined index(es) as subscripts. Some datasets (scalars) have no indexes. Up to 10 unique indexes may classify a dataset. The order of dataset indexes is important in READ and SHOW statements as well as in equations involving the dataset.
- An optional label, which appears as the title of the dataset in later SHOW operations.
- An array of values, equal in number to the product of the sizes of the indexes classifying the dataset. Initially, these values are set to zero.

In addition to the above, there is one other very important characteristic of the dataset array of values. This is the location type for those values. Datasets are said to be fixed, scratch, or TMS depending upon the manner in which the dataset values are assigned a location. Fixed datasets are assigned to fixed location within the program module at the time the dataset is defined. Any value associated with that dataset will remain constant unless explicitly changed by the user even if the program module is saved and loaded. Scratch data sets are assigned temporary locations when they are assigned values. They maintain these locations and values only during the current execution of the program module. Their values are not maintained when their program module is saved; thus, scratch datasets are used for scratch data storage. TMS datasets in essence have two locations - one within a TMS array on some data file and a temporary one within the program module. TMS datasets have all of the advantages of fixed datasets in that they maintain their values at all times and they have the advantages of scratch datasets in that they do not require a fixed amount of space within the program module. It is through the use of TMS datasets that large, data intensive models can be written.

An additional benefit of the use of TMS datasets is that the system automatically performs all space allocations required to bring the TMS array values in and out of the program module. The user of the system need never concern himself with the logistics of the operations in the user language. This fact makes simple the writing of programs within this user language.

2.1.4 Model

The model is a group of instructions that are compiled as a unit under a single, unique identifier for later reference and execution as a unit. Model definition or compilation is initiated with the MODEL statement and is terminated by the END statement. Model execution is initiated by entering the model identifier. The model statements are executed sequentially in the same order as they are defined, i.e., from top to bottom.

In a general sense, a model is a symbolic or mathematical representation of an actual system. The model instructions are usually the equations defining the logical relationships between the variables (datasets) of the system. These relationships may or may not express the interaction of variables through time.

There are two types of models in the system:

- Static models
- Dynamic models.

The equations in static models do not contain any time dependence. The equations in a dynamic model, on the other hand, describe the evolution through time of the system variables, and thus of the actual system being modeled.

Every dynamic model has four implicit parameters associated with time:

- The time parameter itself
- DT, the time increment parameter
- Beginning time
- Ending time.

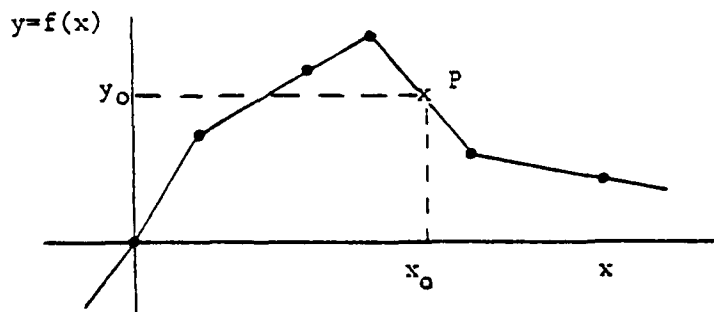
The time parameter is a quasi-continuous variable, designated by the keyword TIME, which is incremented from beginning time to ending time in steps of DT, the time increment parameter. DT is also a system keyword.

The equations in a dynamic model are divided into three sections: the initial section, the rate section, and the level section. The initial equations define the values of the model variables (or levels) at beginning time. The rate equations define the values of those variables defining the rates of growth of the system levels. The level equations define the values of the system levels at an arbitrary time point in the interval between beginning time and ending time.

The specifics of dynamic models are further discussed in a later part of this handbook.

2.1.5 Graph

The graph is defined as a finite set of ordered pairs of numbers. These number pairs may be thought of as the points on a plane representing some functional relationship, $y=f(x)$, such as the one shown below:



This graph is defined as the set of the six x-values and the six y-values defining the x- and y-coordinates of the six points denoted by dots in the above schematic.

A graph may be referenced in the right-hand side of equations, where it takes various values depending on the value of its argument at the time of equation execution. The value (y-value) of a graph for an arbitrary argument (x-value) is automatically computed by the system by using linear interpolation or extrapolation between the points defining the graph. For example, the value at x_0 of the graph shown above is equal to y_0 , the y-coordinate corresponding to the point P on the graph.

2.1.6 Table

The output of a program may be organized into tables, or the user may display each program dataset directly by using the SHOW statement. A table is first defined by means of a unique identifier, a title, a size specification to define the stub (row descriptors) and column widths, and a list of the datasets whose values are to be displayed in the table. A defined table is executed, i.e., displayed or printed out, by simply entering its identifier. Table execution results in a centered, titled, and otherwise properly structured tabular display of the values of the datasets contained in the table. The tabular displays of the system are designed to fit a standard 8-1/2 x 11 inch page. Large tables are appropriately segmented into pages. Table paging is automatic. To satisfy various terminal screen or printer requirements, the length and width of tables may be adjusted by the user.

2.1.7 Program Module

A program module is an ordered set of user language statements that may be saved for later loading and execution.

A program module is bounded by a START or a LOAD statement at its beginning and a SAVE or a STOP statement at its end.

2.1.8 Equation

An equation is any statement that begins with a previously defined dataset identifier, is followed by an equal (=) sign, and contains on the right-hand side (RHS) an arithmetic expression.

Arithmetic expressions are ordered sequences of numerical constants, previously defined identifiers, mathematical operators, and parentheses.

Two types of mathematical operators are available in the system:

- Arithmetic operators
- Functional operators.

The arithmetic operators of the system are:

<u>Operator</u>	<u>Operation</u>
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation.

The functional operators of the system are:

<u>Operator</u>	<u>Function</u>
SUM	Summation over subscripted datasets or arithmetic expressions with subscripted datasets
MAX	Finding the maximum value in a vector dataset
MIN	Finding the minimum value in a vector dataset
ABS	The absolute value of a dataset
LOG	The base 10 logarithm of a dataset
LN	The natural logarithm of a dataset
EXP	The exponential of a dataset.

At execution, the values of the datasets on the RHS of an equation must be well defined, i.e., the datasets contained in the expression must have definite values. Upon execution, the values contained in the storage cells of the dataset on the left-hand side (LHS) are replaced by the values of the arithmetic expression on the RHS. A single identifier may be associated with different numeric values at different points in the program or at different times during execution.

Left-to-Right Hierarchy. In the absence of parentheses, arithmetic expressions are compiled and evaluated from left to right with each arithmetic operator using the preceding result as an operand. All arithmetic operators are of equivalent rank in the processing of equations by the system.

Note. This hierarchy is different from the standard algebraic hierarchy used in FORTRAN.

To alter the implied left-to-right hierarchy in which arithmetic operations are performed, the user may employ parentheses. Parentheses allow the user to specify his own order for the interpretation and execution of arithmetic expressions. An arithmetic expression (or a portion of one) beginning with a left parenthesis and ending with a right parenthesis is called a parenthetical group. A parenthetical group is said to be nested if it is contained within another parenthetical group.

The operations of the most deeply nested parenthetical group of an arithmetic expression are executed first. Operations within a parenthetical group are executed from left to right. Parenthetical groups of equivalent nesting are processed from left to right.

In general, the rules for constructing well formed equations are:

- (1) No two operators may directly follow one another.
- (2) No two constants or variables may directly follow one another.
- (3) The right-hand side of an equation must begin and end with a variable, constant, or parenthetical group.
- (4) A parenthetical group may appear only on the right-hand side of an equation.
- (5) Right and left parentheses must always be balanced, i.e., be equal in number.
- (6) A left parenthesis, (, must be preceded by another (, an operator, or an equal sign, and must be followed by another (, an identifier, or a constant.
- (7) A right parenthesis,), must be preceded by another), an identifier, or a constant, and be followed by another), or an operator.
- (8) An equation may be continued to the succeeding line as long as an operator or a comma is the last character on the line(s) being continued.
- (9) Equations are executed from left to right.

2.1.9 Data File

The data file makes it possible for the user to extend the storage available to him from the actual central memory of the computer to off-line direct storage devices such as the disk. There are two very important reasons why this extension is needed for large-scale modeling applications.

In the first place, it is required that models written by one group of researchers be accessed and used by others at some later point in time. Off-line storage is needed to perform this saving operation. In the second place, large-scale models are usually data intensive, often manipulating millions of numeric values during a single execution cycle. It would be impossible to store all of these values within the central memory of the machine. Off-line storage is needed to extend the numeric value storage area required by the execution of the models.

As is implied in the above, there are two basically different types of data files used by the system. These are referred to as save files and table management system (or TMS) files. Save files are used for the storage of program modules. They are needed primarily so that previously written program modules can be saved and loaded for later use. Table management system files are used for the storage of TMS arrays. They are needed primarily to contain the values of TMS datasets and thus to extend the numeric value storage available to the executing models. TMS files also, of course, allow the saving of these numeric values for later use.

It is through the combined use of save files and table management system files that this system achieves its powerful data base management capabilities. These data base management capabilities make it possible to easily structure and implement large complex models and their associated information systems in such a manner that they are easy to use.

Associated with each data file there is a physical unit number. This physical unit number is used to identify the particular file to the operating system. Before a given file can be used, it must be declared to the operating system via its physical unit number, using the directives of the operating system. Prior to using this system the user should obtain information from his computer staff as to the required form of these directives.

In addition to the physical unit number there is also a size associated with each data file. Prior to creating any new data file, the user must first calculate the total amount of space which he will require. Based on this calculation he can then either initially declare the file so that it has the appropriate size when he defines it to the operating system or he can choose the appropriate physical unit number for a physical file which has the appropriate size. The particular manner in which either of these is accomplished depends upon the operating system. The user should consult with his computer staff before beginning the creation of a new data file.

2.1.10 Table Management System Array

The table management system (TMS) array is the basic unit on a TMS data file. It may be thought of as an n-dimensional ($0 \leq n \leq 10$) array of numeric values. In this sense it is precisely like a dataset. The primary purpose of the TMS array is to contain values for TMS datasets.

A TMS array has the following characteristics:

- A unique identifier consisting of a data base number and an array number as described in Subpart 2.1.1.
- A size specification which is a sequence of from zero to ten integer values which define the number of dimensions in the array and the number of entries in each dimension. An array with zero dimensions consists of a single value. The total number of values associated with a multi-dimensional array equals the product of the number of entries in each dimension.

2.2 General Syntax Rules

The User Language statements are listed in Table I-1. This part describes the syntax of the User Language, i.e., the manner in which the statement words in this table are combined with other keywords, user-supplied identifiers, strings, and special characters to form instructions that are acceptable for processing by the system.

Keywords are reserved words in the User Language that are used to specify more completely the statement beginning with one of the statement words in Table I-1. For example, the word TITLE is a keyword used in the definition of a TABLE statement and it allows the user to specify a title for the table being defined. Similarly, the word LABEL is a keyword used in dataset definitions to allow the user to specify dataset labels or descriptors.

2.2.1 Character Set

The system interpreter recognizes the 47 characters listed in Table I-3. Each character is associated with a character sequence number. Some familiarity with these sequence numbers is helpful for understanding how the system interprets input.

The characters in Table I-3 may be divided into the six groups defined in Table I-4.

Alphabetic characters are used to define and reference all identifiers in a program. Each identifier must have an alphabetic character as its first character.

Numeric characters are used to formulate all numeric constants of the program - values of datasets and graphs, index sizes, index settings, dataset subscript values, format specifications, etc. They may also be used as secondary characters in the formulation of identifiers.

The sign characters, + and -, are used to define the arithmetic operations of addition and subtraction, respectively. They are also used to define the sign, positive or negative, of numeric constants or arithmetic expressions.

TABLE I-3. THE CHARACTER SET OF THE SYSTEM

Seq	Char	Seq	Char	Seq	Char	Seq	Char
1	A	13	M	25	Y	37	+
2	B	14	N	26	Z	38	-
3	C	15	O	27	0	39	*
4	D	16	P	28	1	40	/
5	E	17	Q	29	2	41	(
6	F	18	R	30	3	42	)
7	G	19	S	31	4	43	\$
8	H	20	T	32	5	44	=
9	I	21	U	33	6	45	blank
10	J	22	V	34	7	46	,
11	K	23	W	35	8	47	.
12	L	24	X	36	9		

TABLE I-4. CHARACTER GROUPINGS

Group	Type	Characters	Sequence Numbers
1	Alphabetic	A to Z	1 to 26
2	Numeric	0 to 9	27 to 36
3	Sign	+ -	37 to 38
4	Special	* / () \$ =	39 to 44
5	Blank	blank ,	45 to 46
6	Decimal	.	47

The special characters * and / are the multiplication and division operators. In addition, for batch use of the system, / may be used to skip to the next page in the program listing while an input line having * in the first column indicates a line of comment to be printed but not processed by the interpreter. The character sequence ** is the exponentiation operator. The characters (and) are used to enclose program descriptors, dataset subscripts, graph arguments, parenthetical groups in arithmetic operations, etc. The = character is used in equation statements.

All special characters may be used freely in defining descriptor strings. Special characters may not be used in identifiers.

The two blank characters, blank and comma, serve as TIE-breakers, i.e., they separate the Textual Input Elements (TIEs) in user language statements. In this context, they are very important ingredients of the interpreter of the system. The comma is also used as a continuation character to continue a line of input to the next line.

The decimal character is used to specify real constants.

2.2.2 TIE - Textual Input Element

All statements are recognized and interpreted by the system as ordered sequences of Textual Input Elements (TIEs) which are separated from each other by TIE-breakers.

A TIE is any sequence of characters which are in some sense logically related. Thus, the sequence of characters FRED, preceded and followed by blanks, may be thought of as a single TIE which names something. Similarly, the sequence of characters 3.14159, preceded and followed by blanks, may be thought of as a single TIE whose value approximates the value of π .

Consider, however, the sequence SHOW POPT. Is this sequence one single TIE? Or, is it two TIEs separated by a blank? The second interpretation appears to be more correct, because the sequence SHOW is a request to show something and POPT is the name of whatever is to be shown. This example, demonstrates the need for the notion of a TIE-breaker. A TIE-breaker is any aspect of the input stream which causes a TIE to end. Thus, the TIEs FRED, 3.14159, SHOW, and POPT are broken by the blanks or commas following them.

The notions of TIE and TIE-breaker cannot be considered independently of each other. Both are needed to understand the formulation of system statements. Any system statement is interpreted from left to right as a string of recognizable TIES - words, numbers, etc. - comprised of alphabetic, numeric, and special characters, separated from each other by TIE-breakers, and ordered according to the rules of syntax spelled out in this handbook. It is the recognition of distinct TIEs which is crucial to the understanding of sequences of characters, not just the recognition of the characters themselves.

TIEs may be classified into four types depending on the types of individual characters that make them up. The four types of TIEs may be defined in terms of the six groups of characters defined in Table I-4.

A Type 1 TIE is either a single special character, or a sign character which is not followed by a numeric character, or a decimal character which is neither followed nor preceded by a numeric character. A Type 1 TIE consists always of a single character.

A Type 2 TIE is a sequence of only alphabetic and numeric characters, i.e., it is used as a keyword or as a name. The first character of a Type 2 TIE must be alphabetic. The first nonblank TIE of any statement is a Type 2 TIE. Also, all index, dataset, and model identifiers must be unique Type 2 TIEs.

A Type 3 TIE is an integer, i.e., it is a sequence of numeric characters which may be preceded by a single sign character or by a sign character followed by blank characters. A sequence of numeric characters preceded by alphabetic characters is, of course, a Type 2 TIE, and not Type 3.

A Type 4 TIE is a real number. It consists of a sequence of numeric characters containing one and only one decimal point. A Type 4 TIE sequence may be preceded by a single sign character or by a single sign character followed by blank characters.

A statement is converted into a sequence of values corresponding to the various types of TIEs comprising it, except for blanks and commas. Blanks and commas have no internal values. Blank characters are either ignored (except in descriptors) in an input sequence or they serve as TIE-breakers.

By definition, a TIE of a given type is not ended unless it is followed by a blank character or by a TIE of a different type. A TIE-breaker, therefore, is either a blank character or the beginning character of a TIE of a different type.

The TIE and TIE-breaker notions are not only central to the interpreter of the system; they are also linked to the error processing operations of the interpreter. Usually, the error diagnostics issued by the interpreter are expressed in terms of the TIEs in the input statement that are in error.

2.2.3 Strings and Descriptors

In addition to TIEs, the other basic syntactic element in terms of which the system recognizes and interprets statements is the string. Every statement in a program is viewed as an ordered sequence of TIEs and strings. A string is an arbitrary sequence of characters. To qualify as a string, a sequence of characters must be enclosed in parentheses.

The interpretation of a sequence of characters enclosed in parentheses as a TIE or a string is context sensitive, i.e., it depends on the relative location within the type of statement in which it occurs.

All structural elements of the language have mandatory identifiers, which are mnemonics of up to 5 alphanumeric characters. These mnemonics are abbreviated names of the structural elements. For additional program documentation these identifiers may be further described by explicit descriptors. For example, the identifier POPM for a dataset of population values may be described by the label or descriptor MALE POPULATION.

All labels or descriptors of the language are character strings and are specified as part of the definitions of the structural elements which they define.

2.2.4 Line Format

The User Language is generally a free-form language. Language statements may be begun anywhere on the input line or card, and as many blanks or commas as desired may be inserted between the various TIEs to improve statement readability.

2.2.5 Continuation

If a given statement will not fit on a single card or input line, then continuation of that statement may be indicated in one of three ways, depending upon the context.

- (1) If the system is currently reading a character string, then continuation is automatic. The first character of the following card or input line is concatenated directly behind the last character of the preceding card or input line.
- (2) If the system is currently reading an equation, then continuation to the next cards or input lines will be assumed if and only if the last nonblank character of the current card or line is a comma or an arithmetic operator.
- (3) If the system is currently reading neither a string nor an equation, then the continuation to the next card or input line will be assumed if and only if the last nonblank character on the current card or input line is a comma.

2.2.6 Line Length

The length assumed by the system for each input line depends entirely on the source of those lines. If input is coming from any source other than the standard input file connected to the user terminal in

interactive mode, then each input line is assumed to contain exactly 80 characters. If a physical input line contains more than 80 characters, it will be truncated. If it contains less than 80 characters, it will be padded out to 80 with blanks.

If the input is coming from the user terminal, the situation is more complex. Basically, each input line is as long as the number of characters typed by the user after the system prompt and before entering carriage return. Thus, the line length assumed by the system is independent of the actual line spacing being generated at the particular terminal being used. The user must be careful to make any continuation provisions before he enters carriage return which is not necessarily before he exhausts a given line at the terminal. The maximum length of any line entered at the terminal is 80 characters.

2.2.7 Prompting and Echo Printing

The decision as to whether to prompt the user for the next input line or to echo print an input line after it has been read depends entirely on the source of this line. If input is coming from any source other than the standard input file connected to the user terminal in interactive mode, the system will echo print that input. If the input is coming from the user terminal, then the system will prompt for that input.

For the interactive or on-line user, the system issues a prompt before each instruction. The interactive prompt for a language statement has the following format:

READY/

The interactive prompt for input data following a READ statement is:

DATA/

For the batch user, the system echo prints all the instructions in the program following the very first instruction

READY/SYSTEM USER=OFF

which initiates batch mode. This instruction is not echo printed.

Each batch instruction is preceded by two decimal integers specifying the starting address in core where this instruction is stored and the last available address in core that can be used for the storage of values associated with scratch and TMS data sets. All echo-printed input data cards following READ instructions are preceded by the word DATA.

3.0 THE STATEMENTS OF THE USER LANGUAGE

The statements of the User Language are listed in Table I-1. Most of these statements are used in the formulation of the sample programs shown in Chapter 5.0. In this chapter, the User Language statements are described in terms of their general syntax in the same order as shown in Table I-1.

Each statement description contains a summary description of the basic function of the statement, the general syntax for the statement, and descriptions of all options, secondary keywords, and user-supplied identifiers, descriptors, and parameters used in the general statement syntax.

The general statement syntax consists of the following:

- The statement word - always the first word of the statement
- Secondary keywords or options
- Special characters
- User-supplied identifiers, descriptors, and parameters.

The statement words of the language are listed in Table I-1. All language statements begin either with one of the statement words in Table I-1 or with a previously defined identifier in the case of equations, model execution statements, or table execution statements. In this sense statement words are primary keywords of the language.

The secondary keywords of a statement are reserved words in the language that allow the user to specify various options or parameters associated with the general function of the statement.

The meaning of secondary keywords is context sensitive and always relative to the statement to which they belong. For example, the word TITLE is a keyword used with the TABLE statement to allow the user to specify a title for the table being defined by the TABLE statement.

Some secondary keywords are mandatory for the complete or correct specification of a statement and some are optional.

Special characters - such as blanks, commas, periods, parentheses, asterisks, etc. - are used in the formulation of a statement to separate or link various components of the statement or to define specific statement options.

The meaning of a special character is also context sensitive and it is related to the statement to which it belongs. For example, the use of the asterisk, *, in an equation means "multiply", while its use in a SET index statement means "restore index setting to its original size and ordering".

Even within a statement, a special character may have different meanings. In the following example defining the dataset AGR

```
DATA AGR, LABEL (ANNUAL GROWTH RATE ($/YEAR))
```

the first set of parentheses enclosing the label is mandatory and part of the syntax of the DATA statement, while the second set of parentheses enclosing \$/YEAR is optional and part of the user-supplied descriptor. This descriptor could have been stated without these parentheses

```
DATA AGE, LABEL (ANNUAL GROWTH RATE IN $/YEAR)
```

Some special characters are mandatory and some are optional for the complete or correct specification of a statement.

In this manual, the syntax of a statement or an option is shown enclosed in a box.

The statement word and all secondary keywords defining various statement options are capitalized in the syntax of each statement.

All user-supplied identifiers, descriptors, and parameters are in lower case.

Identifiers are always alphanumeric and may contain up to 5 characters, the first of which must be alphabetic. They must contain no special characters. The notation for an identifier in a syntax description is "id".

Descriptors are character strings that may contain up to 125 characters. All characters in the character set of the system are allowed. Descriptors are always enclosed in parentheses. The notation for a descriptor in a syntax description is "string".

Parameters are usually numeric values that define sizes for various structural elements or various format specifications. Parameters are usually abbreviated by one or two-letter symbols.

The notation "list" in syntax descriptions means a list of items separated by blanks or commas.

The statements of the User Language, as listed in Table I-1, are divided into three groups:

- Program control statements
- Program structure definition statements
- Data manipulation statements

These language statements are presented in the same order as they are listed in Table I-1.

3.1 The SYSTEM Statement

This is the basic statement establishing the control parameters for the user-system interface. The user-system interface is defined in terms of a set of parameters that specify the following:

- Batch or on-line use of the system
- Debugging or production use of the system
- Width of output page
- Length of output page
- Unit numbers for off-line files on which to save program modules, from which to load previously saved program modules, from which to read data inputs, or which constitute the files within the data base set
- Various options for report generation
- Various TMS data file options.

Syntax

SYSTEM parameter, setting

parameter is any one of the following keywords: USER, DEBUG, WIDTH, LINES, INPUT, SAVE, LOAD, LIST, ZERO, EPS, SPACE, DEMO, DB1, DB2, DB3, INITIAL, or HEADING.

setting is the particular parameter setting selected by the user.

Comments

The parameter settings are local, i.e., they remain in effect only until they are reset in a subsequent SYSTEM parameter statement.

SYSTEM

I-31

The user may set multiple parameters with a single SYSTEM statement.

The SYSTEM statement may be issued within or outside of a model.

3.1.1 The USER Parameter

This parameter allows the user to modify the mode of interaction with the system from its default on-line setting to the batch setting, or vice-versa.

Syntax

A. For on-line use,

```
SYSTEM USER=ON
```

B. For batch use,

```
SYSTEM USER=OFF
```

The ON setting is the default.

Comments

Since the default mode of interaction with the system is ON, the first statement of any batch program must be:

```
SYSTEM USER=OFF
```

This statement must precede all program statements, including the START statement.

The keyword USER itself may be optionally omitted.

3.1.2 The DEBUG Parameter

This parameter allows the user to find the syntax errors in his program module in a single pass of the system. This feature simplifies the debugging of complex batch programs. This parameter is relevant only when the USER parameter is off. If the USER parameter is off and if the system is not in DEBUG mode, then the system will end execution after a syntax error is encountered. If the system is in DEBUG mode, it will continue processing even if errors are encountered.

Syntax

- A. To place the system into DEBUG mode,

SYSTEM DEBUG=ON

- B. To take the system out of DEBUG mode,

SYSTEM DEBUG=OFF

The OFF setting is the default.

3.1.3 The WIDTH Parameter

This parameter allows the user to modify the default output from 72 characters per line to the desired width. Given the variety of printer widths and terminal screen sizes, the user must be able to control this parameter, otherwise his output may be malformed.

Syntax

SYSTEM WIDTH=n

- n is an integer ($40 \leq n \leq 135$) defining the desired printer width (or width of terminal screen) in characters.
Default n=72.

3.1.4 The LINES Parameter

This parameter allows the user to modify the default output page length from 55 lines per page to the desired length. Given the variety of terminal screen sizes or printer page lengths, the user must be able to control this parameter, otherwise his output may be malformed.

Syntax

SYSTEM LINES=n

n is a positive integer defining the desired length in lines of each output page.

Default n=55.

3.1.5 The INPUT Parameter

This parameter allows the user to specify the unit number assigned to an off-line data file from which subsequent READ operations will read input data. This parameter allows the user to read data inputs from any previously constructed and appropriately "attached" off-line data file, other than the default input file of the local computer facility.

Syntax

SYSTEM INPUT=n

n is an integer ($1 \leq n \leq 99$) denoting the unit number of the off-line data file containing coded input for subsequent READ operations.

Comments

The "attach" procedures by which the unit number n is assigned to a data file are outside the scope of this handbook. They are described in the local user manuals of the computer facility where the system resides.

3.1.6 The SAVE Parameter

This parameter allows the user to modify the default physical unit number, 1, assigned to the local file where program modules may be saved for later loading. This local file is referred to as the "save" file.

Syntax

SYSTEM SAVE=n

n is an integer ($1 \leq n \leq 99$) denoting the physical unit number of the "save" file.

Default n=1.

Comments

The "save" file cannot be "cataloged", i.e., saved permanently, unless previously "requested". The "catalog" and "request" procedures are outside the scope of this handbook.

3.1.7 The LOAD Parameter

This parameter allows the user to modify the default physical unit number, 2, assigned to the local file from where previously saved modules may be loaded. This local file is referred to as the "load" file.

Syntax

SYSTEM LOAD=n

n is an integer ($1 \leq n \leq 99$) denoting the unit number of the "load" file.

Default n=2.

3.1.8 The LIST Parameter

This parameter controls the echo printing of inputs and is relevant only if the USER parameter is OFF, i.e., only in batch mode.

Syntax

- A. To echo print all input consecutively,

`SYSTEM LIST=ON`

This is the default setting.

- B. To echo print next input line on top of next page,

`SYSTEM LIST=EJECT`

Comments

When list is set to EJECT, the system changes it to ON once the next input line has been printed at the top of the page.

3.1.9 The ZERO Parameter

This parameter allows the user to control the display of "zero" entries in output tables.

Syntax

- A. To display "zero" entries using the character 0 (or 0.00...),

```
SYSTEM ZERO=0
```

- B. To display "zero" entries using special characters, .

```
SYSTEM ZERO(char)
```

char is a string of up to 5 characters.

Comments

The default string for the symbolic display of "zero" entries is 0 (or 0.00..., depending on the number of decimals specified).

3.1.10 The EPS Parameter

This parameter allows the user to control the display of arbitrarily small ("epsilon") entries in output tables. Here "epsilon" means "zero within the level of significance of the display".

Syntax

- A. To display "epsilon" entries as 0 (or 0.00...),

```
SYSTEM EPS=0
```

- B. To display "epsilon" entries using special characters,

```
SYSTEM EPS(char)
```

char is a string of up to 5 characters.

Comments

The default string for the symbolic display of "epsilon" entries is 0 (or 0.00..., depending on the number of decimals specified).

3.1.11 The SPACE Parameter

The report generator within the system produces several blank lines within the heading of reports to increase legibility. At times it is convenient to suppress blank lines so as to increase the number of lines which can be printed on a given page. The SPACE parameter is used to suppress blank lines used for spacing purposes.

Syntax

- A. To turn spacing suppression on,

SYSTEM SPACE=ON

- B. To turn spacing suppression off,

SYSTEM SPACE=OFF

This is the default setting.

3.1.12 The DEMO Parameter

This parameter is used to allow the on-line user to interrupt the printing of a long table and to select from the following options:

- Continue the printing
- Skip the next page of output
- Skip the remaining table entirely and issue the next prompt.

Syntax

- A. To allow printing interruption,

SYSTEM DEMO=ON

- B. To print tables without allowing for interruptions,

SYSTEM DEMO=OFF

This is the default setting.

Comments

When the ON setting is in effect, the printing interruption takes place at the end of an actual output page and a slash, /, prompt is issued.

At the prompt, if the user enters a blank, then the printing will continue with the next page. If the user enters S, then the next page is skipped, and printing continues with the following page. If the user enters anything else, then the remaining table is skipped entirely.

This option applies only in on-line mode, and it is useful when giving "demonstrations" of long printouts, hence the keyword DEMO.

--

3.1.13 The DB1, DB2, and DB3 Parameters

These parameters allow the user to modify the default physical unit numbers assigned to the three table management system files within the data file set.

SYSTEM

I-39

Syntax

SYSTEM DB1=n

SYSTEM DB2=n

SYSTEM DB3=n

n is an integer ($1 \leq n \leq 99$) denoting the physical unit number of a TMS data file

Default: DB1=1, DB2=2, DB3=3.

3.1.14 The INITIAL Parameter

This parameter allows the user to control the allocation of working storage for dataset values.

Syntax

A. To allocate all dataset values when they are defined,

SYSTEM INITIAL = ON

B. To allocated dataset values upon the execution of the first reference to them.

SYSTEM INITIAL = OFF

Comments

The default setting for this parameter is ON. When on, all datasets defined behave as though they were fixed data sets.

3.1.15 The HEADING Parameter

This parameter allows the user to control the printing of page headings.

Syntax

- A. To suppress printing of page headings.

SYSTEM HEADING = OFF

- B. To print page headings,

SYSTEM HEADING = ON

This is the default setting.

3.2 The START Statement

The START statement starts a new program, clears a working space in computer memory (core), and allows the user to define a program descriptor and an area descriptor.

Syntax

START, HEADING (program descriptor), AREA (area descriptor)

Comments

The START statement may not be issued within a model.

3.2.1 The HEADING Option

The program descriptor is an optional descriptor and it provides an overall description of the program or a specific run. The program descriptor will appear along with the page number at the top right-hand corner of each page of tabular program output.

Comments

The keyword HEADING may be optionally omitted.

3.2.2 The AREA Option

In regional analysis programs, the AREA option may be used to specify the name of the area or region being analyzed. The area descriptor will appear in the title of each page of tabular program output, concatenated after the title.

STOP

I-42

3.3 The STOP Statement

The STOP statement ends a program and allows the user to exit the system. All information of the program not saved is lost.

Syntax

STOP

Comments

Execution of a STOP statement within a model causes both a model execution stop and an exit from the system.

3.4 The OPEN Statement

The OPEN statement opens a data file for use. It has two forms depending upon whether the file is to be initialized--i.e. used for the first time or whether the file was initialized by a previous run of the system.

Syntax

- A. Opening a file for the first time,

OPEN (file, n, size)

- B. Opening a previously defined file,

OPEN (file)

file is the physical unit number of the file to be opened

n is the length of the file control vector. If the file is to be used as a TMS file then this parameter equals the number of arrays to be placed on the file. If the file is to be used as a SAVE file then this parameter equals four times the number of modules to be placed on the file plus one added to the product.

size is the total size in words of the file to be allocated

Comments

As was discussed in Subpart 2.1.9 each physical file may have with it a fixed maximum size which is established either by the operating system directives used to establish it, or which is fixed in the system, or both. The particulars depend upon the implementation of the system at a particular site. Before opening a new physical file the user should first decide how much storage space he will require.

Having made this decision he should check with his computer staff to determine the appropriate operating system directives and/or program changes needed.

For certain operating systems the opening of previously defined files is optional. This is not true for all, however. Users at sites where the opening of old files is optional are encouraged to use the statements so as to insure transferability to other operating systems.

The OPEN statement may be issued within or outside of a model.

3.5 The SAVE Statement

The SAVE statement places the information and instructions contained within a program module or within a segment of a program module on a data base for later loading within the same run or within a later run. It is through the use of program module segmentation introduced via the SAVE statement that the user can define and execute large models within a limited amount of core storage.

Syntax

SAVE id. options

id is the identifier of the program module or program module segment being saved

options is a list which may contain the following two options:
XEQ and DATA.

Comments

The program module information saved contains all information and instructions specified in all statements preceding the SAVE statement and including either the last START statement, if the DATA option is not used, or the DATA statement referenced by the DATA option. A saved program module is lost at the end of a batch run or an on-line session, unless it is saved permanently using the procedures of the operating system. The description of these procedures is outside the scope of this handbook. The user should consult his computer staff.

A saved module may be used again by using the LOAD statement.

Normally, the saved module is written on a file whose physical file number is 1. This file is the save data file. To change the physi-

cal file number of the file on which modules are saved the SYSTEM SAVE=n statement may be used.

The SAVE statement may be used either within a model or outside.

3.5.1 The DATA option

The DATA option is used to indicate that only a segment of the program module is to be saved. This is done by specifying the dataset marking the beginning of the segment being saved.

Syntax

DATA = id

id is the identifier of the dataset whose information represents the upper boundary of the program module segment being saved.

3.5.2 The XEQ option

The XEQ option is used to indicate that whenever the saved program module is loaded, then an indicated model within the module is to be executed automatically.

Syntax

XEQ = id

id is the identifier of the model which is to be executed.

Comments

The keyword XEQ may be optionally omitted.

3.6 The LOAD Statement

The LOAD statement loads a previously saved program module or segment of a program module.

Syntax

LOAD id, options

id is the identifier of the program module or program module segment being loaded.

options is a list which may contain the following two options:
 XEQ and DATA.

Comments

The material loaded by this statement may be either an entire program module or a segment of a program module. After a LOAD statement for a program module, the user may continue interacting with the program at the same point of interaction that he left off when he saved the program. If not preceded by a SAVE statement, a LOAD statement permanently erases all information currently in core and replaces it with the LOAD module. In this sense the LOAD statement behaves in the same manner as the STOP statement.

In order to be loaded, a module must reside either on the data file whose physical unit number is contained in the SYSTEM SAVE parameter or whose physical unit number is contained in the SYSTEM LOAD parameter. See Subparts 3.1.6 and 3.1.7 for descriptions of these parameters. If a module with the indicated identifier is present on both data files then the one from the data file indicated by the SAVE parameter will be loaded. If a module with the indicated identifier is present on neither data file, then the system will issue the message "UNABLE TO LOCATE REQUESTED MODULE".

3.6.1 The DATA option

The DATA option is used to indicate that only a segment of a program module is to be loaded. This is done by specifying the dataset marking the beginning of the segment being loaded.

Syntax

DATA = id

id is the identifier of the dataset marking the beginning of the program module segment being loaded.

3.6.2 The XEQ option

The XEQ option is used to indicate that a particular model within the program segment or module being loaded is to be executed automatically.

Syntax

XEQ = id

id is the identifier of the model to be executed.

Comments

If there was an XEQ parameter associated with the SAVE statement which saved the program module, the one associated with the Load takes dominance.

The keyword XEQ may be optionally omitted.

3.7 The TIME Statement

This command is used in dynamic simulation modeling to assign values to four key parameters: the integration interval for time integrals, the beginning time point or lower limit of time integrals, the ending time point or upper limit of time integrals, and the current value for the independent variable time itself. These four parameters are stored by the system in the reserved words DT, BEGINNING, ENDING, and TIME, respectively.

Syntax

`TIME(dt,beginning,ending),option`

`dt` is a number greater than zero defining the size of the time increment for subsequent dynamic simulations.

`beginning` is a number defining the beginning time for subsequent dynamic simulations.

`ending` is a number defining the ending time for subsequent dynamic simulations.

`option` is a list that may include the following options: SIZE

Comments

The beginning and ending parameters define the simulation period for subsequent runs of dynamic simulation models. The simulation will proceed from the beginning to the ending time point in increments of size dt.

Operationally, the dt parameter is the time interval used in specifying the difference equations defining change over time in subsequent dynamic models. Or, the dt parameter is the integration step in sum approximations of time integrals of continuous time-dependent

variables. In this context, the beginning and ending parameters are the lower and upper limits of all time integrals to be computed in subsequent dynamic models.

The constant values for the dt, beginning and ending parameters are stored in the reserved words DT, BEGINNING and ENDING, respectively. Also, the variable value for time itself is stored in the word TIME, which initially is set equal to the value stored in BEGINNING. Upon execution of a dynamic model, the TIME variable takes the values

```
BEGINNING
BEGINNING+DT
BEGINNING+(2*DT)
.
.
.
BEGINNING+(N*DT)
```

where N is the smallest integer to satisfy the following inequality:

$$(N+1) > (ENDING - BEGINNING) / DT + .5$$

The TIME command controls the execution of subsequent dynamic simulation models. Thus, it must be entered prior to the execution of any such models.

Before executing any dynamic simulation models, the DT, BEGINNING and ENDING parameters must have definite values. Also, since most simulations move forward in time, DT must have a positive value, and the ENDING value must be greater than the BEGINNING value.

In a program that has a TIME command, all outputs are subscripted or concatenated by the current value or range of the TIME parameter. For fractional time values, use the SIZE option to specify the number of decimal places to be shown with time subscripts.

3.7.1 The SIZE Option

The SIZE option is used to specify the format for the display of the TIME parameter in all outputs of dynamic simulation programs. Normally, the TIME parameter is displayed as a 5-digit integer (with no decimal digits).

Syntax

SIZE(nc,nd)

nc is an integer specifying the total width in characters in all displays of the TIME parameter as a subscript of simulation results. Its default value is 5.

nd is an integer specifying the number of digits to the right of the decimal point in all displays of the TIME parameter as a subscript of simulation results. Its default value is 0.

3.8 The ADD Statement

The ADD statement adds table management system arrays to data files.

Syntax

ADD (db, table), options

db is an integer value ($1 \leq db \leq 3$) indicating which database within the database set is to contain the array.

table is an integer value indicating the table number of the array.

options is a list that may contain the following option: SIZE

Comments

This statement places the specified table on the data file whose physical unit number is contained in the appropriate SYSTEM parameter--DB1, DB2, or DB3. These are described in Subpart 3.1.4. The values of the array are initialized at real zero. The table number must not exceed the length of the file control vector as specified when the file was opened for the first time. See Section 3.4 for a description of the OPEN statement.

Unless the size options is used, the array being added is assumed to be a scalar--i.e. it is assumed to consist of a single value.

The ADD statement may be used either within or outside of a model.

ADD

I-53

3.8.1 The SIZE Option

The SIZE option is used to indicate that the array being added is not scalar. It specifies the number of dimensions in the array and the number of entries in each dimension.

Syntax

SIZE ($s_1, \dots, s_n$)

n indicates the number of dimensions in the array, ($1 \leq n \leq 10$).

s_i indicates the number of entries in, or size of, the i th dimension, $i=1, n$.

3.9 The INDEX Statement

The INDEX statement defines an index identifier and the information associated with it. Basically, the indexes of a program form the classification scheme for the datasets of the program.

Syntax

`INDEX id(n), options`

or

```
INDEX
id1 (n1), options
id2 (n2), options
.
.
.
END
```

`id` is the index identifier

`n` is a positive integer defining the size of the index

`options` is a list that may include the following options: LABEL and TIME.

Comments

Indexes are used primarily as generalized subscripts of datasets. Thus, index identifiers appear explicitly in DATA statements.

Ordinarily, the index entries are ordered from 1 to `n` in the following sequence: 1,2,3,...`n`.

The size and relative ordering of the entries of an index may be altered locally by the SET statement.

3.9.1 The LABEL Option

This option is used to specify a descriptor for the index.

AD-A085 007

BATTELLE COLUMBUS LABS OH
THE GENERAL AVIATION DYNAMICS MODEL VOLUME III. SYSTEMS MANUAL (U)
JUL 79 M A DUFFY, J M MCCREERY DOT-FAT79A-0003

F/O 1/3

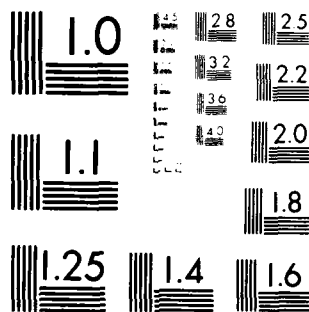
UNCLASSIFIED

FAA-AVP-79-8-VOL-3

ML

3 1/2 4
AD-A085007





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Syntax

LABEL (label)

label is a descriptor for the index being defined. It may contain up to 125 characters.

3.9.2 The TIME Option

This option is used to specify a time index. A time index is used in dynamic models to classify time-dependent datasets, i.e., time series variables whose values are defined at the time points specified by the time index.

Syntax

TIME (list)

list is a list of numbers, t_1 . This list may be defined in two ways:

$$t_1, t_2, \dots, t_n$$

where n is the size of the index, or

$$t_1, t_n$$

in which case n equally spaced time points are generated implicitly by the following formula:

$$t_1, t_1 + h, t_1 + 2h, \dots, t_n$$

where $h = (t_n - t_1)/(n - 1)$ is the constant interval between time points.

3.10 The DATA Statement

The DATA statement is used to define a dataset. Basically, the datasets of a program are the storage arrays for the numeric information that may be manipulated in the equations of the program.

Syntax

DATA id(indexes), options

or

```
DATA
id1 (indexes), options
id2 (indexes), options
.
.
.
END
```

id is the dataset identifier

indexes is the list of previously defined index identifiers classifying the dataset. Up to 10 different indexes may be included in this list. Scalar datasets do not have any indexes.

options is a list that may include the following options: LABEL, VALUE, and TMS.

Comments

If neither of the options VALUE or TMS are used then the dataset is assumed to be a scratch dataset. See Subpart 2.1.3 for a discussion of this notion.

3.10.1 The LABEL Option

The LABEL option is used to define a label or descriptor for the dataset.

Syntax

LABEL (label)

label is a descriptor for the dataset.

Comments

The label of a dataset will appear as the title of all displays of the dataset generated later by SHOW statements. Unlabeled datasets are shown with no titles.

The keyword LABEL may be optionally omitted.

3.10.2 The VALUE Option

The VALUE option is used to define the numeric value or values of the dataset.

Syntax

VALUE (values)

values is a list of numeric values for all storage cells of the dataset.

Comments

The list of values for a multiply-indexed dataset must be entered in the same order as described explicitly in the READ dataset statement, i.e., the first index varies (increases) the fastest, the second index varies the second fastest, and so forth.

When the values of a dataset are specified via a VALUE option, then that dataset becomes fixed. Subpart 2.1.3 discusses the notion of fixed versus scratch datasets.

3.10.3 The TMS Option

The TMS option specifies that the dataset is a table management system, or TMS, dataset. As is discussed in Subpart 2.1.3 TMS datasets have in essence two locations -- one within a TMS array on some data file and one in a temporary location within the program module. The purpose of the TMS option is to specify within which TMS array the dataset values are located, where within the TMS array the dataset values begin, how the dataset values are oriented within the TMS array, and the logical relationship between the dataset and the TMS array. Prior to discussing the syntax of the TMS option itself, these concepts are discussed in general.

TMS arrays are discussed in general in Subparts 2.1.1 and 2.1.11 and in Part 3.8. This discussion is not repeated here. Basically, to identify a TMS array two integer values must be included -- the number of the data base containing the array and the table number of the array. These values are always included within the TMS option.

The specification of the starting location of the dataset value within the TMS array is done via a base point specification. Let T be a TMS array with size specification $(s_1, \dots, s_n)$ as specified in Part 3.7. Now any point P in T can be specified via an ordered sequence $(e_1, \dots, e_n)$, where e_i is the logical entry number of the point in the i th dimension. Obviously $1 \leq e_i \leq s_i$ for all dimensions i . Now a subarray A of T can be defined via the point in T at which the $(1_1, \dots, 1_n)$ point of A occurs and the size of A . The point in T at which the $(1_1, \dots, 1_n)$ point of A occurs is called the base point of A in T .

The specification of the orientation of the dataset within the TMS array hinges upon the notion of transposition. Assume A is an $N \times M$ array and B is the transpose of A . Then B is an $M \times N$ array. Furthermore, it can be said that the rows of B correspond to the columns of A ,

and the columns of B correspond to the rows of A. It is this notion of correspondence in the transpose that is used to define the orientation of the dataset within the TMS array.

Syntax

TMS (db, table, options)

db this integer value ($1 \leq db \leq 3$) indicates which database within the database set contains the array

table this integer value indicates the table number of the array

options includes any number of the suboptions from the following
 list: BASE. TRANSPOSE.

3.10.3.1 The BASE Suboption

The BASE suboption is used to define the base point of the dataset within the TMS array.

Syntax

BASE (val₁, ..., val_n)

n is the number of dimensions in the TMS array identified by db and table

val_i is the ith coordinate for the base point. It may either be an integer constant, a dataset identifier, or one of the system parameters associated with time -- DT, TIME, BEGINNING or ENDING.

Comments

If the BASE suboption is omitted, then the system assumes a base point specified as follows: BASE (1₁, ..., 1_n).

3.10.3.2 The TRANSPOSE Suboption

The TRANSPOSE suboption is used to define the transposition correspondence between the dataset and the TMS array.

Syntax

TRANSPOSE (ind₁, ..., ind_n)

n is the number of dimensions in the TMS array identified by db and table

ind_i is either one of the index identifiers defining the dataset or an asterisk. If it is an index identifier, then the dataset values classified by that index are stored along the ith dimension of the TMS array. If it is an asterisk, then the dataset has no values stored along the ith dimension of the TMS array.

Comments

If the TRANSPOSE suboption is omitted, then the system assumes a one-to-one correspondence between the dataset and the TMS array. This correspondence would be specified as follows: TRANSPOSE (indexes), where indexes is the same list as appeared in the dataset definition.

If the indexes in the TRANSPOSE suboption are fewer than the dimensions in the TMS array, then the remaining dimensions are invariant, i.e., not transposed, relative to the dataset.

3.10.4 Indirect Data Sets

Another variation of the DATA command allows the user to define an "indirect" dataset. This may indirectly represent one of a list of other datasets. Indexes may not be defined with indirects. Indirects may not be used in equations. (See the ELSE DATA option of the ASK statement for examples of use.)

General Syntax

DATA id*

id is the identifier of the indirect dataset. An indirect identifier may be used as a substitute, or indirect representation, for one of a list of "direct" datasets.

3.11 The GRAPH Statement

The GRAPH statement is used to define a graph. Basically, a graph is a set of ordered pairs of numbers that specify and x- and y-coordinates of the points defining the graph.

Syntax

`GRAPH id (n), options`

or

```
GRAPH
id1 (n1), options
id2 (n2), options
END
```

`id` is the graph identifier

`n` is a positive integer defining the size of the graph, i.e., the number of points defining the graph

`options` is a list that may include the following options: X,Y.

Comments

Graphs are used in arithmetic expressions in the RHS of equations to yield by linear interpolation the y-value (or y-values) corresponding to some argument (or arguments).

The argument of a graph may be an arithmetic expression, in general.

The syntax for the general use of a graph in arithmetic expressions is

`id (arithmetic expression)`

where "id" is the graph identifier and an "arithmetic expression" is an expression of numbers, datasets, graphs, and arithmetic operators.

The value of a graph for an arbitrary argument is obtained by linear interpolation or extrapolation between the points defining the graph.

If not specified by the X and Y options, the values of a graph are specified or modified by the READ statement.

Graphs may be used as arguments of other graphs.

The argument of a graph need not be a scalar, i.e., it may be an array of values resulting from an indexed dataset or an arithmetic expression involving such datasets.

3.11.1 The X Option

The X option is used to define the x-coordinates of the n points defining the graph.

Syntax

$$X(x_1, x_2, \dots, x_n)$$

$x_1, x_2, \dots, x_n$ are the numeric values for the x-coordinates of the n points defining the graph.

3.11.2 The Y Option

The Y option is used to define the y-coordinates of the n points defining the graph.

Syntax

$$Y(y_1, y_2, \dots, y_n)$$

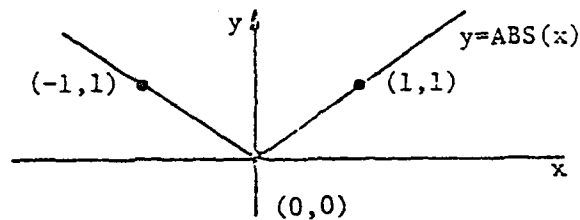
$y_1, y_2, \dots, y_n$ are the numeric values for the y-coordinates of the n points defining the graph.

3.11.3 Examples of GRAPH Statements

1. The Absolute Value Graph

GRAPH ABS(3), X(-1,0,1), Y(1,0,1)

A schematic representation of this graph is:



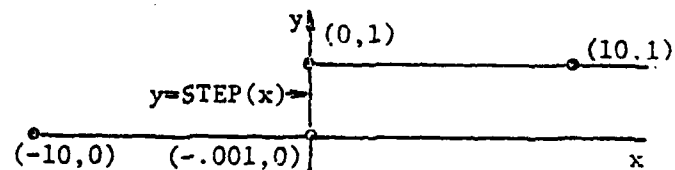
For an arbitrary argument x the value of $ABS(x)$ yields:

$$ABS(x) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

2. The Step Function Graph

GRAPH STEP(4), X(-10,-.001,0,10), Y(0,0,1,1)

A schematic representation of this graph is:



For an arbitrary argument x , the value of $STEP(x)$ yields:

$$STEP(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$

The discontinuity at $x = 0$ is treated approximately to within .001.

3. A Graph of a Graph

Consider the following expression:

$\text{ABS}(\text{STEP}(-20))$

What is its value?

From Example 2, the value of $\text{STEP}(-20)$ is 0. From Example 1, the value of $\text{ABS}(0)$ is 0. Thus, the value of $\text{ABS}(\text{STEP}(-20))$ is also 0.

Similarly, the value of $\text{STEP}(\text{ABS}(10))$ is equal to 1.

4. A Graph with a Multidimensional Argument

Consider the dataset DR classified by an index whose size is 3. If the 3 values of this dataset are: $\text{DR}(1)=-3$, $\text{DR}(2)=2$, $\text{DR}(3)=10$, then the expression $\text{ABS}(\text{DR})$ takes the following 3 values: 3, 2, and 10.

3.12 The MODEL Statement

The MODEL statement is used to introduce a model identifier and the equations and instructions associated with it for later execution.

Syntax

```
MODEL id  
(model instructions)  
END
```

id is the model identifier.

The model instructions are various statements that define the equations and other instructions of the model. These instructions are only interpreted and stored. They are not executed.

The END statement terminates the compilation of the model and returns the user to command mode.

Comments

The MODEL statement initiates compile mode. Under compile mode, all model instructions are interpreted and stored, but not executed. The compiled model instructions are not executed until the model is called into execution.

Model execution is initiated by entering the model identifier. Model execution proceeds sequentially through the compiled instructions of the model in the order in which they are compiled.

If the AREA option is specified in the START statement, a model execution statement may contain an area descriptor characterizing the region associated with this execution of the model.

There are two types of models -- static and dynamic. Both use the same syntax when they are initiated. Dynamic models contain within them the RATE and LEVEL statements. These are discussed in Parts 3.22 and 3.23.

3.12.1 Examples of MODEL Statements1. Model Compilation

```

INDEX ROW(3)
DATA DR (ROW), LABEL (DATA BY ROW)
DATA TOT, LABEL (TOTAL)
MODEL SUMR
READ DR
TOT=SUM(R) (DR(R))
SHOW DR.2
SHOW TOT.2
END

```

Upon model execution (see Step 1 below) the model SUMR will:

- Prompt the user to enter the 3 ROW values for the dataset DR (see Step 2 below).
- Compute the value of the dataset TOT by summing over the 3 ROW values of the dataset DR
- Show the values of the dataset DR, to 2 decimal places (see Step 3 below).
- Show the value of the dataset TOT, to 2 decimal places (see Step 4 below).

2. Model ExecutionStep

```

1      SUMR
2  DATA 1 2 3
3

```

A SAMPLE PROGRAM PAGE 1

	DATA BY ROW		
ROW1	1.00	ROW2	2.00
ROW3	3.00		

```

4  TOTAL      6.00

```

3.13 The TABLE Statement

The TABLE statement defines for later execution a table identifier, the datasets associated with it, and its format.

To call a table into execution, i.e., to print it out or to display it, simply enter its identifier.

Syntax

TABLE id(indexes), DATA(datasets), TITLE(title), SIZE(sw,cw)

id is the table identifier

indexes is a list of previously defined index identifiers. These indexes define the spanner and subtitles of the table. The order of indexes in the list is important (see the comments).

datasets is a list of previously defined dataset identifiers. This list may also contain slashes, / , and strings enclosed in parentheses. Upon table execution, the strings are displayed as part of the table stub in the same relative order as they were specified in this list. The slashes manifest themselves as blank lines or rows in the table in the same relative order as they were specified in this list. Upon execution the values of the datasets included in this list become the table body.

title is a descriptor which upon execution becomes the centered title of the table

sw is a positive integer defining the table stub width (in characters)

cw is a positive integer defining the column widths (in characters) for the table.

TABLE

I-70

Comments

A table execution statement may be entered either in command or in compile (i.e., within a model) mode.

The order of display of the table indexes is: the missing index classifies the table rows, the last index classifies the table columns. All other indexes classify the various row by column table sections in a left-to-right order, i.e., the first index varies the fastest, the second index varies the second fastest, etc.

For datasets not classified by the missing index, the dataset label acts as a stub.

3.13.1 Examples of TABLE Statements1. Table Definition

```
INDEX ROW(3)
INDEX COL(2)
DATA DRC(ROW,COL), LABEL(DATA BY ROW AND COLUMN)
DATA TOTC(COL), LABEL(TOTALS BY COLUMN)
TABLE STAB(COL), DATA(TOTC.2,/, DRC.2,/, (NOTE. THESE ARE DUMMY VALUES)),
      SIZE(20,10), TITLE(A SAMPLE TABLE)
```

2. Table Execution

```
DRC=2
TOTC(C)=SUM(R) (DRC(R,C))
STAB
```

A SAMPLE PROGRAM PAGE 1

A SAMPLE TABLE

	COL ONE	COL TWO
TOTALS BY COLUMN	6.00	6.00
ROW1	2.00	2.00
ROW2	2.00	2.00
ROW3	2.00	2.00

NOTE. THESE ARE DUMMY VALUES

END

I-72

3.14 The END Statement

The END statement is used to terminate the compilation of a model or the section-form definition of a set of indexes, or a set of datasets, or a set of graphs, or a set of tables. It is also used to end the range of a conditional statement such as IF or DO.

Syntax

END

Comments

To distinguish between END statements terminating various structured sections of code, arbitrary comments may be used following the word END and at least a blank or a comma.

3.15 The READ Statement

The READ statement is the basic input operation of the system and is used to read information into the structure of indexes, datasets, or graphs.

Syntax

READ id, options (information being read in)

id is the identifier of the index, dataset, or graph where the information being read in will be stored

options these options vary depending on whether information is read into an index, a dataset, or a graph. See the READ index, READ dataset, and READ graph statements.

Comments

To satisfy the READ statement, information must be entered in the cards or input lines following the READ card.

The type of information entered depends on whether it is read into an index, dataset, or graph.

The amount of information entered depends on the local dimensions of the structural element identified in the READ statement.

The manner in which the information is entered on the input cards or lines depends both on the implicit structure of the structural element identified in the READ statement and an explicit format specified in the options of the READ statement.

READ statements may be specified either in command or in compile mode (except READ index). In command mode, the information being read in must follow immediately the READ statement. In compile mode, the information being read in must not follow the READ statement which is specified within some model; rather, it must be entered after a prompt issued by the

model execution statement.

3.15.1 The READ Index Statement

The READ index statement is used to read information into the structure of an index. With this statement the user may introduce the following information:

- The stub of an index
- The spanner of an index

Syntax

READ id, options
{index entries}

id is an index identifier

options may include format specifications for either the stub or the spanner entries of the index.

Comments

The index entries following the READ statement must be specified on n entry cards or input lines, where n is the size of the index.

The manner in which the index entries are keypunched on the cards or typed in the input lines depends on the index option being specified.

The READ index statement may be issued only outside of a model, and is not sensitive to the SET command.

3.15.1.1 The Stub Option

The stub option is used with the READ index statement to read a stub into the structure of an index. The index stub is the set of descriptors that will become the row descriptors for the displays of all datasets

READ

I-75

that are classified by the index.

Syntax

(ic,lc) {n stub entries}

ic is a positive integer (≤ 80) defining the initial column on each stub card or line where the stub entry begins

lc is a positive integer ($lc \geq ic$) defining the last column on each stub card or line where the stub entry ends.

Comments

Each of the n stub cards or lines following the READ statement must contain a stub entry for the index identified in the READ index statement. Each stub entry must be located between the ic and lc columns of each card.

The number of stub entries is n, where n is the size of the index identified in the READ index statement. These entries are stored in the order: 1, 2, 3, . . . , n.

Both format parameters ic and lc may be omitted, in which case the default values $ic = 1$ and $lc = 20$ apply.

3.15.1.2 The Spanner Option

The spanner option is used with the READ index statement to read a spanner into the structure of an index. The index spanner is the set of descriptors that will become the column headings for the tabular displays of all datasets that are classified by the index.

Syntax

(ic,lc,nc) {n spanner entries}

- ic is a positive integer (≤ 80) defining the initial column on each spanner card or line where the spanner entry begins
- lc is a positive integer ($lc \geq ic$) defining the last column on each spanner card or line where the spanner entry ends
- nc is a positive integer defining the number of columns or characters (including blanks) in each line section of the spanner. The following exact relationship must be satisfied:

$$(lc - ic + 1)/nc = n1$$

where n1 is a positive integer denoting the number of lines in each column heading of the spanner.

Comments

Each of the n spanner cards or lines following the READ statement must contain a spanner entry for the index identified in the READ statement. Each spanner entry must be located between the ic and lc columns of each card or line, forming n1 groups of nc characters each.

The number of spanner entries is n, where n is the size of the index identified in the READ index statement. These entries are stored in the order: 1, 2, 3, . . . , n.

The format parameters *ic*, *lc*, and *nc* are necessary for specifying the line structure of each spanner and thus may not be omitted.

3.15.2 The READ Dataset Statement

The READ dataset statement is used to introduce values into the structure of a dataset.

Syntax

```
READ(indexes) id, (ic,lc,nc)
{data entries}
```

id is a dataset identifier

indexes is an ordered list of the indexes, minus one, classifying the dataset. The missing index plays an important role (see the comments). The ordering of the indexes in the list is also important (see the comments). By definition, this list is not necessary for scalar datasets. This list may be omitted, in which case the default ordering of indexes applies (see comments).

ic is a positive integer (≤ 80) indicating the initial column on each data card or line where the reading operation will start

lc is a positive integer ($lc \geq ic$) indicating the last column on each data card or line where the reading operation will end

nc is a positive integer indicating the number of columns or characters forming each data entry on each data card or line. The following exact relationship must be satisfied:

$$(lc - ic + 1)/nc = ne$$

READ

I-78

where n_e is a positive integer denoting the maximum number of entries that may be entered on each data card.

Comments

The fixed format parameters ic , lc , nc may be omitted, in which case the values of the dataset may be entered in free format. In free format, the values of a dataset may be entered anywhere on an input card or line provided they are separated from each other by at least one blank or a comma.

3.15.2.1 Examples of READ Dataset Statements

The reading operation specified by a READ dataset statement can best be demonstrated by means of examples.

Consider the dataset $A(I,J,K)$ classified by the indexes I , J , and K whose sizes are 3, 2, and 2, respectively. I is the first, J is the second, and K is the third index of the dataset A .

Example A. To satisfy the statement

READ A

the values of A must be arranged on the data entry cards as follows:

<u>Card</u>	<u>Values</u>		
1	$A(1,1,1)$	$A(2,1,1)$	$A(3,1,1)$
2	$A(1,2,1)$	$A(2,2,1)$	$A(3,2,1)$
3	$A(1,1,2)$	$A(2,1,2)$	$A(3,1,2)$
4	$A(1,2,2)$	$A(2,2,2)$	$A(3,2,2)$

In this arrangement, the first index of A , I , varies the fastest across each card, the second index, J , varies the second fastest, and the third

READ

I-79

index, K, varies the third fastest.

Example B. To satisfy the statement

READ(I,J) A

the values of A must be arranged as follows:

<u>Card</u>	<u>Values</u>	
1	A(1,1,1)	A(1,1,2)
2	A(2,1,1)	A(2,1,2)
3	A(3,1,1)	A(3,1,2)
4	A(1,2,1)	A(1,2,2)
5	A(2,2,1)	A(2,2,2)
6	A(3,2,1)	A(3,2,2)

In this arrangement, the missing index K varies the fastest across each card, the index I varies the second fastest, and the index J varies the third fastest.

Example C. To satisfy the statement

READ(I,J,K) A

the values of A must be arranged on the data entry cards as follows:

<u>Card</u>	<u>Values</u>
1	A(1,1,1)
2	A(2,1,1)
3	A(3,1,1)
4	A(1,2,1)
5	A(2,2,1)
6	A(3,2,1)

<u>Card</u>	<u>Values</u>
7	A(1,1,2)
8	A(2,1,2)
9	A(3,1,2)
10	A(1,2,2)
11	A(2,2,2)
12	A(3,2,2)

Free versus Fixed Format. The format parameters *ic*, *lc*, *nc* are optional. If specified, the dataset values must be entered according to the fixed format specified by these parameters. If not specified, the dataset values may be entered in free format, provided they are separated by commas or blanks.

Examples A, B, and C are in free format. The following example is in fixed format.

Example D. To satisfy the statement

READ A, (11,40,10)

the values of A must be arranged on the data entry cards in the same order as shown in Example A. However, the entries on each card must lie between columns 11 and 40, and each entry must not exceed 10 columns or characters. Columns 1 to 10 are ignored by the reading operation. However, they may be used for data identification purposes.

In this example the format parameters *ic*, *lc*, and *nc* have the values:

ic = 11
lc = 40
nc = 10

Thus, the number of entries per card is:

$$n_e = (l_c - i_c + 1)/n_c = (40 - 11 + 1)/10 = 3.$$

Note. A READ dataset operation is not satisfied unless all values of the dataset are entered appropriately.

Normally, a dataset contains n values, where n is equal to the product of the sizes of the indexes classifying the dataset. Thus, the sample dataset A contains $3 \times 2 \times 2 = 12$ values, where 3, 2, and 2 are the sizes of the indexes, I, J, and K.

However, the size and relative ordering of an index may be modified by a SET index statement. Thus, the range and structure of a READ dataset operation will be modified by those SET index statements that modify the indexes of the dataset. This is illustrated in the next example.

Example E. To satisfy the following READ statement:

SET I(1,3), J(2)

READ A

the values for A must be arranged as follows:

<u>Card</u>	<u>Values</u>	
1	A(1,2,1)	A(3,2,1)
2	A(1,2,2)	A(3,2,1)

3.15.3 The READ Graph Statement

The READ graph statement is used to introduce values into the structure of a graph.

Syntax

READ id, options (n x-values) (n y-values)

id is a graph identifier
n is the graph size
options include the optional specification of fixed format parameters.

Comments

To satisfy the READ graph statement n x-values and n y-values must be entered, where n is the size of the graph.

The n x-values must be entered first on as many cards or input lines as are necessary, followed by the n y-values.

Since the x- and y-values form pairs of coordinates on the x-y plane for the n points of the graph, the n y-values must be entered in the same order as the corresponding n x-values.

In free format the x- and y-values may be entered in the desired order separated by commas or blanks.

The fixed format option may be specified as follows:

(ic,lc,nc)

where

ic is a positive integer (≤ 80) denoting the initial column on each data card or line where the reading operation will start

lc is a positive integer ($lc \geq ic$) denoting the last column on each data card or line where the reading operation will end

nc is a positive integer (≤ 80) indicating the number of columns or characters forming each x- or y-values on each data card or line. The following exact relationship must be satisfied:

$$(lc - ic + 1)/nc = ne$$

where ne is a positive integer denoting the maximum number of entries that may be entered on each data card.

READ

I-83

3.15.3.1 Examples of READ Graph Statements

1. The Absolute Value Graph Specified in Free Format

GRAPH ABS(3)

READ ABS

-1 0 1

1 0 1

2. The Step Function Graph Specified in Fixed Format

GRAPH STEP(4)

READ STEP, (6,25,5)

STEPX -10 -.001 0 10

STEPLY 0 0 1 1

Since the first 5 columns of each data card are ignored by the READ operation, they may be used for identification. In this example,

1c = 6

1c = 25

nc = 5

and the number of entries per card, ne, is:

$$ne = (1c - 1c + 1)/nc = (25 - 6 + 1)/5 = 4,$$

the number of points defining the graph STEP.

3.16 The SET Statement

The SET statement is used to set locally the range of an index and the relative ordering of its entries.

The range of an index may be a positive integer less than or equal to n , the size of the index.

The normal ordering of the entries of an index is: 1, 2, 3, . . . , n . The SET index statement may modify this ordering.

Syntax

SET id. setting

id is an index identifier

setting is the setting for the index entries. This setting may be specified in two different ways:

(value list)

*

The * setting restores the index to its normal setting as established in the index definition. - i.e., range = n and entry ordering: 1, 2, . . . , n

The value list may include up to n positive values or as many nonoverlapping value intervals as are necessary to cover the set {1, 2, . . . , n }. A value interval is specified by the two values defining the interval separated by a hyphen, -. The two values of an interval may be specified either in ascending or in descending order.

Comments

A single SET statement may be used to set more than one index, as follows:

SET id_1 (setting₁), id_2 (setting₂), ...

The setting of an index stays in effect until the index is reset by another SET statement and it affects the following instructions: READ dataset, SHOW dataset, and subscripted equations.

Note. The SET index statement is very powerful because it affects subsequent operations. Thus, it must be used with caution and only locally, i.e., once the operations which should be done under the influence of a SET statement are completed the indexes affected should be restored to their normal setting by using the * option.

3.16.1 Examples of the SET Statement

```
INDEX
ROW(3)
COL(2)
TYPE(5)
END
```

Normally, the ROW index has size 3 and its entries are ordered as follows:

<u>Entry</u>	<u>Order</u>
ROW(1)	1
ROW(2)	2
ROW(3)	3

Normally, the COL index has size 2 and its entries are ordered as follows:

SET

I-86

<u>Entry</u>	<u>Order</u>
COL(1)	1
COL(2)	2

Normally, the TYPE index has size 5 and its entries are ordered as follows:

<u>Entry</u>	<u>Order</u>
TYPE(1)	1
TYPE(2)	2
TYPE(3)	3
TYPE(4)	4
TYPE(5)	5

1. Reordering Index Entries

A. SET ROW(3,1,2)

The range of this setting is 3. The index entries are reordered as follows:

<u>Entry</u>	<u>Order</u>
ROW(3)	1
ROW(1)	2
ROW(2)	3

B. SET TYPE(5,1-3)

The range of this setting is 4. The index entries are reordered as follows:

<u>Entry</u>	<u>Order</u>
TYPE(5)	1
TYPE(1)	2
TYPE(2)	3
TYPE(3)	4

SET

I-87

2. Setting Two Indexes

SET ROW(3-1), COL(2)

3.17 The SHOW Statement

The SHOW statement is the basic output instruction of the language and it is used to show the values of a dataset.

Syntax

SHOW(sw,cw) id.n, option

id	is a dataset identifier
sw	is a positive integer denoting the stub width in characters for the row descriptors of the dataset display
cw	is a positive integer denoting the column width in characters for the column headings of the dataset display
n	is a positive integer (<cw) denoting the number of decimal digits to be shown with each dataset value
option	is one of the optional parameters ORDER, TITLE, and TOTAL.

Comments

The stub width, sw, and column width, cw, parameters may be omitted. If omitted, the default values in effect are: sw = 15
cw = 8.

The decimal point and the decimal digits parameter, n, may be omitted. If omitted, the default value in effect is: n = 0. The dataset values are shown as integers.

A scalar dataset is shown on a one-line display consisting of the label of the dataset, if specified, followed by the value of the dataset.

An indexed dataset is shown on a centered, paged, and titled tabular display that is properly described by the entries of the indexes classifying the dataset. The page heading is shown at the top right-hand corner of the display and it consists of the program descriptor specified in the START statement followed by the page number. The title is the label of the dataset. The row descriptors of the display are the stub entries of the first index of the dataset. The column headings of the display are the spanner entries of the second index of the dataset. The stub entries for the third, fourth, etc., indexes of the dataset are the subtitles of the row-by-column sections of the display.

The tabular display of the dataset is designed to fit properly within an 8-1/2 x 11 inch page for direct insertion in reports.

Large dataset displays are segmented into sections, each section fitting properly within an 8-1/2 x 11 inch page.

Dataset values greater than or equal to 1,000 are shown with commas separating the value into 3-digit groups.

3.17.1 The ORDER Option

The ORDER option allows the user to control the order in which the display is organized--i.e., which index is placed in the rows, which in the columns, and so forth.

Syntax

ORDER(indexes)

indexes is a list of indexes specifying the order of the indexes in the display. The first index classifies the rows and the second the columns of the dataset being shown. All other indexes classify as subtitles the row-by-column sections of the dataset being shown.

3.17.2 The TITLE Option

The TITLE option is used to specify a title for the dataset being shown. If the TITLE option is not used, the title for the tabular display is the label of the dataset being shown.

Syntax

TITLE (list)

list is the title to be shown for the dataset. This list may contain one or more of the following items:

- (1) a string of up to 125 characters, and/or
- (2) the special character /, to skip to a new line for multiple line titles, and/or
- (3) id.L, where the notation "id.L" refers to the label of the dataset whose identifier is "id".

If Item 1 is not listed alone, then it must be enclosed in parentheses.

The items in a multi item list must be separated by blanks or commas.

3.17.3 The TOTAL Option

The TOTAL option allows the user to specify that all (or a subset) of the values of a dataset be totaled and the totals (or subtotals) be shown in the display of the dataset values.

Syntax

TOTAL(indexes)

indexes is a list of identifiers specifying those indexes that classify those values of the dataset which are to be totaled. If omitted, the values of the dataset will be totaled and subtotaled over each and all indexes classifying the dataset. If included, only those values will be subtotaled that are classified by the indexes specified.

3.18 Equations

This is a continuation of the discussion on equations given in Part 2.1.

Syntax

id=arithmetic expression

id is the identifier of a previously-defined dataset (explicitly subscripted or not).

Arithmetic expression is a well-formed, ordered sequence of numerical constants, previously-defined identifiers, mathematical operators, and parentheses.

Comments

The following are additional comments to those given in Part 2.1. These comments address the following topics:

- left-to-right hierarchy
- dummy or symbolic subscripts
- implicit subscripts
- matrix equations
- division by zero
- linear interpolation

Left-to-Right Hierarchy. In the absence of parentheses, the arithmetic operations on the right-hand side of equations are compiled and executed from left to right.

All arithmetic operators $+$, $-$, $*$, $/$, and $**$ for addition, subtraction, multiplication, division, and exponentiation, respectively, are of equivalent rank in the processing of equations by the system. In other computer languages and in standard algebraic notation this is not the case. In FORTRAN, for example, there are three levels of hierarchy: exponentiation is processed first, then division and multiplication, and then addition and subtraction.

Hierarchy rules for processing arithmetic expressions are very important. Without a fixed set of rules, expressions such as $7*4+3/2$ could have several meanings and values:

$$\begin{aligned}(7*4)+(3/2) &= 29.5 \\ \text{or } ((7*4)+3)/2 &= 15.5 \\ \text{or } 7*(4+3)/2 &= 24.5 \\ \text{or } 7*(4+(3/2)) &= 38.5\end{aligned}$$

Which is the correct answer? Algebraic hierarchy rules state that multiplication and division must be performed before addition. Thus, algebraically the correct answer is $(7*4)+(3/2)=29.5$.

In the system hierarchy, however, the arithmetic operations are performed from left-to-right, regardless of operation type. Thus, the arithmetic expression $7*4+3/2$ is processed as follows:

- The multiplication $7*4$ is done first yielding 28
- The addition $28+3$ is done next yielding 31
- The division $31/2$ is done next yielding 15.5

The operations in an arithmetic expression are performed from left to right with each operator using the preceding result as the operand. Even though this convention is different than the standard FORTRAN convention, it may be easier to remember, especially by beginner programmers, and it has the advantage that the actual programming code required to process it is shorter and faster.

To alter the implied left-to-right hierarchy in which arithmetic operations are performed, the user may employ parentheses. The meaning and use of parenthetical groups in arithmetic expressions is discussed in Part 2.1.

Dummy Subscripts. The following single equation:

$$DRC(R,C) = DR(R) * DC(C)$$

defined in terms of the following indexes and datasets:

INDEX ROW(3)
INDEX COL(2)
DATA DRC(ROW,COL)
DATA DR(ROW)
DATA DC(COL)

is equivalent to the following set of 6 single-valued equations:

$$\begin{aligned} DRC(1,1) &= DR(1) * DC(1) \\ DRC(2,1) &= DR(2) * DC(1) \\ DRC(3,1) &= DR(3) * DC(1) \\ DRC(1,2) &= DR(1) * DC(2) \\ DRC(2,2) &= DR(2) * DC(2) \\ DRC(3,2) &= DR(3) * DC(2) \end{aligned}$$

Here, the dummy subscripts R and C - symbolic for the indexes ROW and COL, respectively - take on the values:

R: 1,2,3
C: 1,2

as dictated by the respective sizes of the ROW and COL indexes.

Any TIE of type 2 can be used as a dummy subscript, provided it has not been used previously as an identifier. Usually, single-character type 2 TIEs make convenient dummy subscripts.

Actual index identifiers may serve as their own dummy subscripts, of course.

As can be seen from the example above, dummy subscripting represents substantial simplification in notation. The power of this notation may be appreciated more fully when compared with its FORTRAN equivalent, the DO Loop.

Explicit subscripts, i.e., integer constants or dataset identifiers, must have nonzero, positive values never exceeding the sizes of the indexes which they represent.

A dummy subscript in the left-hand side of an equation takes on all the values allowed by the size - actually, the current range as specified by the last SET index statement - of the index it represents.

For a multiply-subscripted dataset the order of the dummy subscripts must correspond to the order of the indexes classifying the dataset. For example, the dummy subscripts I,J,K in the following dataset reference

DRTC(I,J,K)

correspond to the indexes ROW, TYPE, and COL respectively that classify the dataset DRTC, as specified in the dataset definition:

DATA DRTC(ROW, TYPE, COL)

An equation with dummy subscripts corresponds to a set of independent, single-valued equations. The number of single-valued equations represented by such an equation is equal to the product of the current ranges of the dummy subscripts classifying the dataset in the left-hand side of the equation. The dummy subscripts of the right-hand side of an equation take on the same values as the subscripts with identical symbols in the left-hand side.

Implicit Subscripts. In the equation

$$DR=2$$

the index ROW classifying the dataset DR is implicit, i.e., not specified explicitly as a subscript of DR. This equation is equivalent to the following 3 equations - one for each of the 3 entries of the ROW index classifying the DR dataset:

$$DR(1) = 2$$

$$DR(2) = 2$$

$$DR(3) = 2$$

This example brings out the notational economy of implicit subscripting, which can be considerable in equations involving multi-dimensional datasets classified with large indexes.

Matrix Equations. Equations with dummy or implicit subscripts may be referred to as matrix equations. A single matrix equation corresponds to a number of independent, single-valued equations. The number of single-valued equations represented by a matrix equation is equal to the product of the current ranges of the dummy or implicit subscripts classifying the dataset in the left-hand side of the equation.

The dummy or implicit subscripts of the right-hand side of a matrix equation take on the same values as the corresponding subscripts of the left-hand side. A redundant subscript in the right-hand side, i.e., a subscript with no corresponding subscript in the left-hand side, is set at the first sequence value in the current setting of the index corresponding to the subscript. For example, the notation

$$SET K(2,1), L^*$$

$$A(I, J) = B(I, J, K, L)$$

is equivalent to

$$A(I,J) = B(I,J,2,1)$$

where 2 is the first sequence number in the current setting of the index K, and 1 is the first sequence number in the current setting of the index represented by the dummy subscript L. However the notation

$$\begin{aligned} &\text{Set } K(2,1), L* \\ &A = B \end{aligned}$$

is equivalent to

$$A(I,J) = B(I,J,1,1)$$

Dummy and implicit subscripts must not be mixed. When using dummy or explicit subscripts, all subscripts of all datasets in the equation must be specified explicitly and in the appropriate order. When using implicit subscripts, all subscripts of all datasets in the equation must be implicit.

If not subscripted with explicit or dummy indexes, the datasets in the right-hand side of a matrix equation must be classified by implicit indexes that match exactly the implicit indexes classifying the dataset in the left-hand side of the equation.

For example, the equation

$$A = B$$

is malformed if the dataset A is indexed by the ROW (size 3) index and the dataset B is indexed by the COL (size 2) index. This can be seen from the following single-cell equations that are equivalent to $A = B$:

$$A(1) = B(1)$$

$$A(2) = B(2)$$

$$A(3) = B(3)$$

However, what is the value of $B(3)$? The subscript 3 is out of range for the dataset B. The dataset B has only two values, $B(1)$ and $B(2)$, as dictated by the size of the COL index that classifies it. In this case, the system uses the first value of the dataset B, i.e., it interprets the third equation above as follows:

$$A(3) = B(1)$$

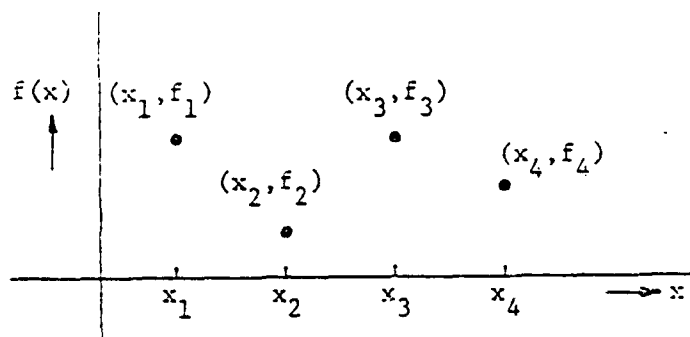
This is true in general: a subscript out of range in the right-hand side of an equation is set to one.

Division by Zero. Since all datasets are initialized at zero, it is possible to accidentally introduce divisions by zero in the execution of a program. Divisions by zero are computed as zero; not infinity. A division by zero causes neither a program execution error nor a program stop.

Linear Interpolation. Linear interpolation is used implicitly by the system to evaluate the values of graphs for arbitrary arguments.

In general, the method of linear interpolation is a method for approximating the value of a function at some arbitrary point from the known values of the function of a finite set of points.

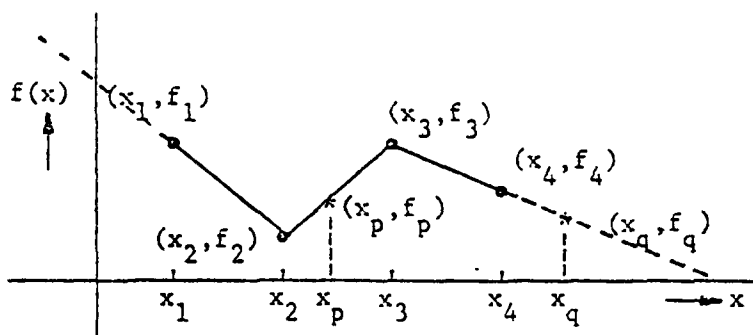
Consider, for example, the 4-point function $f(x)$ depicted below, whose values are given only at four points:



These four points are defined by the pairs of numbers

(x_1, f_1) , (x_2, f_2) , (x_3, f_3) , and (x_4, f_4) .

What is the value of this function at an arbitrary point x ? If it is assumed that the four points are connected by straight lines in the manner shown:



then the value of the function at an arbitrary point x_p is given by the point of intersection of a vertical line drawn at x_p and the line connecting the two known points of the function that are nearest to x_p , on either side.

In terms of the diagram above, the value f_p of the function at x_p is given by the linear interpolation formula:

$$f_p = f_2 + ((f_3 - f_2) * (x_p - x_2) / (x_3 - x_2))$$

where (x_2, f_2) and (x_3, f_3) are the two points that are nearest to (x_p, f_p) on either side.

Similarly, the value f_q of the function at x_q is given by the linear extrapolation formula:

$$f_q = f_3 + ((f_4 - f_3) * (x_q - x_3) / (x_4 - x_3))$$

which is given in terms of the points at x_3 and x_4 , which are nearest to x_q but lie to the left of it.

In general, the value of the n-point function, $f(x)$, at an arbitrary point x is given by the general linear interpolation formula:

$$f(x) = f_i + ((f_{i+1} - f_i) * (x - x_i) / (x_{i+1} - x_i))$$

where x_i and x_{i+1} are those x coordinates which are nearest to x on either side and f_i and f_{i+1} are the corresponding function values.

If x is less than x_1 or greater than x_n , then the following linear extrapolation formulae apply:

$$f(x < x_1) = f_1 + ((f_2 - f_1) * (x - x_1) / (x_2 - x_1))$$

$$f(x > x_n) = f_{n-1} + ((f_n - f_{n-1}) * (x - x_{n-1}) / (x_n - x_{n-1}))$$

3.19 Model Execution Statement

The model execution statement executes the statements in order within a previously defined user model. It can be used to execute a static model or a dynamic model. It can also be used to resume the execution of a dynamic model.

Syntax

- A. Executing a model,

`model`

- B. Resuming a dynamic model,

`model = ending`

`model` is the identifier of a previously defined model

`ending` is the new value of ending time until which the model execution should continue.

Comments

See Part 3.7 on the TIME statement and Part 3.12 on the MODEL statement for descriptions of the manner in which models are executed.

3.20 Table Execution Statement

The table execution statement generates a tabular report as specified by the identified table.

Syntax

table

table is the identifier of the table specifying the tabular report.

Comments

See Part 3.13 on the TABLE statement for details of specifying tabular reports.

3.21 The RATE Statement

The RATE statement is used in the dynamic simulation models and it has two functions: (a) it signals the end of the initial section of the model and the beginning of the rate section, and (b) it declares that the time dependent variables be computed at each time point of the simulation by linear interpolation or extrapolation from specified exogenous time series.

Syntax

RATE(exogenous variables list)

where the exogenous variables list is a list of correspondences between previously-defined exogenous time series and time-dependent variables that must be used locally in the rate (and/or level) section of a dynamic simulation model. This list must be enclosed in parentheses, and each list item must be formulated as follows:

exogenous time series=local variable

where exogenous time series is a previously-defined dataset that is indexed by a time index, and local variable is a previously-defined data set that is used explicitly in the rate (and/or level) section of a dynamic model and is computed at every time point of the simulation. Based on this equivalence, the values of the local variable will be computed at the arbitrary time points of the dynamic simulation by linear interpolation or extrapolation that is based on the fixed time points defining the exogenous time series.

Comments

The RATE and LEVEL statements are mandatory in dynamic simulation models. Each dynamic simulation model must have one RATE and

one LEVEL statement in order to separate the model equations into three sections: The initial section, the rate section, and the level section.

The initial section is the first section of the model and its equations are evaluated at the beginning time point (or interval) of the simulation period.

The rate section is the second section of the model and its equations are evaluated at each time point (or interval) of the simulation run. In contrast to level equations, both sides of rate equations are evaluated at the same time point (or interval).

The level section follows the rate section and its equations are also evaluated at each time point (or interval) of the simulation. The left-hand side of each level equation, however, is evaluated at $TIME+DT$ in terms of the time variables in the right-hand side which are evaluated at $TIME$, the previous time point (or interval). It is the equations of the level section which move the dynamic variables through time.

Only those exogenous time series that are used explicitly in the rate or the level section need be included in the exogenous variables list of the RATE instruction.

An exogenous time series must not be used explicitly in the equations of the rate or level sections, for its fixed time index points must be distinguished from the arbitrary time points of the $TIME$ parameter of the simulation run.

The RATE statement may only be used inside the MODEL statement, i.e., you must not use it while in command mode.

3.22 The LEVEL Statement

The LEVEL statement is used in dynamic simulation models, and it has two functions: (a) it signals the end of the rate section and the beginning of the level section, and (b) it declares that endogenous variables be computed and stored at the fixed time points of the time indexes classifying the output variables time series. The values of an output time series at each time point of the time index are set equal to the values of the local endogenous variable corresponding to the nearest simulation time point plus or minus $DT/2$.

Syntax

LEVEL(endogenous variables list)

where the endogenous variables list is a list of correspondences between output time series and endogenous variables that are used locally in the equations of the level (and/or rate) section of a dynamic simulation model. This list must be enclosed in parentheses, and each list item must be formulated as follows:

output time series=endogenous variable

where output time series is a previously-defined data set that is indexed by a time index, and endogenous variable is a previously-defined data set that is used explicitly in the level (and/or rate) section of a dynamic simulation model. Based on this equivalence, the values of the output time series will be computed and stored at the fixed time points of the time index classifying the series. The value of an output series at a time point T is set equal to the computed value of the corresponding endogenous variable that is associated with the interval $(T-DT/2, T+DT/2)$. This interval is closed at $T-DT/2$ and open at $T+DT/2$: if T is the exact midpoint of the interval, then the $T-DT/2$ value applies.

Comments

The LEVEL statement is mandatory in each dynamic simulation model, for it separates the rate from the level section of the model, and it "moves" the model variables through time.

The LEVEL statement causes the TIME parameter to be incremented by DT units from its value in the preceding rate section.

The variable in the left-hand side of each level equation is evaluated at TIME+DT, the incremented time point or interval, in terms of the variables in the right-hand side of the equation which are evaluated at TIME, the previous time point or interval.

The output time series in the endogenous variables list of the LEVEL statement must not be referenced explicitly in the equations of a dynamic simulation model.

Only those endogenous variables that you intend to save for later use as time series need to be included in the endogenous variables list of the LEVEL statement.

The LEVEL statement must not be used in command mode, i.e., it may only be used inside dynamic simulation models.

3.23 The WRITE Statement

Though the reading and writing of TMS array values is normally taken care of automatically via the definition and manipulation of TMS datasets, there are occasionally times when the user might simply wish to treat the TMS array data file as though it were a standard database. The WRITE statement allows the user to explicitly write dataset values to a TMS array.

Syntax

WRITE data, TMS (pram)

data is the identifier of the dataset containing the values
 to be written

pram contains a set of TMS parameters formed exactly as described
 in Subpart 3.10.3.

Comments

The keyword TMS may be optionally omitted.

3.24 The INPUT Statement

The INPUT statement allow the user to explicitly input dataset values from a TMS array.

Syntax

INPUT data, TMS (pram)

data is the identifier of the data set containing the values of
the dataset to receive the TMS array information

pram contains a set of TMS parameters formed exactly as described
in Subpart 3.10.3.

Comments

The keyword TMS may be optionally omitted.

3.25 The PUT Statement

The PUT statement is used to control the correspondence for TMS dataset values between working storage and off line storage.

Syntax

- A. To put off line values into working storage

PUT CORE (dataset list)

- B. To put working storage values into off line storage

PUT TMS (dataset list)

dataset list contains the datasets whose values are to be moved from one medium to the other.

Comments

The keyword CORE may be omitted.

3.26 The DROP Statement

The DROP statement allows the user to drop the space associated with nonfixed datasets from working storage; thus, making that space available for other purposes.

Syntax

- A. To drop some values

DROP(list)

list is a list of nonfixed datasets whose values are to be dropped.

- B. To drop all values

DROP*

IF

I-111

3.27 The IF Statement

It is often desirable to execute a certain section of a model only if some particular condition is met. The IF statement allows this operation.

Syntax

```
IF Boolean
{statements}
END
```

Boolean is a Boolean expression as described below.

Comments

The statements between the IF and END are executed only IF the Boolean is true.

3.27.1 Boolean Expressions and Variables

Examples

ENSIM LT 1970

ENSIM GT 1985

Boolean expressions are expressions formulated with one-dimensional data sets and constants and relational and Boolean operators. A Boolean expression may be either true or false. If it is true, it has the value 1. If it is false, it has the value 0. As such, Boolean expressions may be thought of as variables that may take the values 1 or 0.

The Boolean variable in the first example has the value 1, i.e., it is true, if the value of the dataset ENSIM is less than 1970. If the value of ENSIM is greater than or equal to 1970, then this Boolean variable is false and it has the value zero.

The Boolean expression in the second example is true, if the value of ENSIM is greater than 1985; otherwise, it is false.

The system allows the following six relational operators:

<u>Operator</u>	<u>Meaning</u>
LT	Less than
LE	Less than or equal to
GT	Greater than
GE	Greater than or equal to
NE	Not equal to
EQ	Equal to

Relational operators are used to formulate relational expressions between one dataset and another or one dataset and a constant.

In addition, the system allows the following three Boolean operators:

<u>Operator</u>	<u>Meaning</u>
AND	and
OR	or
NOT	not

Boolean operators may be used to link Boolean variables. Boolean expressions, therefore, are either single Boolean variables or strings of such variables linked with Boolean operators.

In constructing Boolean expressions you may link as many Boolean variables as you wish.

Also, Boolean expressions may be nested, the degree of nesting being unlimited, in principle. When formulating complicated nested Boolean expressions, it is advisable that you enclose in parentheses the component Boolean expressions being nested. As in arithmetic operations, nested parenthetical groups of Boolean operations are executed in the following order: the most deeply nested Boolean expression is executed first. As in every other statement, parentheses in Boolean expressions must be balanced.

Boolean expressions are used primarily in IF statements where they define the various alternative conditions that must be satisfied for execution to take alternative paths.

Boolean variables may be used in arithmetic expressions where they take only two values: 1 or 0. A Boolean variable has the value 1 if it is true, and it takes the value 0 if it is false.

3.28 The DO Statement

It is often desirable to perform repeatedly a given set of instructions until a given condition is met or while a given condition is met. The DO statement allows these operations.

Syntax

A. Do until a condition is met,

```
DO UNTIL Boolean  
{statements}  
END
```

B. Do while a condition is met,

```
DO WHILE Boolean  
{statements}  
END
```

Comments

See Subpart 3.27.1 for a discussion of Boolean expressions.

3.29 The SHOW Statement - with XYPLOT, SCATTER, or BAR

This statement permits the user to plot either the values of a time series dataset versus time, or the values of one vector dataset (the dependent variable or y-coordinate) versus the values of another vector dataset (the independent variable or x-coordinate). In the latter case, the x-coordinates for the points of the plot correspond to the values of the time index classifying the values of the time series dataset being depicted. XYPLOT displays a continuous line of asterisks (default) interpolated between values, whereas SCATTER displays only a single asterisk for each value SHOWN in the relationship. BAR displays a bar graph. Examples of each type of plot are given in 3.29.1.

Syntax

SHOW, { SCATTER XYPLOT BAR }	, data ,	options
SHOW, { SCATTER XYPLOT BAR }	, (data ₁ , data ₂),	options

data is the time-series dataset to be plotted versus time

data₁ is the independent values (i.e., X-axis variable); the minimum and maximum values of data₁ will appear as numeric captions underneath the display

data₂ is the dependent values (i.e., Y-axis variable); minimum and maximum values of data₂ will appear as numeric captions down the left side of the display

option is one of the optional parameters POINT, TITLE, XLABEL, XRANGE, YLABEL, YRANGE and SUBTITLE.

Comments

The first syntax must contain a dataset that is defined on a TIME index. The TIME series information (the independent variable) will be displayed for each TIME point specified in the TIME index.

In the second syntax, data₁ is the independent variable and data₂ the dependent variable. Data₁ and data₂ may be multiple-dimensional data sets defined providing they have at least one index that is common to both and that the extra indices are SET to the desired entries for the display.

3.29.1 The POINT Option

The POINT option allows the user to select the character used in the plot.

Syntax

POINT(string)

string is a single character to be used in the plots. Default
 is an asterisk.

3.29.2 The TITLE Option

The TITLE option allows the user to specify the title for the plot. The default title for non-TIME series datasets is blank (i.e., no title); for a TIME series dataset the default TITLE is the label of the dataset followed by the TIME range (e.g., 1970 to 1980).

Syntax

TITLE (list)

list consists of one or more of the following in any order

- cc** - carriage control characters (e.g., / to skip to the next line).
- string** - a string must be enclosed within a second set of parenthesis unless the entire list consists only of the string.
- id. L** - id is a dataset identifier; id. L will print the label of the dataset id.

3.29.3 The XLABEL Option

The XLABEL option allows the user to supply a descriptor for the X-axis; default is no descriptor.

Syntax

XLABEL (string)

string is a string of characters to appear below the X-axis.

3.29.4 The X RANGE Option

The X RANGE option is used only with XY PLOT and SCATTER. It is used to supply the minimum and maximum values for the X-axis.

Syntax

X RANGE (min, max)

min is the minimum value for the X-axis.
max is the maximum value for the X-axis.

Comments

The default value used for min is the actual minimum value of the independent variable. The default value used for max is the actual maximum value of the independent variable. If the actual values of the independent variable lie outside the (min, max) range supplied, then the resulting plot may be affected adversely.

3.29.5 The YLABEL Option

The YLABEL option allows the user to supply a descriptor for the Y-axis; default is no descriptor.

Syntax

YLABEL (string)

string is a string of characters to appear to the left of the Y-axis
 in a column.

3.29.6 The YRANGE Option

The YRANGE option is used to supply the minimum and maximum values for the Y-axis.

Syntax

YRANGE (min, max)

min is the minimum value for the Y-axis.
max is the maximum value for the Y-axis.

Comments

The default value used for min is the actual minimum value of the dependent variable. The default value used for max is the actual maximum value of the dependent variable. If the actual values of the dependent variable lie outside the (min, max) range supplied, then the resulting plot may be affected adversely.

3.29.7 The SUBTITLE OptionSyntax

SUBTITLE (string)

string is a string of characters to be positioned below the TITLE of the display.

Comments

The default SUBTITLE is the fixed index settings for multi-dimensional datasets; if singly-dimensional vectors are used, the default SUBTITLE is blank.

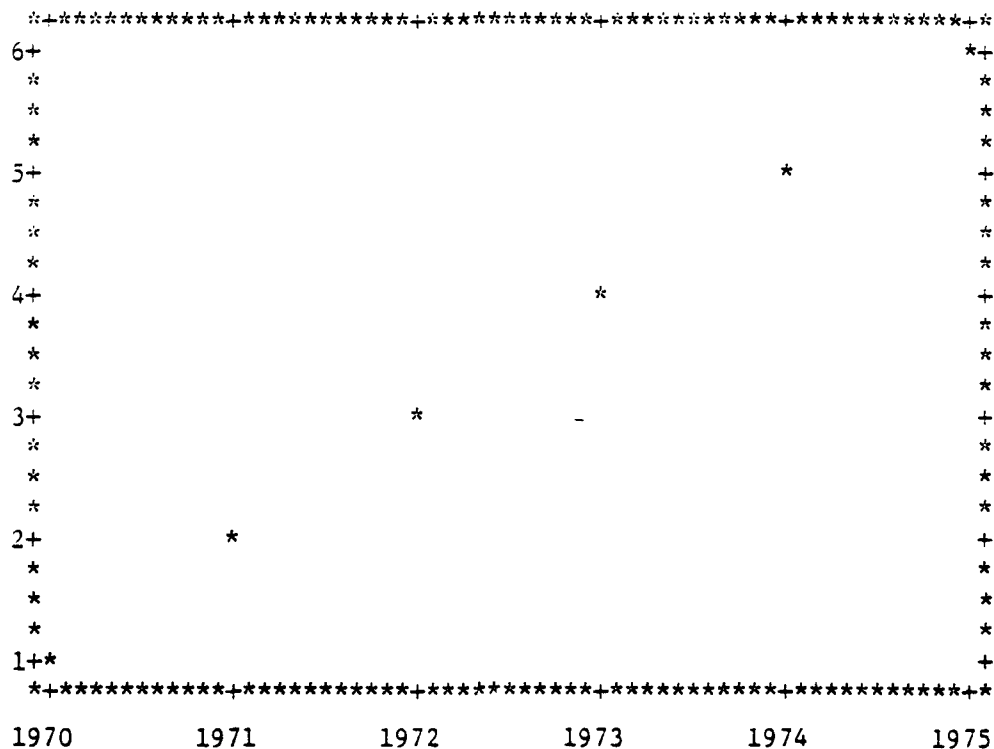
3.29.9 Examples of SHOW with SCATTER, XYPLOT and BAR

1. If A is a time series dataset defined yearly from 1970 to 1975 then
SHOW SCATTER A
produces the following plot

SHOW PLOTS

I-120

DATASET A, 1970 TO 1975



- If A and B are both time-series datasets defined yearly from 1970 to 1975 then

```
SHOW XYPLOT(A,B),XLABEL(DATASET A),YLABEL(DATASET B),
TITLE(DATASET A VERSUS DATASET B)
```

produces the following plot.

I-121

DATASET A

SHOW BAR A

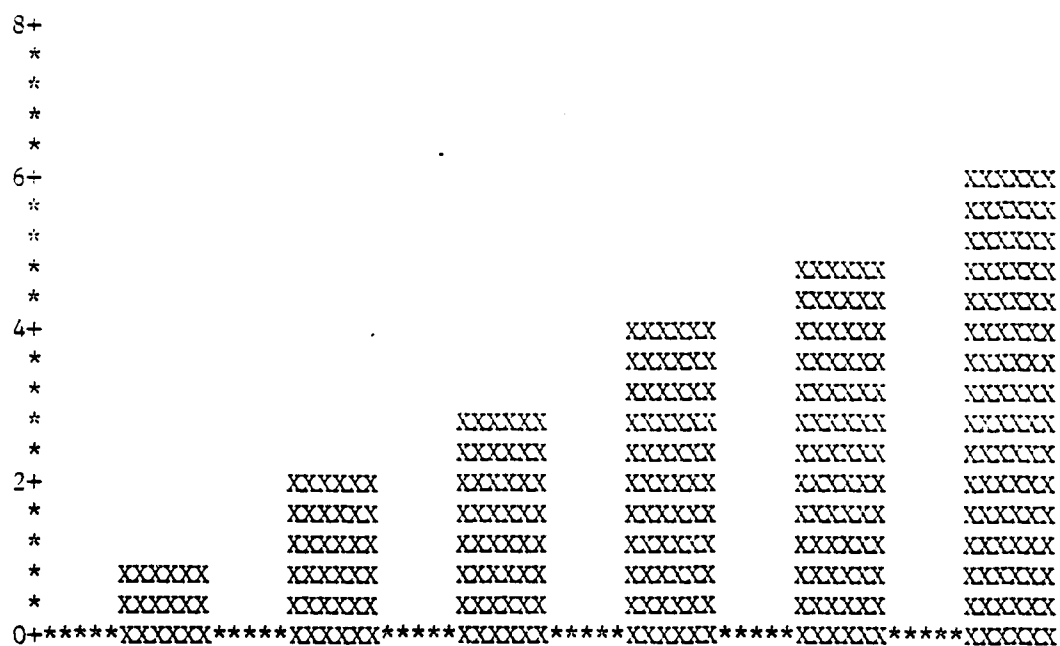
produces the following plot

C

SHOW PLOTS

I-122

DATASET A, 1970 TO 1975



3.30 The SHOW Statement - with TEXT

When used in conjunction with TEXT, the SHOW statement will generate a textual display that may be used for descriptive purposes, documentation etc. A combination of stored information and textual strings may be displayed.

Syntax

	<code>SHOW TEXT (list)</code>	left justifies "list"
or	<code>SHOW TEXT = CENTER (list)</code>	centers "list"
or	<code>SHOW TEXT = LEFT (list)</code>	left justifies "list"

TELL

I-124

3.31 The TELL Statement - with String

The TELL statement "tells" the user something, i.e., it issues a message, during execution.

Syntax

TELL(string)

string is a message of up to 125 characters.

3.32 The TELL Statement - with DATA

This statement allows the user to display a complete or partial listing of the datasets in a program. Each item in the listing contains the identifier and the label of the dataset listed.

Syntax

TELL DATA(data ₁ ,...data _i ,...data _n)

data_i is the identifier of the ith dataset to be listed. If no dataset identifiers are specified, then all datasets in the program segment currently in working space will be listed.

3.33 The ASK Statement - General

This statement "asks" the user a question and, depending upon his response, executes one of a number of alternative procedures.

Syntax

```
ASK(question), response1  
  procedure1  
  ELSE responsei  
    procedurei  
  END ASK
```

question is a question or statement eliciting a user response.

response₁ is a single, non blank work (no special characters),
 denoting a possible response to the question.

procedure₁ is the set of instruction(s) to be executed if the
 user enters "response₁".

ELSE response_i begins the *i*th alternative response/procedure
 branch of the ASK statement.

procedure_i is the set of instructions to be executed if the
 user enters "response_i".

Comments

The last response entry in the ASK may be blank in which case the procedure following that ELSE statement will be executed if any response, other than the previously specified ones, is given.

3.33.1 The ASK Statement - with ELSE DATA=indirect

This statement "asks" a question and allows the user to select a single dataset from a list of previously-defined datasets to be used in the following procedure.

Syntax

```
ASK(question), response1
ELSE DATA=indirect(data list1)
procedure1
.
.
.
ELSE DATA=indirect(data listn)
proceduren
END ASK
```

response ₁	passes control to END
ELSE DATA=	is used to specify a list of datasets
indirect	is a previously-defined indirect data set
data list ₁	is any list of previously-defined data sets, excluding indirects
procedure ₁	is the set of instructions to be performed on indirect (i.e., on the data set specified by the user from data list ₁)
data list _i	is any list of previously-defined data sets, excluding indirects
procedure _i	is the set of instructions to be performed on indirect in data list _i .

Comments

If data list_i is left blank any defined dataset may be substituted for the indirect.

3.33.2 The ASK Statement - with ELSE INDEX=index

This statement "asks" a question which allows the user to select a setting for an index. This is a conversational SET statement.

Syntax

```
ASK(question),response  
.  
.  
ELSE INDEX=index  
procedure  
END ASK
```

question	is the message asking the user to select a setting for an index
response	is the response to be entered if the user does not wish to modify the present index setting.
procedure	is the set of instructions to be executed after the index setting operation. These instructions are affected by this setting.
ELSE INDEX	puts an unconditional SET on an index, using user designated index entries. It nullifies and clears any previous SET commands on that index
index	is the identifier of a previously-defined index whose setting is being modified.

4.0 ERROR MESSAGES

If in formulating a statement the user does not conform to the syntax rules of the User Language, an error diagnostic will be issued immediately following the statement containing the error. In interactive or on-line mode, the user may correct the statement in error and continue. In batch mode, however, program execution will stop immediately after the error diagnostic, unless the system parameter DEBUG is on. If the system is in DEBUG mode, it will continue processing the program even if errors are encountered and error messages issued. With this option on, the user obtains a listing of all error messages in a single pass of the program through the system interpreter.

The system issues the following diagnostics:

- (1) ILLEGAL TYPE *n* TIE IN COLUMN *m* VALUE = *v*

n is an integer code taking the values 1, 2, 3, or 4 depending on the type of the TIE (Textual Input Element) in error. As discussed elsewhere (Part 2.2), there are four types of TIEs:

<u>TIE</u>	<u>Type</u>
1	special character
2	alphanumeric
3	integer
4	real number

m is the column number of the beginning character of the illegal TIE

v is the value of the illegal TIE

(2) ILLEGAL END OF INSTRUCTION

This diagnostic is issued whenever the preceding statement is not completed properly. This usually happens when left and right parentheses are not balanced properly. Also, it happens when the continuation of a long statement is not specified properly (with a comma or an operator) at the end of the preceding input line or card.

(3) MISSING STRING TERMINATOR

This diagnostic is issued whenever a descriptor exceeds 125 characters in length.

(4) DOUBLY DEFINED IDENTIFIER

This message is issued after an attempt to define an identifier that has previously been assigned to a structural element (index, dataset, model, graph, table, or program module). Identifiers must be unique within a structural element type. However, it is good practice to have unique identifiers across all structural elements of a program.

(5) UNRECOGNIZABLE COMMAND

This message is issued when the first TIE of the preceding statement is a Type 2 TIE but does not belong to the set of primary statement words listed in Table I-1 or the set of previously defined dataset, model and table identifiers.

(6) INDECIPHERABLE ENTRY

This message is issued when the first TIE of the preceding statement is not a Type 2 TIE. Note that all statements in a program must begin with a legal Type 2 TIE.

(7) ILLEGAL ENTRY DURING DATA READ

This message is issued whenever a dataset or graph value entered during a READ operation is in error. A dataset or graph entry is illegal if it is not numeric (integer or real).

(8) EMPTY DATA BASE

This message is issued when the file parameter of an OPEN statement is misspecified, i.e., is an unacceptable physical unit number.

(9) LOAD MODULE NOT DEFINED ON DATA BASE

This message follows a LOAD statement attempting to load from the database a nonexistent program module, i.e., a module not previously saved on the database.

(10) TABLE NOT DEFINED ON DATA BASE

This message is issued whenever reference is made to a TMS array number that has not been properly defined on a TMS database (using the ADD statement).

(11) DATA BASE SIZE EXCEEDED

This message is issued after an ADD (or a SAVE) statement attempting to save on a database a TMS array (or a program module) whose size exceeds the allocated storage capacity of the database. This usually happens because the database has not been properly opened using the OPEN statement.

(12) INSUFFICIENT SPACE FOR DATA VALUES

This message is issued whenever a storage allocation statement (a READ or an equation) requires more space than the allocated storage capacity of a program. The default storage capacity of a program is 10000 words.

(13) INSUFFICIENT SPACE TO PROCESS STATEMENT

This message is issued whenever a storage definition statement, such as a DATA statement defining a fixed dataset, requires more space than the allocated storage capacity of the program. The storage capacity of a program is equal to 10000 words (default value).

Examples of error messages are given in Figure I-1. The causes/ explanations of these sample error messages are given below:

READY

- 1 READY/STAART(ERROR MESSAGES)
UNRECOGNIZABLE COMMAND
READY/START(ERROR MESSAGES)
READY/INDEX ROW(3)
- 2 ILLEGAL END OF INSTRUCTION
READY/INDEX ROW(3),LABLE(ROWS)
- 3 ILLEGAL TYPE 2 TIE IN COLUMN 14 VALUE = LABLE
READY/INDEX ROW(3),LABEL(ROWS)
READY/INDEX COL(2)
- 4 ILLEGAL TYPE 1 TIE IN COLUMN 10 VALUE = /
READY/INDEX COL(2),44
- 5 ILLEGAL TYPE 3 TIE IN COLUMN 14 VALUE = 44
READY/INDEX COL(2.5)
- 6 ILLEGAL TYPE 4 TIE IN COLUMN 11 VALUE = 2.5
READY/INDEX COL(2)
READY/INDEX LONG(4),LABEL(THIS IS A CHARACTER STRING WHICH IS MORE THAN
/125 CHARACTERS LONG AND WHICH THEREFORE CANNOT BE ACCEPTED BY
- 7 MISSING STRING TERMINATOR
READY/INDEX ROW(4)
- 8 DOUBLY DEFINED IDENTIFIER
READY/+DATA A
- 9 INDECIPHERABLE ENTRY
READY/DATA A(ROW),(A DATASET)
READY/READ A
/1 2 X
- 10 ILLEGAL ENTRY DURING DATA READ
/1 2 4
READY/DATA A(COL)
- 11 DOUBLY DEFINED IDENTIFIER
READY/OPEN(12,23)
- 12 EMPTY DATA BASE
READY/OPEN(1,23)
READY/SYSTEM LOAD=1
READY/LOAD PROG
- 13 LOAD MODULE NOT DEFINED ON DATA BASE
READY/INPUT A,TMS(1,2)
- 14 TABLE NOT DEFINED ON DATA BASE
READY/ADD(1,2),SIZE(100,10),

FIGURE I-1. SAMPLE ERROR MESSAGES

<u>Error</u>	<u>Description</u>
1	STAART is not a recognizable command word: START is.
2	The right parenthesis required in the index size definition is missing.
3	TABLE is not an acceptable option for the index statement; LABEL is.
4	To define the size of index COL a left parenthesis must be used, not a /.
5	The integer 44 is a spurious TIE of Type 3.
6	The size of an index must be specified as a nonnegative integer. The real number 2.5 is inappropriate as an index size.
7	This index label is too long.
8	The index ROW is already defined.
9	Every program statement must start with a legal Type 2 TIE, except in continuation statements. The + in Column 1 is unacceptable.
10	The X is not an acceptable value for the dataset A. The number 4 is.
11	The identifier A has already been assigned to another dataset.
12	The physical unit number 12 is too large for the operating system processing this program. An integer less than 10 is always acceptable.
13	The program module PROG has not been previously saved (using the SAVE statement).
14	Table or TMS array 2 has not been defined previously on the database (using the ADD statement).

5.0 THE FORTRAN INTERFACE

A major advantage of the present approach is that most of the information manipulated by programs written in the user language can be stored on or retrieved from off-line files automatically. The values of any dataset defined with a TMS parameter are saved and can be referenced later by other FORTRAN programs. As a result of this fact, if the user wishes to interface other FORTRAN programs with the system, he can do so simply by manipulating the same TMS arrays as are manipulated by the system.

The only software link to a TMS database is a generalized FORTRAN subroutine which has the same capabilities as the TMS option on datasets. This subroutine may be called directly from FORTRAN programs. This subroutine, SELNDT, is described in this chapter. This discussion assumes familiarity with the TMS concepts and with FORTRAN.

5.1 When to Use SELNDT

Subroutine SELNDT can be used whenever one needs to read values from or write values to TMS arrays in existing TMS databases via a FORTRAN program external to the system. SELNDT assumes that the file already exists and that all TMS arrays to be referenced have already been added to the database. These operations can most easily be performed by the host system itself; therefore, SELNDT has not been given these capabilities.

For those computer systems which require that an "open database" operation be performed before any direct access can be executed by a FORTRAN program, the "open database" statement must be placed into the FORTRAN program by the user. Generally, this statement is a DEFINE. Users who are uncertain about these conventions on their particular systems should consult with their computer staff.

5.2 Parameters for SELNDT

Subroutine SELNDT has the following ten (10) calling parameters or arguments:

- IOP - Specifies if read (1) or write (0) is desired
- FILE - Contains the physical unit number of the file
- TABLE - Contains the number of the TMS array
- THR - Is a scratch storage array used by SELNDT
- DAT - Contains or returns the data values
- BASE - Defines the base point
- DIM - Defines the dimensions of DAT
- ENT - Defines the number of entries in DAT
- TRANS - Defines the transposition to be performed
- NDIM - Contains number of data dimensions.

This part discusses these parameters in detail.

5.2.1 The Parameter IOP

The parameter IOP is an integer variable. It specifies whether values are to be read from the TMS array into a core resident array or whether core resident values are to be written to the TMS array. A value of zero (0) indicates write, while a value of one (1) indicates read.

5.2.2 The Parameters FILE and TABLE

The parameters FILE and TABLE are both integer variables. They specify the physical unit number of the file containing the TMS array and the table number of the array, respectively. Any file referenced by the FILE parameter has to have been previously created by the host system. A table on that file referenced by the TABLE parameter has to have been previously added to the file by the host system.

5.2.3 The Parameter THR

The parameter THR is a one-dimensional integer array needed by SELNDT for scratch storage. The user must allocate at least $4 + ND + NV$ words to this array in his calling program. ND represents the maximum number of array dimensions associated with any TMS array to be manipulated. NV represents the maximum number of virtual sheets associated with any TMS array to be manipulated. Usually, $ND=3$ and $NV=0$; thus, the dimension of THR is 7.

From the standpoint of the calling routine only the first cell of the THR is important. If this cell returns a nonzero value, then the access was successful. If the cell returns zero, then the access was not successful.

5.2.4 The Parameter DAT

The parameter DAT is a real array which contains the core resident values being read or written. Its dimensionality is defined by the DIM and NDIM parameters in the calling sequence.

5.2.5 The Parameters DIM and NDIM

The parameters DIM and NDIM are used to define the dimensionality of the parameter DAT. NDIM is an integer variable which contains the number of dimensions in DAT. DIM is a one-dimensional integer array which contains NDIM entries. The values in DIM should be identical both in magnitude and order to the values in the DIMENSION declaration for DAT. Thus, if DAT is dimensioned

DAT(10,20,15)

DIM would be initialized as follows

5.2.6 The Parameter ENT

The parameter ENT is a one-dimensional integer array containing NDIM entries. It is used to define the actual number of entries to be processed for each dimension in the DAT array.

5.2.7 The Parameter BASE

The parameter BASE is a one-dimensional integer array containing an entry for each dimension in the TMS array. This array simply defines the base point of the core storage array relative to the TMS array.

5.2.8 The Parameter TRANS

The parameter TRANS is a one-dimensional integer array containing an entry for each dimension in the TMS array. This array defines the transposition relationship between the core storage array and the TMS array. If TRANS (I) is zero, then the Ith dimension on the TMS array is invariant relative to the core storage array -- i.e., it has the value specified for the base point. If TRANS (I) equals J, then the Jth core storage array dimension moves along the Ith TMS array dimension.

Note that if word one of TRANS simply contains a minus one, then no transposition is performed.

APPENDIX D. USER'S GUIDE

The General Aviation Dynamics Computer System

Users Guide

The General Aviation Dynamics model (GAD) is implemented in NUCLEUS*, a computer software system developed at Battelle. It resides currently on the Battelle computer system and on the United Computing Systems (UCS) computer, and it can be accessed remotely on either machine via telephone from anywhere in the U.S. Access is possible via either a remote batch terminal or an interactive terminal, providing that an authorized user name and password for the particular computer system being accessed are available. Procedures for accessing the GAD model on the Battelle and UCS computers are given in Sections I and II; Section III shows an example of an interactive session with GAD and Section IV describes batch use of the model.

Section I: Accessing on the Battelle computer

The GAD model resides on the Battelle computer and may be accessed via telephone in either remote batch mode using a remote batch terminal that is appropriately linked to the Battelle computer, or in on-line or interactive mode using an interactive terminal. Details of the actual access procedure for interactive use follow.

* NUCLEUS: NUmerical CLassification and EvalUation System

I.1 Login/Logout Procedures for the Battelle Computer System
via TYMNET

- (1) Dial TYMNET number from the list in Table 1.
Wait for data tone from TYMNET connection.
- (2) Place telephone headset in acoustic coupler, or initialize other type of modem.
- (3) The message
PLEASE ENTER TERMINAL IDENTIFIER
is printed or displayed on the terminal. On some terminals this message will appear garbled. If this occurs simply continue with Step (4).
- (4) Type the appropriate identifier for your terminal from the list in Table 2.
- (5) TYMNET responds:
PLEASE LOGIN:
- (6) Login by typing the sequence
(CTRL H)BCL6400;INTERCOM(CR)
where
(CTRL H) denotes simultaneous keying of the control and H keys, and (CR) denotes keying of the carriage return (or cursor return) key.
No blanks (spaces) are allowed in the above sequence. Wait a few seconds (about 10) for connection with the Battelle INTERCOM system. The TYMNET username/password combination (BCL6400;INTERCOM) should be treated with discretion and made available only to users with valid Battelle username/password combinations.
- (7) INTERCOM responds:
BATTELLE INTERCOM 4.5
DATE MM/DD/YY
TIME HH.MM.SS
PLEASE LOGIN
- (8) Enter the following sequence:
LOGIN,username,password,SUP(CR)
where username and password are valid code words issued by the Battelle Computer Center. No blanks (spaces) are allowed in the above sequence.
- (9) INTERCOM responds:
COMMAND-
- (10) Carry out dialogue with INTERCOM (see Login/Logout Procedures via Direct Dial-Up, Step 6).
- (11) To exit from INTERCOM normally, enter
LOGOUT
after the INTERCOM prompt COMMAND-

- (12) To exit from INTERCOM abnormally (i.e., abort)
 enter the sequence
 % A
 after a system pause. Or, if in the middle of a
 lengthy response, enter the sequence
 (CRTL Z) % A
 where
 (CRTL Z) denotes the simultaneous keying
 of the control and Z keys. This keying
 causes the system response to halt.
 After an abnormal termination, the system responds
 USER ABORT
 COMMAND-
 At this point, you may log out normally.

- (13) After LOGOUT, the system gives you an itemized
 summary of your interaction with the Battelle
 Computer. This summary looks like the following
 CP X.XXX SEC.
 IO Y.YYY SEC.
 CM Z.ZZZ SEC
 SS S.SSS SEC
 CH CCC CHARS.
 CONNECT TIME H HRS. M MIN. MM/DD/YY
 LOGGED OUT AT HH.MM.SS.
 where
 CP is the central processor time of your run
 IO is a measure of your use of peripheral equipment
 CM is a measure of your use of central memory
 SS is the combined amount of computer time for which
 you are charged
 CH is the number of characters transmitted
 CONNECT TIME is the time between login and logout.

The cost for TYMNET services is approximately 17¢ per connect minute which
 is in addition to the normal INTERCOM processing cost incurred. These
 charges are subject to change.

- NOTES:
- To backspace, type CTRL and H simultaneously
 - To skip a line of entry, type CTRL and X simultaneously
 - To suspend printing, type CTRL and Z simultaneously
 - To abort, enter %, then A
 - All user entries must be terminated by (CR).

The following TYMNET messages may occur if a TYMNET connection is not immediately available.

<u>Message</u>	<u>Meaning</u>
ALL PORTS BUSY OR HOST OUT OF PORT	A connection with the host computer cannot be made at this time because the TYMNET ports on that host are all busy. Try again in a few minutes.
HOST NOT AVAILABLE THROUGH NET	This message can occur if: 1) the TYMCOM or its neighbor(s) is down, 2) an invalid host has been requested (perhaps mistyped), or 3) a new Supervisor is taking over the network and has not yet picked up that host. Try again in a few minutes.
HOST DOWN OR HOST "n" DOWN	The network is fully operational but the host computer itself is down.
HOST SHUT	The Supervisor has been notified to route no new users to the host: current users are not affected.
DROPPED BY HOST SYSTEM	The user has logged off and/or has been disconnected by the host computer. If the session is finished, hang up; if not, try again.
CIRCUITS BUSY	All available paths to the TYMCOM are busy; try again in a few minutes.
TRY AGAIN IN 5 MINUTES	A new Supervisor is taking over the network and a log-in will normally be possible in a few minutes.

TABLE 1 (Continued)

[illegible]

Table I (Continued)

THE FOLLOWING LOCATIONS NOW HAVE 1200 BAUD SERVICE AVAILABLE
UPDATE AS OF 01/05/78

NAME	LOCATION	PHONE NO.	MODEM TYPE*
SF2	SAN FRANCISCO, CA	415/421-7121	1200 VADIC
SF2	SAN FRANCISCO, CA	415/982-3770	1200 BELL 212
SF2	SAN FRANCISCO, CA	415/397-8461	1200 BELL 202S
HOU3	HOUSTON, TX	713/780-7596	1200 VADIC
HOU3	HOUSTON, TX	713/780-7593	1200 BELL 202S
DAL1	DALLAS, TX	214/688-1117	1200 BELL 212
LA2	LOS ANGELES, CA	213/687-8083	1200 VADIC
LA2	LOS ANGELES, CA	213/629-3001	1200 BELL 202S
LA2	LOS ANGELES, CA	213/626-0365	1200 BELL 212
CRP3	SAN JOSE, CA	408/446-6932	1200 BELL 202S
CRP3	SAN JOSE, CA	408/446-7001	1200 VADIC
DET3	DETROIT, MI	313/963-2353	1200 VADIC
CRP4	SAN JOSE, CA	408/446-7309	1200 BELL 212
DET3	DETROIT, MI	313/963-4676	1200 BELL 202S
DET3	DETROIT, MI	313/963-2353	1200 VADIC
WAS3	WASHINGTON, DC	703/841-9330	1200 VADIC
WAS3	WASHINGTON, DC	703/525-6290	1200 BELL 212
WAS3	WASHINGTON, DC	703/527-7106	1200 BELL 202S
SL1	ST LOUIS, MO	314/621-4660	1200 BELL 212
NY3	NEW YORK, NY	212/689-8910	1200 VADIC
NY3	NEW YORK, NY	212/689-8850	1200 BELL 202S
BOS3	BOSTON, MA	617/965-5520	1200 VADIC
BOS3	BOSTON, MA	617/244-1240	1200 BELL 202S
CHI3	CHICAGO, IL	312/372-0391	1200 VADIC
CHI3	CHICAGO, IL	312/368-0022	1200 BELL 202S
ATLSR	ATLANTA, GA	404/581-0619	1200 BELL 212
SCOSR1	HOUSTON, TX	713/977-7671	1200 BELL 212

*--WHEN USING 1200 BAUD SERVICE THE USER MUST SELECT A PHONE
NUMBER EQUIPPED WITH THE SAME TYPE OF MODEM AS THE MODEM
CONNECTED TO THE USER'S TERMINAL.

TABLE 2. USER TERMINAL IDENTIFICATION CODES

<u>Identifying Character</u>	<u>Terminal</u>	<u>Speed In/Speed Out</u>
A	All CRT	30/30 or 120/120
C	Beta, DCT500	30/30
E	All Thermal	30/30
G	Memorex, GE Terminet, Data-speed 40 with printer	30/30 or 120/120
B	Mod 37, CRT - odd parity	15/15 (odd parity)
F	Beta	15/30
J	All CRT - Even parity	15/15 (even parity)
N	All Thermal	15/30
CR (Carriage Return)	2741 and other Selectric based	14.8/14.8
D	TTY, CRT	10/10

Type (CTRL H) after identifying character, but before username to denote a half duplex terminal.

Type (CTRL F) to denote full duplex terminal.

I.2 Login/Logout Procedures for the Battelle Computer System
and the NUCLEUS System
via

Direct Dial-Up

- (1) Dial (614)424-5850. Telephone will ring; a high-pitched tone follows.
 - (2) Place receiver in coupler. The green carrier light will come on. Terminal is connected to computer.
 - (3) Key CR (carriage return) to indicate the terminal's baud (300 or 110) to the Battelle CDC equipment.
 - (4) A message like the following is displayed or printed on the terminal:
BATTELLE INTERCOM 4.5
DATE MM/DD/YY
TIME HH.MM.SS.

PLEASE LOGIN
 - (5) Enter the following sequence:
LOGIN,username,password,SUP.(CR)
where username and password are valid code-words issued by the Battelle Computer Center. No blanks (spaces) are allowed in the above sequence.
- NOTE: To correct mistakes in a given line of entry, backspace to the character in error by pressing the CTRL and H keys simultaneously as many times as necessary. Then retype the remainder of the line.
- NOTE: To delete line of entry altogether, press the CTRL and X keys simultaneously. Then continue entering correct information
- NOTE: (CR) denotes carriage return (or cursor return).

- (6) The Battelle INTERCOM system responds:

COMMAND-

To extend the time allowed for the on-line interaction,
enter

ETL, 100

This ensures enough time for the entire interactive dialogue, calculations, and desired output. The system will again respond:

COMMAND

- (7) To attach the GAD procedure file enter
ATTACH,PROFIL,AVIATION,ID=NUCLEUS,MR=1.
- (8) At the system response, COMMAND, begin execution of the GAD procedure by entering
BEGIN,GAD

The NUCLEUS system will respond

READY/

To execute the dynamic model enter

LOAD GAD

- (10) To terminate abnormally at any time the interaction with NUCLEUS:
 - If at a system pause, enter %, then A, then (CR).
 - If in the middle of a lengthy NUCLEUS, response, enter CTRL and Z simultaneously, then %, then A, then (CR).
- (11) INTERCOM responds:
 - USER ABORT
 - COMMAND-
- (12) To exit from INTERCOM, enter
LOGOUT
any time you have the prompt COMMAND
- (13) After an exit from NUCLEUS, you are back in INTERCOM with the prompt
COMMAND
you may enter the dynamic simulation model again or end your interaction with the system
- (14) To exit from INTERCOM, enter
LOGOUT
- (15) After LOGOUT, don't forget to hang up the phone.

Examples of login/logout procedures are given in Table 3.

TABLE 3. LOGIN/LOGOUT PROCEDURES - EXAMPLES

All user entries are underlined. It is assumed that you have already dialed in, either direct or via TYMNET.

(1)	(2)	(3)
Login/logout for INTERCOM. Long format. User password is overwritten. BATTELLE INTERCOM 4.7 DATE 02/06/79 TIME 12.42.17. PLEASE LOGIN LOGIN AVIATION XXXXXXXXX ENTER PASSWORD- 02/06/79 LOGGED IN AT 12.42.42. WITH USER-ID SE EQUIP/PORT 01/077 COMMAND- LOGOUT CP .201 SEC. IO .373 SEC. CM .038 SEC. SS 1.000 SEC. CH 165 CHARS. CONNECT TIME 0 HRS. 0 MIN. 02/06/79 LOGGED OUT AT 12.42.51.	Login/logout for INTERCOM. Short format. User password is shown explicitly. BATTELLE INTERCOM 4.7 DATE 02/06/79 TIME 12.42.44. PLEASE LOGIN LOGIN AVIATION DEMO SUP COMMAND- LOGOUT CP .148 SEC. IO .368 SEC. CM .034 SEC. SS 1.000 SEC. CH 0 CHARS. CONNECT TIME 0 HRS. 0 MIN. 02/06/79 LOGGED OUT AT 12.44.12.	Login/logout for INTERCOM. Long format with error in username/password combination. BATTELLE INTERCOM 4.7 DATE 02/06/79 TIME 12.44.49. PLEASE LOGIN AVIATION FORMAT ERROR READY..LOGIN AVIATION XXXXXXXXX ENTER PASSWORD- INVALID USER NAME OR PASSWORD ENTER USER NAME-AVIATION XXXXXXXXXX ENTER PASSWORD- 02/06/79 LOGGED IN AT 12.45.28. WITH USER-ID SE EQUIP/PORT 01/366 COMMAND- LOGOUT CP .302 SEC. IO .675 SEC. CM .054 SEC. SS 1.000 SEC. CH 245 CHARS. CONNECT TIME 0 HRS. 0 MIN. 02/06/79 LOGGED OUT AT 12.45.36.

AD-A085 007

BATTELLE COLUMBUS LABS ON P/S 1/3
THE GENERAL AVIATION DYNAMICS MODEL VOLUME III. SYSTEMS MANUAL.(U)
JUL 79 M A DUFFY, J H MCCREERY DOT-FA77WA-0003

UNCLASSIFIED

FAA-APP-79-0-VOL-3

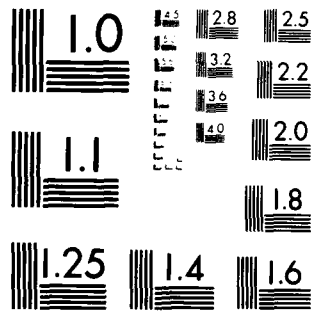
NL

4 of 4

AL
JAN 80 07



END
DATE
FILMED
7-80
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Section II: Accessing GAD on the UCS Computer

The GAD model also resides on the UCS computer and may be accessed via telephone in either remote batch mode using a remote batch terminal that is appropriately linked to UCS, or in on-line mode using an on-line or interactive terminal. Details of the actual access procedure for interactive use follow.

II.1 Logging into the System

Your UCS representative must furnish to you the local telephone number which connects terminals in your city with the UCS National Data Center in Kansas City, Mo. He will also furnish you with a unique seven-character user number with which you need to identify yourself to the computer each time you log-in. This user number is assigned to you for security and for billing purposes.

Log-in by following this sequence:

- If your terminal is equipped with a HALF/FULL duplex switch, set it to HALF.
- Be sure that the coupler is turned on if it has controls separate from those of your terminal.
- Turn on the teletypewriter if the controls are separate from the coupler.
- Dial the special phone number and wait until you hear a high-pitched tone like a whistle. This means that the computer has a free line which it has just dedicated to you.
- Place the telephone receiver in the acoustical coupler of your terminal.
- As soon as the connection is made between your terminal and the central site computer, you will notice a signal. Depending on the terminal speed and type, your terminal may give a small "jump", a button will light up or some output will be printed. For example, the 10 cps teletype will print L?
- Your response will depend on the speed of your terminal and on the system code you have been instructed to use.

Line Speeds

10
13.4
15
30
120

Log-in Codes

?
3
8
T
P

If you have any questions concerning the appropriate response for your particular type of terminal, consult your UCS representative.

System Codes

61
62
63
65

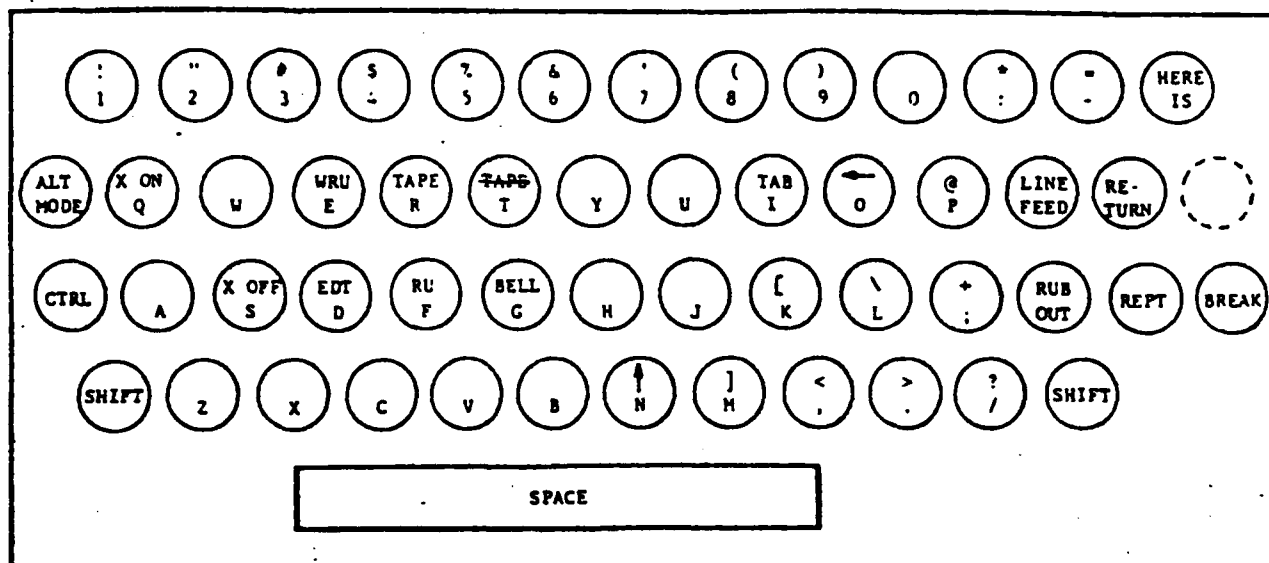
- The operating system will print some heading information consisting of date, time and communication information and then request your user number. Type it in and return the carriage.
- The system will then print an overscored area in which you must enter your password. Actually type the password over the blackened letters; this prevents anyone from seeing your password if you leave your terminal. If you are not interested in the security of the overprinting, you may enter the password after your user number on the same line. Simply type a comma (or a blank) and the password directly after the user number.
- If your user number and password are accepted by the system, a message prints and you are ready to begin. If either your user number or password is invalid, the message ILLEGAL LOGIN, TRY AGAIN prints, and you must enter both again. The system allows you five tries to enter the proper combination and then will print ILLEGAL TERMINAL and disconnect you.

L? Enter login code
UNINET NODE 12 CHNL 40331

SERVICE : Enter system code

UCS 02/06/79. 10.22.10. Q113
USER NUMBER: Enter user number
~~XXXXXXXX~~ Enter password
PROJECT-ID REQUIRED: Enter project-id
RDY-FOR

Teletypewriter Keyboard for UCS



Notes:

- Teletypewriters having different keyboards than above will usually perform the same functions if the key positioned in the same place is used, e.g. for + use Shift O.
- To delete one character, type +.
- To delete a line, type ALTMODE.
- To abort while the terminal is printing, enter S.
- To abort while the terminal is not printing, enter STOP.
- All user entries must be terminated by RETURN.

II.2 Accessing GAD interactively on the UCS computer

- (1) After the log-in procedure the system will respond

RDY-FOR

Execute the GAD command file by entering

CMD, GAD

- (2) The system will respond

01/25/79. 15.20.23.

PROGRAM NUCLEUS

READY ?

Execute the GAD model by entering

LOAD, GAD

II.3 Log-Off Command

When a user is finished working at the terminal, he should disconnect from the system as soon as possible because connect time is a billable resource. This is done by entering GOODBYE (or BYE).

The GOODBYE Command

The GOODBYE command, or the abbreviation BYE, disconnects the terminal from the time-sharing system and prints the total connect time for the session plus applicable service unit statistics for your billing option.

Command format: GOODBYE

or

BYE

Section III. AN EXAMPLE OF THE GENERAL AVIATION DYNAMICS
MODEL IN THE INTERACTIVE MODE

Sensitivity Analysis

In general, there are two ways to use model results or simulations - individually as projections and in pairs as sensitivity measures. Use of the model simply to make projections is fraught with dangers. Many potential users will not understand how the projections were derived and will expect unreasonable accuracy. The model is better used by employing extensive sensitivity analysis to evaluate a range of policies under a range of exogenous conditions. This process will identify the principal areas of model uncertainty and those portions of the model that deserve the greatest additional research.

The logical structure of the GAD model has been constructed such that relative comparisons can be made between the model forecasts from any two simulations. In particular, during a sensitivity analysis, absolute forecasts for each simulation are available, as well as percent deviations between the two cases. These deviations can be displayed over time either graphically or in tabular format.

A sensitivity analysis can be performed between any two simulations which are compatible with the model's capabilities. All GAD model output data from the first simulation are stored on a separate file. This base case need not be the "baseline" forecast representative of expected future conditions, but can be the result of any consistent set of conditions chosen by the analyst. Intermediate absolute forecast results from this base case can be obtained by the analyst, if desired. After obtaining all required intermediate output, the second simulation is specified and run. Absolute results of the second simulation are also available to the analyst. Sensitivity results are derived within the program logic by subtracting the results of the first simulation from the second simulation, dividing by the first simulation, and multiplying by 100 to convert differences to percent deviations from the base case; mathematically,

$$\% \text{ Deviation} = \frac{AA(I,J)_2 - AA(I,J)_1}{AA(I,J)_1} \times 100$$

where,

$AA(I,J)_1$ = the number of active aircraft of type J within category I from the first (base) simulation

$AA(I,J)_2$ = the number of active aircraft of type J within category I from the second simulation

Values for these parameters are, of course, obtained at the same instant in time during their respective simulations.

Should conditions within the second simulation not change immediately from the base case, percent deviations, until the change becomes effective, will be zero. Furthermore, by continually computing these deviations over time, the non-linearity in model response is preserved. Most previous sensitivity analyses of general aviation activity were predicated on either linear or log-linear sensitivities.

An Example

The GAD model uses the interactive dialogue feature of NUCLEUS to guide the analyst through a series of procedures and options. This technique eliminates the need for preliminary calculations by the user. Simple yes/no responses to NUCLEUS questions establish the conditions of the particular simulation to be run. If the user is uncertain of the parameter values contained in the model, NUCLEUS will display them. If the user desires to change these values, NUCLEUS will accept the new values. Incorrect (or unexpected) responses to NUCLEUS questions will simply cause the same question to be repeated.

A sensitivity example, comparing the normal "baseline" forecast to an increased fuel tax (effective January 1, 1979), is discussed below. This example was run on the UCS computer. Not all the options available for input/output are displayed; only enough to illustrate the procedures. In this example, all user entries are underlined.

Following the usual log-in procedures and LOAD GAD, the computer will respond

YOU ARE ENTERING THE GENERAL AVIATION DYNAMICS MODEL CREATED AT BATTELLE COLUMBUS LABORATORIES, WRITTEN IN THE MODELING LANGUAGE NUCLEUS. IN THIS SESSION YOU WILL PROJECT CERTAIN LEVELS OF GENERAL AVIATION ACTIVITY FOR THE YEARS 1977 TO 1987.

ENTER ENDING YEAR FOR SIMULATION

?

Any year between 1977 and 1987 is an acceptable response; the simulation will be from 1977 through the year specified and the results will be reported for that range of years. A response within the acceptable range

1980

allows the system to continue with

Step 1 -- WOULD LIKE TO COMPUTE THE FORECAST WITH THE INITIAL ASSUMPTIONS UNCHANGED (YES OR NO) OR VIEW THE STEPS OF THIS MODEL (TEACH)

?

The response

TEACH

causes the steps of the model to be printed and then the question to be repeated. "Teach" also causes the steps of the model to be printed out as those steps are executed

- STEP 1 -- COMPUTE THE FORECAST USING THE INITIAL ASSUMPTIONS UNCHANGED.
- STEP 2 -- DISPLAY AND/OR CHANGE INITIAL ASSUMPTIONS.
- STEP 3 -- COMPUTE THE FORECAST OF GENERAL AVIATION ACTIVITY.
- STEP 4 -- PRINT TABLES OF RESULTS OF THE FORECAST.
- STEP 5 -- PLOT THE RESULTS OF THE FORECAST.
- STEP 6 -- COMPARE THE RESULTS OF THE PRESENT FORECAST TO THOSE OF A PREVIOUS FORECAST FOR SENSITIVITY ANALYSIS.
- STEP 7 -- PRINT TABLES FOR SENSITIVITY ANALYSIS.
- STEP 8 -- PLOT THE RESULTS OF SENSITIVITY ANALYSIS.
- STEP 9 -- SAVE THE RESULTS OF THIS FORECAST FOR FUTURE SENSITIVITY ANALYSIS.

STEP 1 -- WOULD YOU LIKE TO COMPUTE THE FORECAST WITH THE INITIAL ASSUMPTIONS UNCHANGED (YES OR NO) OR VIEW THE STEPS OF THIS MODEL (TEACH)

?

Now the appropriate response to the question is either YES or NO; a response of

YES

causes the normal "baseline" simulation to be executed. Since the TEACH flag was set on by the TEACH request, the steps are printed out as they are executed

STEP 3 -- THE FORECAST OF GENERAL AVIATION ACTIVITY IS BEING COMPUTED.

Having executed the simulation the system prints,

STEP 4 -- PRINT TABLES OF RESULTS OF THE FORECAST.

and then asks the question

DO YOU WANT TO SEE TABLES OF RESULTS OF THE FORECAST
?

The response

YES

causes the system to ask

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
?

The unfamiliar user will not know the available tabular output options. By responding

LIST

the following list of output table options will be printed.

IDENTIFIER	DESCRIPTION
AA1	ACTIVE AIRCRAFT BY YEAR
AA2	ACTIVE AIRCRAFT BY USER CATEGORY
AIRPORTS	LOCAL AND ITINERANT OPERATIONS PLUS IFR FLIGHT PLANS FILED
AIRUTIL	AIRCRAFT UTILIZATION RATES
ECONOMIC	DPI,GNP,RAD
FIXEDCOST	FIXED COST
VARCOST	VARIABLE COST
FUEL	FUEL CONSUMED IN MILLIONS OF GALLONS
HOURSFLOWN	HOURS FLOWN IN THOUSANDS
OPERATIONS	TOTAL OPERATIONS, IN THOUSANDS
PILOTS	SP,PP,CP,ATP,P,HP,TP,IP,HR,THP
REVENUE	FEDERAL TAX REVENUE
TOTALS	TOTAL AIRCRAFT, TOTAL HOURS FLOWN, TOTAL OPERATIONS

When the list is complete, the previous question will be repeated:

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
?

A response of

PILOTS

will generate the following table

GENERAL AVIATION DYNAMICS MODEL PAGE 1

PILOT DATA, 1977 TO 1980

	1977	1978	1979	1980
STUDENT PILOTS	188,801	183,794	183,654	183,176
PRIVATE PILOTS	309,005	323,821	335,104	344,608
COMMERCIAL PILOTS	187,801	189,699	192,068	195,342
AIR TRANSPORT PILOTS	45,072	47,784	50,879	53,766
PILOT SUBTOTAL	730,679	745,098	761,705	776,891
HELICOPTER PILOTS	4,804	4,333	3,940	3,608
TOTAL PILOTS	735,483	749,431	765,645	780,499
INSTRUMENT RATINGS	211,364	221,497	232,437	243,969
HELICOPTER RATINGS	23,012	24,395	25,739	27,052
TOTAL HELIC RATINGS	27,816	28,728	29,679	30,659

Note that the table is printed for only the requested years, 1977-1980. Upon completion of the requested output table, the same question is repeated.

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
?

The user may request as many of the table options as he wants. When no more tabular data is required the response is

NONE

Since the TEACH flag is on, the computer prints the next step in the model,

STEP 5 -- PLOT THE RESULTS OF THE FORECAST.

and then asks the question

DO YOU WANT TO SEE PLOTS OF RESULTS OF THE FORECAST
?

By answering

YES

the computer responds

WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
?

Not being familiar with the plot options the user responds

LIST

which will generate the following list of variables:

IDENTIFIER	DESCRIPTION
AA	NUMBER OF ACTIVE AIRCRAFT
AASUM	TOTAL NUMBER OF AIRCRAFT
ATP	AIRLINE TRANSPORT PILOTS
AUR	AIRCRAFT UTILIZATION RATE (HRS/AC/YR)
CP	COMMERCIAL PILOTS
DPI	DISPOSABLE PERSONAL INCOME (1972 \$, 1972-1)
FC	FUEL CONSUMED (MILLION GALLONS)
FIX	FIXED COST INDEX (\$/HR), (1972 \$, 1972-1)
FTR	FEDERAL TAX REVENUE (MILLION DOLLARS)
GNP	GROSS NATIONAL PRODUCT (1972 \$, 1972-1)
HF	HOURS FLOWN (THOUSANDS)
HFSUM	TOTAL HOURS FLOWN (THOUSANDS)
HP	HELICOPTER PILOTS
HR	HELICOPTER RATINGS
IP	INSTRUMENT RATINGS
OPS	OPERATIONS (THOUSANDS)
OPSUM	TOTAL OPERATIONS (THOUSANDS)
P	PILOT SUBTOTAL
PP	PRIVATE PILOTS
RAD	REVENUE AIRCRAFT DEPARTURES (1972 \$, 1972-1)
SP	STUDENT PILOTS
TC	TOTAL COST
THP	TOTAL HELICOPTER RATINGS
TP	TOTAL PILOTS
VC	VARIABLE COST INDEX (\$/HR), (1972 \$, 1972-1)

followed by a repeat of the question

WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
?

Any variable identifier from the above list can be specified. For example, in order to plot the total number of aircraft, the user responds

AASUM

Now the computer will ask

PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OR LIST
?

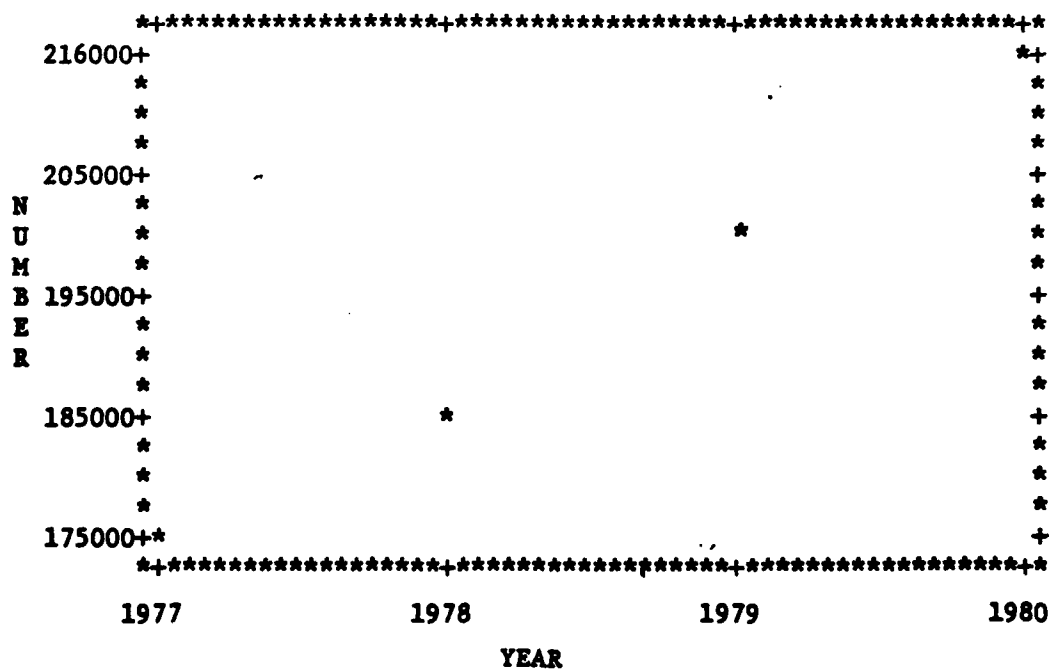
By responding

TIME

the following plot will be displayed,

GENERAL AVIATION DYNAMICS MODEL PAGE 2

TOTAL NUMBER OF AIRCRAFT, 1977 TO 1980



Upon completion of the plot the computer will again ask

WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
?

Enter the next variable to be plotted

HFSUM

Now the computer will ask

PLOT THIS VARIABLE AGAINST TIME OR ANOTHER VARIABLE OR LIST
?

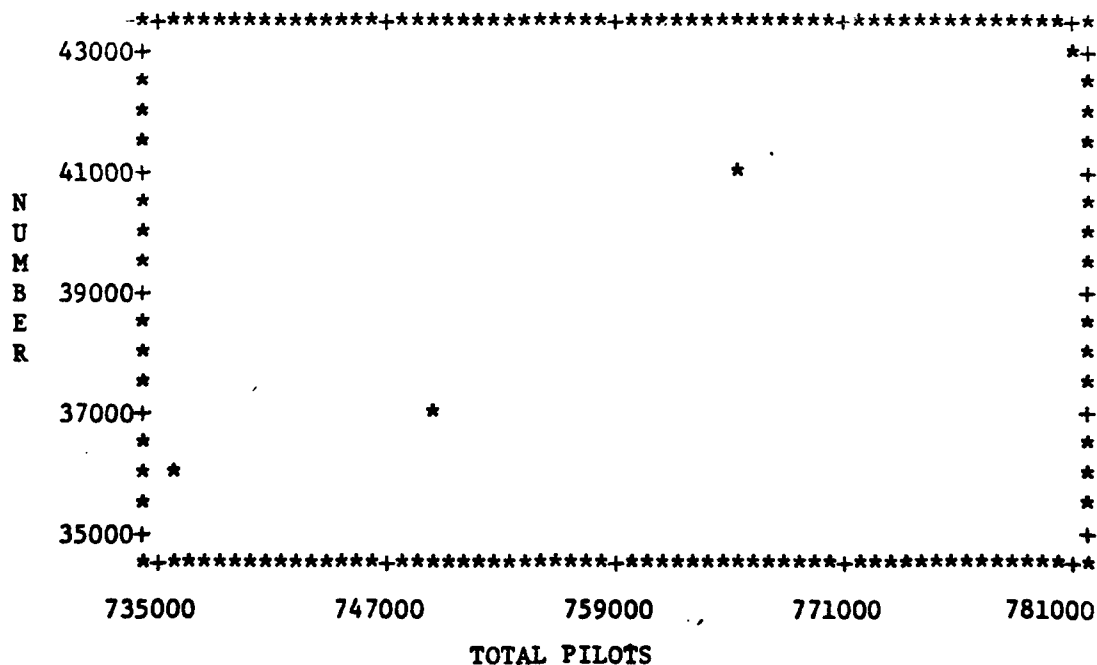
To plot the total hours flown against the number of pilots, enter

TP

The following plot will be displayed

GENERAL AVIATION DYNAMICS MODEL PAGE 3

TOTAL PILOTS VS TOTAL HOURS FLOWN (THOUSANDS), 1977 TO 1980



Again the computer will ask

WHAT PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
?

Entering

NONE

will cause the computer to print

STEP 9 -- SAVE THE RESULTS OF THIS FORECAST FOR FUTURE SENSITIVITY ANALYSIS.

followed by

WOULD YOU LIKE TO SAVE THE RESULTS OF THIS SESSION FOR LATER SENSITIVITY
ANALYSIS
?

*When performing a sensitivity analysis, any simulation run may become the base-
line for future comparison. If the current run is desired to be a baseline
for comparisons, enter*

YES

The computer will save the results of the current run and respond

WOULD YOU LIKE TO CONTINUE WITH ANOTHER FORECAST
?

The reply

YES

will cause the computer to print

ENTER ENDING YEAR FOR SIMULATION
?

Entering a valid year

1980

allows the system to continue with

STEP 1 -- WOULD YOU LIKE TO COMPUTE THE FORECAST WITH THE INITIAL ASSUMPTIONS UNCHANGED (YES OR NO) OR VIEW THE STEPS OF THIS MODEL (TEACH)
?

Since we are now familiar with the steps of the model we do not need to set the TEACH flag on. Answer

NO

since we already have stored the results of the simulation with the initial assumptions unchanged. The computer will ask

ENTER NAME OF VARIABLE TO BE CHANGED, OR LIST, OR NONE
?

The user has already decided that the second simulation will involve a fuel tax increase, but does not know how to implement that in the model. Therefore answer

LIST

which will generate the following list:

AIRCRAFT VARIABLES

IDENTIFIER DESCRIPTION

ADRN AIRCRAFT DESTRUCTION RATE, NORMALIZED (AC/YR)
 FC FIXED COST INDEX (1972 \$, 1972=1). ONE COMPONENT OF FC IS THE
 ANNUALIZED INVESTMENT
 FCINF FIXED COST INFLATION FACTOR
 VC VARIABLE COST INDEX (\$/HR), (1972 \$, 1972=1). COMPONENTS OF
 VC ARE FTAX (FEDERAL FUEL TAX) AND LFEE (LANDING FEE).
 VCINF VARIABLE COST INFLATION FACTOR
 FHRF FLYING HOURS REQUIRED FACTOR

ECONOMIC VARIABLES

IDENTIFIER DESCRIPTION

DPI DISPOSABLE PERSONAL INCOME (1972 \$, 1972=1)
 GNP GROSS NATIONAL PRODUCT (1972 \$, 1972=1)
 RAD REVENUE AIRCRAFT DEPARTURES (1972=1)
 ECONOMIC DPI, GNP, RAD

FUEL VARIABLES

IDENTIFIER DESCRIPTION

SFC SPECIFIC FUEL CONSUMPTION

PILOT VARIABLES

IDENTIFIER DESCRIPTION

ATPDN AIRLINE TRANSPORT PILOT DEPARTURE RATE, NORMALIZED
 CPDN COMMERCIAL PILOT DEPARTURE RATE, NORMALIZED
 IPDN INSTRUMENT PILOT DEPARTURE RATE, NORMALIZED
 PCIN PRIVATE CERTIFICATES ISSUED RATE, NORMALIZED
 PPDN PRIVATE PILOTS DEPARTURE RATE, NORMALIZED
 SPDN STUDENT PILOTS DEPARTURE RATE, NORMALIZED
 URIPN UPGRADE TO INSTRUMENT PILOT RATE, NORMALIZED
 SCIX STUDENT CERTIFICATES ISSUED MULTIPLIER
 PILOT ATPDN, CPDN, IPDN, PCIN, PPDN, SPDN, URIPN

Upon completion of the list, the computer will repeat the question

ENTER NAME OF VARIABLE TO BE CHANGED, OR LIST OR NONE

?

The user can see from the above list that the federal fuel tax is a component of VC the variable cost index and so enters

VC

The computer responds by displaying the current values of the variable cost index components

THE COMPONENTS OF THE VARIABLE COST INDEX ARE FTAX, THE FEDERAL FUEL TAX, AND LFEE, THE LANDING FEE
THE CURRENT VALUES FOR THE VARIABLE COST INDEX COMPONENTS ARE
FEDERAL FUEL TAX (\$/GAL)

				AV GAS		JET FUEL	
				1977	1978	1979	1980
				0.07	0.07	0.07	0.07
				0.07	0.07	0.07	0.07
				0.07	0.07	0.07	0.07
				0.07	0.07	0.07	0.07
LANDING FEE (\$/LANDING)							
	SNGL-P NON-AER	SNGL-P AER	MULTI- PISTON	TURBO PROP	TURBO JET	PISTON HELIC	TURBINE HELIC
1977	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1978	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1979	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1980	0.00	0.00	0.00	0.00	0.00	0.00	0.00

and then asks

DO YOU WISH TO CHANGE THE VALUES OF FTAX, LFEE OR NONE
?

To change the fuel tax values the user enters

FTAX

and the computer responds

WHAT YEAR WOULD YOU LIKE THE NEW FUEL TAX TO BEGIN
?

For a January 1, 1979 date of effectiveness enter

1979

The computer will ask

WOULD YOU LIKE THE FUEL TAX TO REMAIN CONSTANT FOR ALL SUBSEQUENT YEARS
?

If the desired fuel tax change is to increase the fuel tax by 5¢ a year, the answer is

NO

The computer will ask

WOULD YOU LIKE THE FUEL TAX TO CHANGE AT A CONSTANT RATE FOR ALL SUBSEQUENT YEARS

?

Again the answer is

NO

The computer will respond by asking for the fuel tax values to be entered explicitly for each year, starting with the year of the change

ENTER FUEL TAX VALUES, IN DOLLARS, FIRST FOR AVIATION GAS, THEN FOR JET FUEL. ENTER VALUES FOR EACH YEAR.

?

The 1979 fuel tax values are entered first

0.12 0.12

The computer then asks for the values for the next year

?

and the values for 1980 are entered

0.17 0.17

Since this simulation is to end in 1980 no more values are required. The computer then displays the new values of the fuel tax

THE NEW VALUES FOR THE FUEL TAX ARE
FEDERAL FUEL TAX (\$/GAL)

	AV GAS	JET FUEL
1977	0.07	0.07
1978	0.07	0.07
1979	0.12	0.12
1980	0.17	0.17

and then repeats the question

DO YOU WISH TO CHANGE THE VALUES OF FTAX, LFEE OR NONE
?

Since the new fuel tax has been entered as desired and the user does not want to impose a landing fee, enter

NONE

The computer then asks whether any other variables are to be changed

ENTER NAME OF VARIABLE TO BE CHANGED, OR LIST, OR NONE
?

Since no other variables are to be changed enter

NONE

Since there are no more changes, the simulation is run. On finishing execution of the simulation the computer asks

DO YOU WANT TO SEE TABLES OF RESULTS OF THE FORECAST
?

Responding

YES

the computer then asks

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.
?

Responding as for the first simulation,

PILOTS

will generate the following table

GENERAL AVIATION DYNAMICS MODEL PAGE 4

PILOT DATA, 1977 TO 1980

	1977	1978	1979	1980
STUDENT PILOTS	188,801	183,794	183,654	179,129
PRIVATE PILOTS	309,005	323,821	335,104	345,428
COMMERCIAL PILOTS	187,801	189,699	192,068	194,521
AIR TRANSPORT PILOTS	45,072	47,784	50,879	53,766
PILOT SUBTOTAL	730,679	745,098	761,705	772,844
HELICOPTER PILOTS	4,804	4,333	3,940	3,534
TOTAL PILOTS	735,483	749,431	765,645	776,378
INSTRUMENT RATINGS	211,364	221,497	232,437	243,148
HELICOPTER RATINGS	23,012	24,395	25,739	27,052
TOTAL HELIC RATINGS	27,816	28,728	29,679	30,586

Upon completion of the table the computer will again ask

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE.

?

If no more output tables are desired, enter

NONE

The computer will ask

DO YOU WANT TO SEE PLOTS OF RESULTS OF THE FORECAST

?

Responding

NO

will cause the computer to ask

DO YOU WANT SENSITIVITY ANALYSIS, (THE PREVIOUSLY SAVED FORECAST IS THE BASELINE),
(YES OR NO)

?

Answer

YES

The computer will ask

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
?

The list of output tables for sensitivity analyses is a subset of the list for absolute forecasts. Therefore, the unfamiliar user should enter

LIST

which will generate the following list of output tables

TABLE	VARIABLES IN TABLE
AA1	ACTIVE AIRCRAFT BY YEAR
AA2	ACTIVE AIRCRAFT BY USER CATEGORY
AIRPORTS	LOCAL AND ITINERANT OPERATIONS PLUS IFR FLIGHT PLANS FILED
AIRUTIL	AIRCRAFT UTILIZATION RATES
FUEL	FUEL CONSUMED
HOURSFLOWN	HOURS FLOWN
OPERATIONS	TOTAL OPERATIONS
PILOTS	SP,PP,CP,ATP,P,HP,TP,IP,HR,THP
REVENUE	FEDERAL TAX REVENUE
TOTALS	TOTAL AIRCRAFT, TOTAL HOURS FLOWN, TOTAL OPERATIONS

The computer will then repeat the question

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
?

By responding

PILOTS

the following table of percent deviations will be generated

GENERAL AVIATION DYNAMICS MODEL PAGE 5

PILOT DATA, PERCENT DEVIATION FROM BASELINE, 1977 TO 1980

	1977	1978	1979	1980
STUDENT PILOTS	0.00	0.00	0.00	-2.21
PRIVATE PILOTS	0.00	0.00	0.00	0.24
COMMERCIAL PILOTS	0.00	0.00	0.00	-0.42
AIR TRANSPORT PILOTS	0.00	0.00	0.00	0.00
PILOT SUBTOTAL	0.00	0.00	0.00	-0.52
HELICOPTER PILOTS	0.00	0.00	0.00	-2.05
TOTAL PILOTS	0.00	0.00	0.00	-0.53
INSTRUMENT RATINGS	0.00	0.00	0.00	-0.34
HELICOPTER RATINGS	0.00	0.00	0.00	0.00
TOTAL HELIC RATINGS	0.00	0.00	0.00	-0.24

Note that since the pilot data reported in 1979 is for the year 1978 as reported on January 1, 1979, the effects of the changed fuel tax in 1979 are not felt until the 1980 pilot data. The computer will again ask

WHAT OUTPUT TABLE WOULD YOU LIKE, OR ENTER LIST OR NONE
?

If no more sensitivity output tables are desired, enter

NONE

The computer will ask

DO YOU WANT TO SEE PLOTS OF THE SENSITIVITY ANALYSIS RESULTS
?

If sensitivity plots are desired enter

YES

The computer will ask

WHAT SENSITIVITY PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
?

To obtain the list of possible sensitivity plots enter

LIST

and the computer will generate the following list

IDENTIFIER	DESCRIPTION
AA	ACTIVE AIRCRAFT BY PRIMARY USE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
AASUM	TOTAL AIRCRAFT
ATP	AIR TRANSPORT PILOTS
CP	COMMERCIAL PILOTS
FC	FUEL CONSUMED DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
FTR	FEDERAL TAX REVENUE DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
HFSUM	TOTAL HOURS FLOWN
HP	HELICOPTER PILOTS
HR	HELICOPTER RATINGS
IP	INSTRUMENT RATINGS
OPS	OPERATIONS (THOUSANDS) DURING PREVIOUS YEAR, AS REPORTED ON JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION FROM BASELINE
OPSUM	TOTAL OPERATIONS
PP	PRIVATE PILOTS
P	PILOT SUBTOTAL
SP	STUDENT PILOTS
TC	TOTAL COST, AS PERCENT DEVIATION FROM BASELINE
THP	TOTAL HELIC RATINGS
TP	TOTAL PILOTS

Upon completion of the list the computer will repeat the question

WHAT SENSITIVITY PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
?

To see the effect of the changed fuel tax on the fuel consumption, enter

FC

The computer will respond

PLEASE ENTER EITHER 1 FOR AVIATION GAS OR 2 FOR JET FUEL FOR THE PLOT
?

For aviation gas, enter

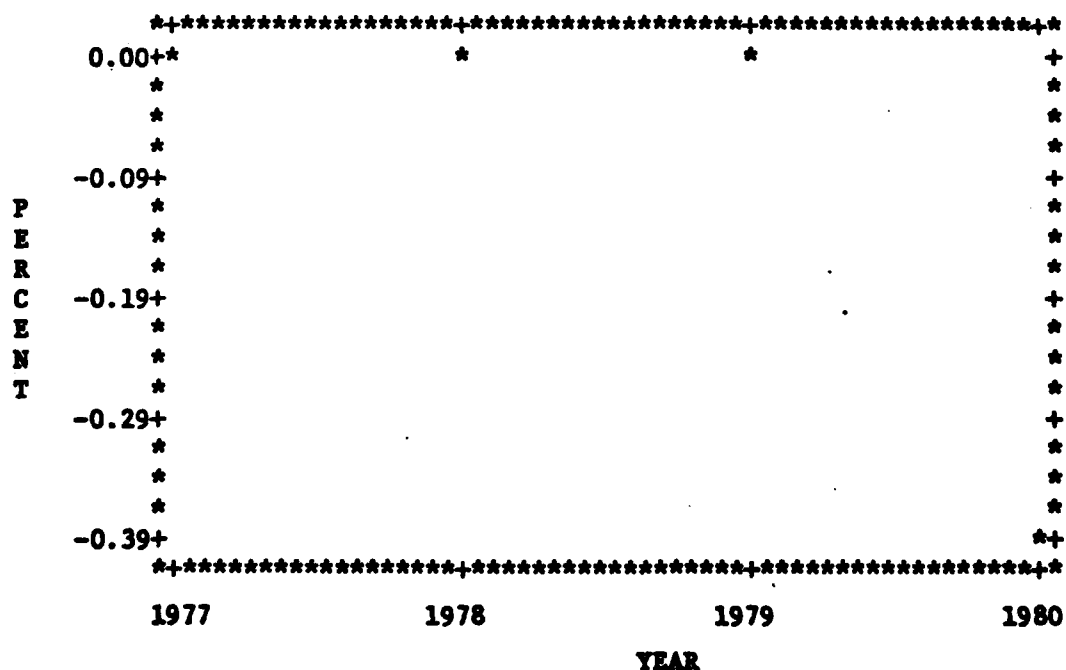
1

and the computer will generate the following sensitivity plot

GENERAL AVIATION DYNAMICS MODEL PAGE 6

FUEL CONSUMED DURING PREVIOUS YEAR, AS REPORTED ON
JANUARY 1 OF DESIGNATED YEAR, AS PERCENT DEVIATION
FROM BASELINE, 1977 TO 1980

AVIATION GAS



Upon completing the plot the computer will repeat the question

WHAT SENSITIVITY PLOT WOULD YOU LIKE. ENTER THE VARIABLE, OR LIST, OR NONE
?

For no further sensitivity plots enter

NONE

The computer will ask

WOULD YOU LIKE TO SAVE THE RESULTS OF THIS SESSION FOR LATER SENSITIVITY
ANALYSIS
?

If the previous run is not to become a new baseline enter

NO

The computer will ask

WOULD YOU LIKE TO CONTINUE WITH ANOTHER FORECAST
?

If no more forecasts are desired, enter

NO

The computer will respond

YOU ARE NOW LEAVING THE GENERAL AVIATION DYNAMICS MODEL.

*and return the user to the interactive session with the computer for log-out.
On the UCS computer the computer responds*

END UNICMD

and the user can log-out by entering

BYE

The above interactive session was run on the UCS computer. The only difference between it and a similar run on the Battelle computer, apart from the log-in and log-out procedures, is in the occurrence of a / as a user prompt instead of ? as above.

Section IV: Batch use of the GAD model

The GAD model may be run in batch mode on both the Battelle Computer and the UCS computer

IV.1 Batch mode on the Battelle computer

The control cards required are as follows:

Job card
ATTACH,PROFIL,AVIATION,ID=NUCLEUS,MR=1.
BEGIN,GAD.
*EOR
SYSTEM,USER=OFF,WIDTH=131
LOAD,GAD
Appropriate responses to GAD conversation, one per card.
.
.
.
*EOF

IV.2 Batch mode on the UCS computer

The control cards required are as follows:

Job card
Account card
Cards giving instructions for mailing output, if any.
GET,NUCLEUS.
GET,TAPE1=GADCOMP,TAPE2=GAFILE2,TAPE3=GAFILE3.
NUCLEUS.
EOR
SYSTEM,USER=OFF,WIDTH=131
LOAD,GAD
Appropriate responses to GAD conversation, one per card.
.
.
.
EOF

ATE
MED
80