NASA Technical Paper 1690

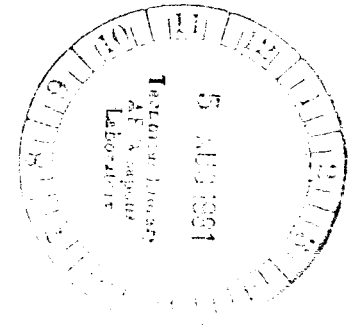


# Programmer's Manual for MMLE3, a General FORTRAN Program for Maximum Likelihood Parameter Estimation



Richard E. Maine

JUNE 1981





NASA

NASA Technical Paper 1690

# Programmer's Manual for MMLE3, a General FORTRAN Program for Maximum Likelihood Parameter Estimation

Richard E. Maine
*Dryden Flight Research Center*
*Edwards, California*

National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1981

CONTENTS

# PROGRAMMER'S MANUAL FOR MMLE3,

# A GENERAL FORTRAN PROGRAM FOR

# MAXIMUM LIKELIHOOD PARAMETER ESTIMATION

Richard E. Maine
Dryden Flight Research Center

## INTRODUCTION

This report is a programmer's manual for the FORTRAN IV computer program MMLE3, a maximum likelihood parameter estimation program capable of handling general bilinear dynamic equations of arbitrary order with measurement noise and/or state noise (process noise). The basic MMLE3 program is quite general and, therefore, applicable to a wide variety of problems. The basic program can interact with a set of user-written problem-specific routines to simplify the use of the program on specific systems. A set of user routines for the aircraft stability and control derivative estimation problem is provided with the program. A companion document, the User's Manual (ref. 1), describes the theory and use of the program. This paper contains program listings and suggestions for implementation on various computer systems. Enough information is given about the purpose and operation of each subroutine so that users can make modifications if desired. Complete listings and reference maps of the routines are included on microfiche as supplement 1. Four test cases are discussed; listings of the input cards and program output for the test cases are included on microfiche as supplement 2.

It is advised that sections 1 and 2 of this paper be read carefully before attempting to implement the MMLE3 program on a computer system. The remainder of the paper, particularly appendix A, is a reference for detailed information about the structure and coding of the program.

## NOMENCLATURE

| | |
|---|---|
| $A$ | state equation matrix |
| $a_n$ | normal acceleration, g |
| $a_x$ | longitudinal acceleration, g |
| $B$ | state equation matrix |
| $C$ | observation matrix |
| $C_A$ | axial force coefficient |
| $C_L$ | lift coefficient |
| $C_\ell$ | rolling moment coefficient |
| $C_m$ | pitching moment coefficient |
| $C_N$ | normal force coefficient |
| $C_X$ | longitudinal force coefficient |
| $C_Y$ | lateral force coefficient |
| $D$ | observation matrix |
| $E$ | observation matrix |
| $FF^*$ | state noise power spectral density matrix |
| $GG^*$ | residual covariance matrix |
| $\mathcal{G}\mathcal{G}^*$ | measurement noise covariance matrix |
| $H$ | observation matrix |
| $K$ | Kalman filter gain matrix (program variable name KGAIN) |
| $n$ | state noise vector |
| $P$ | Riccati covariance matrix |
| $q$ | pitch rate, deg/sec |

| | |
|---|---|
| $\bar{q}$ | dynamic pressure, $N/m^2$ ($lbf/ft^2$) |
| $R$ | state equation matrix |
| $S$ | state equation matrix |
| $t$ | time, sec |
| $u$ | control vector |
| $V$ | velocity, m/sec (ft/sec) |
| $\nu$ | forcing function in state equation |
| $w$ | forcing function in observation equation |
| $x$ | state vector |
| $\hat{x}$ | corrected state vector |
| $z$ | observation vector |
| $\tilde{z}$ | predicted observation vector |
| $\alpha$ | angle of attack, deg |
| $\beta$ | angle of sideslip, deg |
| $\Delta t$ | time interval, sec |
| $\delta_a$ | aileron deflection, deg |
| $\delta_e$ | elevator deflection, deg |
| $\eta$ | measurement noise vector |
| $\theta$ | pitch angle, deg |
| $\varphi$ | bank angle, deg |
| $\psi$ | integral of the transition matrix |
| $\nabla$ | gradient (row vector) |

Superscript:

| | |
|---|---|
| * | transpose |

Subscripts:

| | |
|---|---|
| $i$ | general index |
| $\alpha$, $\alpha^2$, $\beta$, $\delta_a$, $\delta_e$ | derivative with respect to indicated quantity, per deg or per deg$^2$ |
| $\dot{\alpha}$ | derivative with respect to rate of change of angle of attack, per rad/sec |
| 0 | bias |

Prefix to matrix names:

| | |
|---|---|
| APR | *a priori* weighting |

Suffixes to matrix names:

| | |
|---|---|
| I | inverse |
| L | dimensionalization addition |
| M | dimensionalization ratio |
| N | nondimensional |
| V | variation |

Computer labels:

| | |
|---|---|
| ALPHA | angle of attack, deg |
| ALT | altitude, m (ft) |
| AN | normal acceleration, g |
| AX | longitudinal acceleration, g |
| AY | lateral acceleration, g |
| BETA | angle of sideslip, deg |
| DELTA-A | aileron deflection, deg |
| DELTA-E | elevator deflection, deg |
| DELTA-R | rudder deflection, deg |
| MACH | Mach number |
| P | roll rate, deg/sec |

| | |
|---|---|
| PHI | bank angle, deg |
| Q | pitch rate, deg/sec |
| Q-BAR | dynamic pressure, $N/m^2$ ($lbf/ft^2$) |
| R | yaw rate, deg/sec |
| THETA | pitch angle, deg |
| V | velocity, m/sec (ft/sec) |

# 1.0 FORMAT OF DECKS, LISTINGS, AND MODIFICATIONS

In this section, the conventions used for reference to specific cards in the MMLE3 program are defined. The program is maintained at the Dryden Flight Research Center as an UPDATE (ref. 2) file under the CDC SCOPE 3.4 and NOS 1.4 operating systems. Users familiar with CDC computers will recognize the UPDATE format and the FORTRAN extended reference maps (ref. 3). Supplement 1 is a microfiche listing of the MMLE3 program and reference maps. Appendix A contains detailed discussions of individual subroutines and common blocks.

## 1.1 Decks and Common Decks

This section describes the format of the card decks of the MMLE3 program. The format of these decks is as an UPDATE (ref. 2) source file. This format is used for the relative ease of changing matrix dimensions (see sec. 3.2).

Users with access to UPDATE will find it most convenient to maintain the program as an UPDATE library; the card decks provided can be used directly as input to UPDATE. For users without access to UPDATE, the program COMSUB is provided (appendix B) to translate the UPDATE source decks into FORTRAN code. Most of the sophisticated features of UPDATE are avoided, so COMSUB is a very simple program. Its only function is common deck substitution.

A common deck is a group of cards that can be copied into several different subroutines. Although the name is similar, it has no direct relation with FORTRAN COMMON statements; FORTRAN COMMON statements are one convenient application of UPDATE common decks. The main advantage of using common decks is in the ease of making program modifications. If a common deck is modified, the modifications will automatically apply to every copy of that common deck. Since it is not rare for a common deck to be copied in 10 or more subroutines, the work of making program modifications can be reduced by an order of magnitude.

The first section of the MMLE3 cards consists of the common decks. Each common deck is preceded by an identifying card in the format

```
*COMDECK common-deck-name
```

The MMLE3 program and subroutines are placed after the common decks in the order shown in appendix A. Each routine (including the main program) is preceded by a card in the format

*DECK deck-name

This card can be ignored unless UPDATE is used. Wherever a copy of a common deck is desired, there is a card in the format

*CALL common-deck-name

The program COMSUB substitutes a copy of the appropriate common deck in place of this card.

## 1.2 Reference Maps

The subroutines in supplement 1 are followed by reference maps. The reference maps list the variable names in alphabetical order and give the line number of each use of the variable names. The line numbers used in these reference maps are the numbers that appear on the left of the listings every fifth line. These line numbers are used only in the reference maps; all other references to individual cards in this report will use the card numbers (sec. 1.3). A complete description of the information in the reference maps is found in reference 3.

## 1.3 Card Numbers

Except for the reference maps, all references to individual cards use the card numbers shown on the right of the listings in supplement 1 (columns 73 to 90). The card numbers consist of an ident name and a number; both are necessary to specify a card. Most of the cards within a subroutine have the subroutine name as an ident.

Cards with an ident different from the subroutine name are either part of a modification or a common deck. If the first character of the ident is $, the card is part of a modification. There are no modifications in the current listing; this description is included to allow for possible future changes (see sec. 1.4).

If the ident of a card is not the same as the subroutine name and does not begin with $, that card is part of a common deck. Common decks are described in section 1.1 Any reference to a card in a common deck applies to every copy of the common deck. All of the *CALL cards calling for common decks are listed separately, after the listings of the common decks, before the main program.

## 1.4 Modifications

The MMLE3 program is designed as a working tool that can be modified to fit specific applications, rather than as an inviolate whole. In addition, it is possible that modifications will be necessary to correct program bugs (naturally, we hope not). Therefore, this section describes the conventions to be used by any modifications; these conventions are a subset of CDC UPDATE (ref. 2). Users with access to

CDC computers will find it convenient to use UPDATE to implement modifications. For other users, the descriptions and example below should be adequate to define the actions required to implement any modifications.

Each group of modifications is preceded by a card in the format

*ID correction-set-name

This card defines the ident (sec. 1.3) to be used as part of the card number for any cards added by this correction set. For the MMLE3 program, the convention has been established that the first character of every correction set name will be $. Correction set names are limited to nine characters in length.

Insertion of new cards into the program is defined by a card in the format

*I card-number

immediately followed by one or more cards to be inserted. The card number is in the format

ident.number

and defines the card after which the new cards are to be inserted. If the card number describes a card in a common deck, the new cards are to be inserted in every copy of the common deck. Cards may be inserted after any card in the program, including cards inserted by previous modifications and cards which call for inclusion of common decks. Cards to be inserted may include cards that call for inclusion of common decks.

Deletion of cards from the program is defined by a card in the format

*D card-number

or

*D card-number-1.card-number-2

The first format describes a single card to be deleted; the second format is used to delete all cards from card number 1 to card number 2 inclusive. The *D card may be followed by one or more cards to be inserted in place of the deleted card(s). The number of cards inserted in this manner does not have to equal the number of cards deleted; it can be larger or smaller.

Cards beginning with */ are comments and can be ignored.

The following simple example should help clarify some of the ideas of this section. The correction set reads the time history from a tape with the time in total milliseconds instead of hours, minutes, seconds, and milliseconds. The dimensions are increased to allow tape records up to 150 words long, instead of 100.

```
*ID $LONGTAPE

*/ READS LONG TAPE RECORDS

*/ TIME IN TOTAL MS

*D RECRD.2

     COMMON /RECRD/ EOFTH,T(4),RECORD(150)

*D READTH.20

     READ(UDATA) ITMS,(RECORD(I),I=1,NREC)

     CALL IHMSMS(ITMS,T)
```

Note, in particular, that the card RECRD.2 is changed in both copies of the common deck, RECRD (in subroutines READTH and THDATA). The original subroutines READTH and THDATA are in supplement 1. The routines resulting from the above modification are shown below; the listing of THDATA is truncated since there are no changes in the latter part of the subroutine.

```
      SUBROUTINE READTH(INSTAT)                                          READTH     2
C                                                                        READTH     3
C     READS ONE POINT OF INPUT TIME HISTORY                              READTH     4
C     STANDARD VERSION FOR CARD OR TAPE INPUT                            READTH     5
C     INSTAT GIVES INPUT STATUS                                          READTH     6
C     0 INDICATES FIRST CALL TO READTH FOR THIS CASE                     READTH     7
C     1 INDICATES SEARCHING FOR A START TIME, BUT NOT FIRST CALL         READTH     8
C     2 INDICATES READING DATA                                           READTH     9
C     THIS ROUTINE SHOULD NOT ALTER INSTAT                               READTH    10
C                                                                        READTH    11
      COMMON /FILES/ LCARD,LPUNCH,LPRINT,LDATA,LT1,LT2,LTHOUT,LWT,LPLOT  FILES     2
      INTEGER LCARD,LPUNCH,LPRINT,LDATA,LT1,LT2,LTHOUT,LWT,LPLOT         FILES     3
      COMMON /INOPT/ CARD,TAPE                                           INOPT     2
      LOGICAL CARD,TAPE                                                  INOPT     3
      COMMON /INORD/ NREC,ZCHAN( 08),UCHAN( 04),EXCHAN( 20)              INORD     2
      INTEGER ZCHAN,UCHAN,EXCHAN                                         INORD     3
      COMMON /RECRD/ EOFTH,T(4),RECORD(150)                              SLONGTAPE 1
      LOGICAL EOFTH                                                      RECRD     3
      INTEGER T                                                          RECRD     4
      COMMON /TAPPOS/ ITM,REW                                            TAPPOS    2
      LOGICAL REW                                                        TAPPOS    3
C                                                                        READTH    17
      IF(CARD) GO TO 100                                                 READTH    18
      IF(REW) REWIND LDATA                                               READTH    19
      READ(LDATA) ITMS,(RECORD(I),I=1,NREC)                              SLONGTAPE  2
      CALL IHMSMS(ITMS,T)                                                SLONGTAPE  3
      GO TO 500                                                          READTH    21
  100 READ(LCARD,1000) T,(RECORD(I),I=1,NREC)                            READTH    22
C                                                                        READTH    23
 1000 FORMAT(3I2,I3,1X,7F10.4/(8F10.4))                                  READTH    24
  500 RETURN                                                             READTH    25
      END                                                                READTH    26
```

```
      SUBROUTINE THDATA                                                 THDATA     2
C                                                                       THDATA     3
C     READS INPUT TIME HISTORIES, FINDS AVERAGES OF ALL SIGNALS.        THDATA     4
C                                                                       THDATA     5
C     LOOPS WITH T=1,LORD DEPEND ON RELATIVE ORDER OF Z, U, AND EXTRA   THDATA     6
C          VARIABLES, SCALES, BIASES, CHANNEL NUMBERS, OR AVERAGES      THDATA     7
C          IN THE COMMON BLOCKS.                                        THDATA     8
C          REFERENCES TO Z VARIABLES THEN EXTEND TO U AND EXTRA.        THDATA     9
C                                                                       THDATA    10
      COMMON /AVGCOM/ ZAVG( 08),UAVG( 04),EXAVG( 20),ZSIG( 08),        AVGCOM     2
     -     USIG( 04),EXSTG( 20),ZMINM( 08),UMINM( 04),EXMINM( 20),     AVGCOM     3
     -     ZMAXM( 08),UMAXM( 04),EXMAXM( 20)                           AVGCOM     4
      COMMON /BILIN/ USEAVG,TIMVAR,Z( 08),U( 04),EXTRA( 20),ONES( 04)  BILIN     2
      LOGICAL USEAVG,TIMVAR                                            BILIN     3
      COMMON /COM/ NCASE,NPTT,NPTS(15),ITMSTS(15)                       COM       2
      COMMON /FILES/ UCARD,UPUNCH,UPRINT,UDATA,UT1,UT2,UTHOUT,UWT,UPLOT FILES     2
      INTEGER UCARD,UPUNCH,UPRINT,UDATA,UT1,UT2,UTHOUT,UWT,UPLOT        FILES     3
      COMMON /HEADNG/ TITLE(20),ADATE,ATIME,                            HEADNG    2
     -     SIGLAB(2, 08),XLAB(2, 07),CONLAB(2, 04),EXLAB(2, 20)        HEADNG    3
      COMMON /INORD/ NREC,ZCHAN( 08),UCHAN( 04),EXCHAN( 20)             INORD     2
      INTEGER ZCHAN,UCHAN,EXCHAN                                        INORD     3
      COMMON /INTEGR/ DT,NEAT                                           INTEGR    2
      COMMON /MAXIMS/ MAXX,MAXZ,MAXU,MAXB,LEX,LORD                      MAXIMS    2
      COMMON /MODCOM/ UMOD                                              MODCOM    2
      LOGICAL UMOD                                                      MODCOM    3
      COMMON /RECRD/ EOFTH,T(4),RECORD(150)                             $LONGTAPE 1
      LOGICAL EOFTH                                                     RECRD     3
      INTEGER T                                                         RECRD     4
      COMMON /TAPPOS/ TIM,REW                                           TAPPOS    2
      LOGICAL REW                                                       TAPPOS    3
      COMMON /TCDATA/ STC(15),ETC(15),THIN,PRINTI,MAXREC,               TCDATA    2
     -     ZBIAS( 08),UBIAS( 04),EXBIAS( 20),ZSCALE( 08),USCALE( 04),  TCDATA    3
     -     EXSCAL( 20)                                                 TCDATA    4
      INTEGER THIN,STC,ETC                                              TCDATA    5
      LOGICAL PRINTI                                                    TCDATA    6
C                                                                       THDATA    23
      LOGICAL FIRST                                                     THDATA    24
C                                                                       THDATA    25
```

## 1.5 Tapes

The following format will be used for tape transmittal of the MMLE3 program unless explicitly requested otherwise. The availability of tape copies of the MMLE3 program can be ascertained by writing the author.

Tapes are nine-track 800 BPI labeled tapes. The VSN of the tape is MMLE3T. The label is American National Standard Institute (ANSI) standard with the name MMLE3. All data are ASCII-coded card images. (EBCDIC code is available on request.) Each card image is a fixed-length 80-character record. Records are blocked in fixed-length blocks of length 1200 characters. Each block contains exactly 15 records with no padding; records do not span blocks.

Each file of data on a tape is terminated by a card with "END-OF-FILE-nn" in the first 14 columns. The nn is replaced by the file number. The remaining 66 columns of the card are filled with dashes. Actual system end-of-files are not used because of possible incompatibility between computers.

There are 12 files of data on the tape. The first file is the UPDATE source cards for the MMLE3 program as described in section 1.1. The second file is the UPDATE source cards for the EISPACK routines used by MMLE3. The third file is the CDC segmentation directives. The fourth and fifth files are the COMPUN program and the template used by COMPUN as described in appendix C. The sixth file is an alternate template for COMPUN, which results in a correction set for CDC UPDATE instead of a complete set of common decks. The seventh file is the COMSUB program

described in appendix B. The eighth and ninth files contain the program modification and input cards for the one-dimensional check case listed in appendix D. The 10th and 11th files contain the program modification and input cards for the two longitudinal aircraft check cases described in appendix D. The 12th file contains the input cards for the lateral-directional standard aircraft test case described in appendix D.

A microfiche listing of the tape contents will be supplied with the tape. The total length of the tape is approximately 10,400 card images.

The following short program can be used to pick a file from the tape. The file number desired is specified on a single card in I10 format. The tape is assigned FORTRAN unit number 11 and the file requested is written to FORTRAN unit number 12.

```
      PROGRAM PICK(INPUT,OUTPUT,MMLE3T,FILEN,
     -     TAPE1=INPUT,TAPE3=OUTPUT,TAPE10=MMLE3T,TAPE11=FILEN)
C
C     PICK A "FILE" OFF OF MMLE3T TAPE AND COPY IT TO FILEN.
C     INPUT IS A SINGLE CARD WITH THE DESIRED FILE NUMBER IN FORMAT I10.
C
      INTEGER UREAD,UPRINT,UIN,UOUT
      REAL CARD(20),END(3)
      DATA END/4HEND-,4HOF-F,4HILE-/
      DATA UREAD/11/,UPRINT/3/,UIN/10/,UOUT/11/
C
C     ******************** READ REQUESTED FILE NUMBER.
      READ (UREAD,8100) NPICK
      WRITE (UPRINT,8300) NPICK
C     ******************** SKIP PRECEEDING FILES.
      REWIND UIN
      NSKIP = NPICK-1
      IF (NSKIP.LE.0) GO TO 300
      DO 200 IFILE = 1 , NSKIP
           DO 100 ICARD = 1 , 20000
                READ (UIN,8900) (CARD(I),I=1,4)
                IF (CARD(1).EQ.END(1) .AND. CARD(2).EQ.END(2) .AND.
                    CARD(3).EQ.END(3)) GO TO 150
  100      CONTINUE
  150      WRITE (UPRINT,8301) CARD(4),ICARD
  200 CONTINUE
C     ******************** COPY DESIRED FILE.
  300 DO 400 ICARD = 1 , 20000
           READ (UIN,8900) CARD
           IF (CARD(1).EQ.END(1) .AND. CARD(2).EQ.END(2) .AND.
     -         CARD(3).EQ.END(3)) GO TO 500
           WRITE (UOUT,8900) CARD
  400 CONTINUE
  500 WRITE (UPRINT,8302) CARD(4),ICARD
C
 8100 FORMAT(I10)
 8300 FORMAT("0FILE NUMBER",I3," REQUESTED FROM THE TAPE.")
 8301 FORMAT(" FILE ",A2," SKIPPED.",I6," CARDS IN FILE.")
 8302 FORMAT("0FILE ",A2," COPIED.",I6," CARDS IN FILE.")
 8900 FORMAT(20A4)
      STOP
      END
```

# 2.0 IMPLEMENTATION CONSIDERATIONS

This section discusses the considerations in implementing the MMLE3 program on various computer systems. The program was designed with generality in mind, so code peculiar to specific systems was avoided. The MMLE3 program has been checked out on CDC computers with SCOPE 3.4 and NOS 1.4 operating systems and on IBM computers with the OS/360 operating system. It has also been run on Univac and Harris equipment. It should not be difficult to implement on any large computer system with a FORTRAN IV compiler and a CalComp plotter (ref. 4).

## 2.1 FORTRAN

The MMLE3 program uses mostly ANSI standard FORTRAN IV (ref. 5). All character data are stored four characters per word for machine generality. The few statements used that do not conform to ANSI standards are described below. Compilers on most large computer systems accept these usages.

The program card, which is nonstandard, is necessary on CDC systems to define the files to be used. This card appears only in the main program and should be deleted if it is not appropriate to a particular system.

Nonstandard subscripts are used in several places. Most large system compilers accept the forms used. If a particular compiler does not accept them, the nonstandard subscripts can be replaced by dummy variables, defined immediately before their use.

Unsubscripted array names are used in data statements throughout the program. This usage, although nonstandard, is very common. The equivalent ANSI standard forms are quite tedious but can be substituted if necessary.

ANSI standard FORTRAN restricts the assignment of values to formal parameters in a subroutine if two or more formal parameters are associated with the same actual parameter. The MMLE3 program does not abide by this restriction. The restriction is a quite conservative way of avoiding potential problems in optimizing compilers. The specific usage in MMLE3 is not prone to such problems. The violation of this restriction in the ANSI standard cannot be detected at compilation time and thus does not result in compilation errors.

Quotation marks are used in many subroutines to delimit Hollerith fields in FORMAT statements (never in DATA statements). Most compilers accept quotation marks or some other character in this role. A simple character translation program can be used to change the quotation marks to acceptable characters if necessary.

The use of NAMELIST input is the most serious potential problem of FORTRAN compatibility for MMLE3. Although NAMELIST input is nonstandard, it is available on many computer systems. If NAMELIST is not available, a substitute type of input will be necessary. If such a substitute does not embody many of the general features of NAMELIST, the program will become very cumbersome to use. In particular, because of the large numbers of variables and options available, it is desirable to

have default values that are used if no values are input. The user then needs to set only the values that are different from the defaults. It is also desirable to have the input variables identified by name instead of position, because variable names are much easier to remember than positions assigned on the input cards.

The program does not depend on memory being initialized to 0. However, matrix elements which might not be defined are tested to see if they contain the special flag value "TEST" (see sec. 3.1). Therefore, memory must not be initialized to infinite or indefinite values on computers that have such values.

## 2.2 Files

Input and output conventions vary from system to system; therefore, some flexibility must be built into any program that will be used on different computer systems. All input or output in MMLE3 uses the variables in common block FILES for unit numbers. The values of these variables are set in subroutine VARDEF. The unit numbers can be changed as desired for compatibility with various operating systems; for CDC systems, it will be necessary to change the file names in the program card to correspond to the unit numbers used.

Descriptions of the files used by the program are given in reference 1, section 3.2. On IBM systems, the following DD cards are samples of those necessary to define the files. It is assumed that a cataloged procedure is used that defines the card reader, card punch, and line printer files. The file numbers and record lengths in these samples correspond to the values currently used in the program.

```
//GO.FT02F001 DD SYSOUT=B


//GO.FT04F001 DD DISP=OLD,UNIT=2314,VOL=SER=volume,
//  DCB=(RECFM=VSB,LRECL=420,BLKSIZE=4204),DSN=name


//GO.FT07F001 DD DISP=NEW,UNIT=SYSDA,SPACE=(CYL,(10,2)),
//  DCB=(RECFM=VSB,LRECL=148,BLKSIZE=4444),DSN=INTERNAL


//GO.FT08F001 DD DISP=NEW,UNIT=SYSDA,SPACE=(CYL,(10,2)),
//  DCB=(RECFM=VSB,LRECL=192,BLKSIZE=3844),DSN=TOPLOT


//GO.FT09F001 DD DUMMY
```

or

```
//GO.FT09F001 DD DISP=(NEW,CATLG),UNIT=2314,VOL=SER=volume,
//  SPACE=(CYL,(5,1),RLSE),
//  DCB=(RECFM=VSB,LRECL=148,BLKSIZE=2964),DSN=THOUT


//GO.FT10F001 DD DISP=(NEW,CATLG),UNIT=2314,VOL=SER=volume,
//  SPACE=(CYL,(5,1),RLSE),
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),DSN=WTDATA
(DISP can also be OLD)


//GO.FT13F001 DD DISP=(NEW,CATLG),UNIT=2314,VOL=SER=volume,
//  SPACE=(CYL,(5,1),RLSE),DSN=PLOT
(on some systems replace FT13F001 by PLOTTAPE)
```

On CDC systems, no special control cards are needed except those to attach, request, or catalog any permanent files or tapes.

Core space can be conserved by lowering the I/O buffer sizes. The program uses the scratch files UT1 and UT2 (7 and 8, respectively) extensively. If the buffer size for these two files is lowered too much, execution time will increase. The buffer size of file UDATA (4) can also affect execution time if this file is long. The remaining files are used seldom enough that even very small buffer sizes will not affect the execution time. On CDC computers, buffer size is specified directly on the program card. On IBM computers, the total buffer size is determined by the block size and number of buffers specified on the DD cards.

## 2.3 Segmentation or Overlay

Efficient operation of the MMLE3 program requires the use of segmented or overlaid loading. The program requires over $200,000_8$ words of core to execute on a CDC computer if segmentation is not used. The use of segmented loading reduces the core requirements to approximately $54,000_8$ words. This large difference in core requirements results because the program was written to take advantage of segmentation.

The segmentation structure for MMLE3 is shown in figure 1. Each box represents a segment. The name assigned to the segment (usually the name of one of the

subroutines in the segment) is underlined at the top of the box. A brief description of the major functions accomplished in the segment follows.

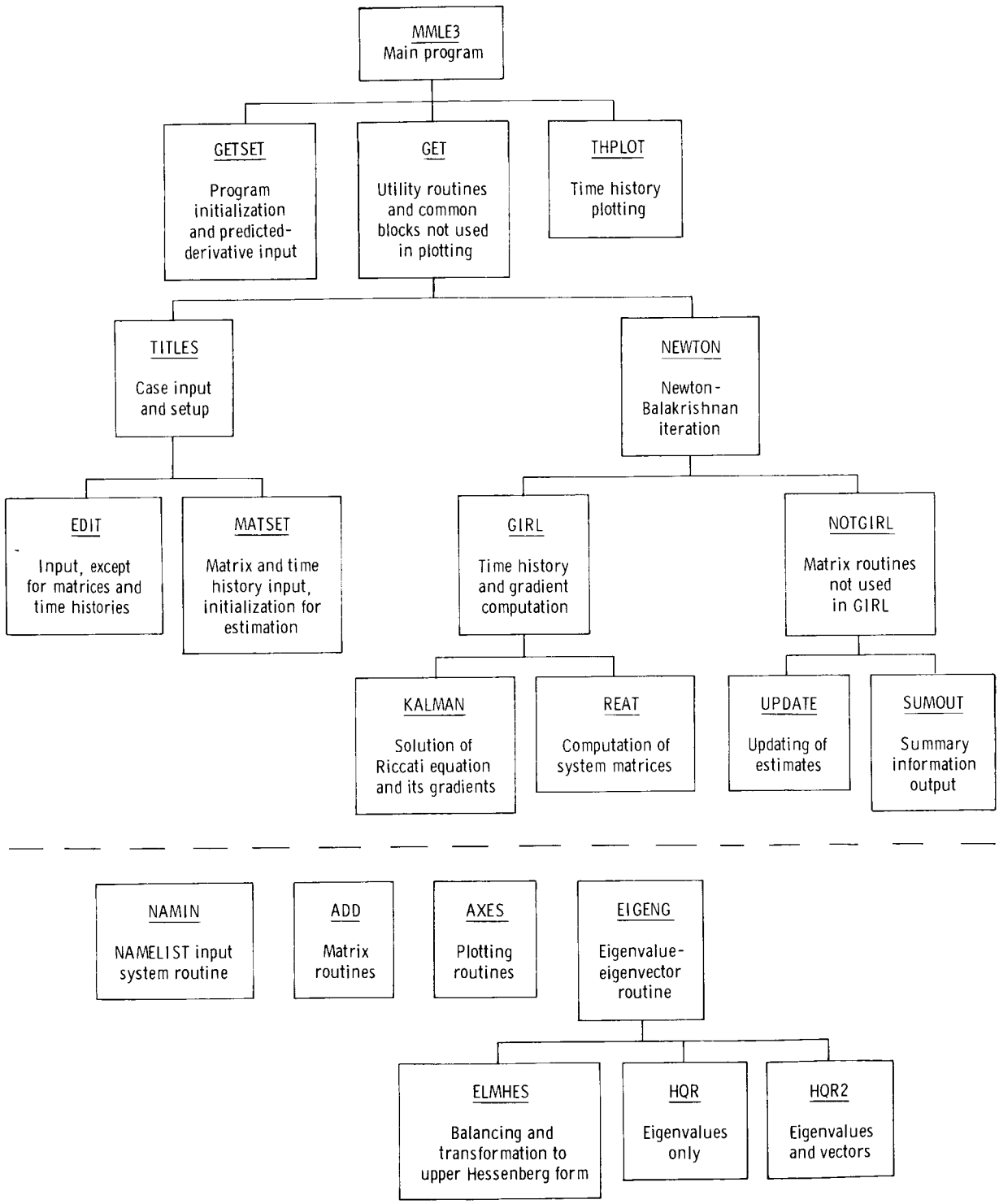



*Figure 1. Segmentation structure.*

The segmentation directives required to implement this structure on a CDC SCOPE 3.4 system are given below. The TREE and LEVEL directives define the basic structure. The INCLUDE directives assign subroutines to the appropriate segments, and the GLOBAL directives assign the common blocks. All of the subroutines and common blocks of the MMLE3 program itself are explicitly assigned by these directives. System routines and common blocks are not included, except for the CalComp plotter routines and the NAMELIST input routine NAMIN=. The name of the NAMELIST input routine will be different on different systems. On some systems, additional directives may be required to assign some of the system routines and common blocks to the root segment.

```
*      MMLE3 SEGMENTATION DIRECTIVES     31 JUL 80     RICH MAINE
*
          TREE  MMLE3-(GETSET,NOPLOTS,THPLOT)
NOPLOTS   TREE GET-(INTREE,OOTREE)
INTREE    TREE TITLES-(EDIT,MATSET)
OOTREE    TREE NEWTON-(GTREE,NGTREE)
GTREE     TREE GIRL-(KALMAN,REAT)
NGTREE    TREE NOTGIRL-(UPDATE,SUMOUT)
*
MMLE3     INCLUDE  MMLE3,ABEND,MIL
GETSET    INCLUDE GETSET,VARDEF,WTIN
GET       INCLUDE GET,GETLAB,HEAD,LOCATE,SET,SET1,SET2,SPIT,SPITER,ZOT1
GET       INCLUDE ZOT2
EDIT      INCLUDE EDIT,DIGIT,USERIN
MATSET    INCLUDE MATSET,ALLOW,AVERAG,COMPAT,CONIN,CONSTR,GVALVE,HARDC
MATSET    INCLUDE INTERP,LOADED,MATDEF,MATLD,MATNO,MTLOAD,MTSET,ONCE
MATSET    INCLUDE READTH,SETCON,THDATA,THMOD,VARY,WIDEF,WTTRAN
KALMAN    INCLUDE KALMAN,GRADK,GRADP,LYAPCB,RICATC
REAT      INCLUDE REAT,CALLAM,DIM1,DIM2,EAT,GRAD,GRADIC,INIT,OBSERV
REAT      INCLUDE SPIDIM,VMADD,MAKEL,MAKEM,MAKEVW,THOUT,UINIT
NOTGIRL   INCLUDE REDUCE,SINV,SSIMEQ
UPDATE    INCLUDE UPDATE,APRADD,BIAS,DFACT,FADJ,FLIMIT,MOVE,MVMULT
UPDATE    INCLUDE RESIDS
SUMOUT    INCLUDE SUMOUT,CRAMER,CRSET,ERRTHP,OUTPUN,PLOP
THPLOT    INCLUDE THPLOT,IHMSMS
*
          GLOBAL BUF,COM,DIMALL,FILES,HEADNG,INTEGR,MAXCON,MAXIM,MAXIMS
          GLOBAL MODCOM,OUTOPT,SIZE,TAPPOS,TOPLOT
          GLOBAL FLCOND,GFDEFS
GET       GLOBAL AVGCOM,BILIN,DETERM,DIMMAT,ECOM,FCOM,GICOM,GRSIZE
GET       GLOBAL ICOND,MAPCOM,MATRAT,MATRIX,OBSRV,PHICOM,SOFCOM,SUMSAV
GET       GLOBAL TOGIRL
GET       GLOBAL INERTS,INSTR,LONLAT
TITLES    GLOBAL INMAT,INOPT,INORD,MATIN,MATLAB,TODATA,UVCOM
MATSET    GLOBAL RECRD
NEWTON    GLOBAL AMCOM,DUMCOM,DUMVEC,ERLIST,GRADS,GRDCOM,KCOM,PBCOM
NEWTON    GLOBAL SUMCOM,XSUMS
GIRL      GLOBAL TOGRAD,BIASES
GIPL      GLOBAL GRAV
KALMAN    GLOBAL RICCOM
SUMOUT    GLOBAL CRMAT
*      FOR SCRATCH STORAGE AS NEEDED
          GLOBAL GLOBAL
*
          LEVEL
*
          TREE NAMIN
NAMIN     INCLUDE NAMIN=,IDIGIT,ROWCOL
*
          TREE ADD
ADD       INCLUDE ADD,ADDPAR,DMULT,GETP,GETPAR,IDENT1,INV,MAKE,MULT
ADD       INCLUDE MULTT,SMULT,SUB,SUMULT,SYM,TRANSP,UNSET,ZMULT,ZOT
*
          TREE AXES
AXES      INCLUDE AXES,LINES,PLTDAT,SCALE2,SYMBL4,TITPLT
AXES      INCLUDE FACTOR,NUMBER,SYMBOL
*
          TREE EIGENG-(ELMHES,HQR,HQR2)
ELMHES    INCLUDE ELMHES,BALANC,ELTRAN
HQR       INCLUDE HQR
HQR2      INCLUDE HQR2,BALBAK
*
          END
```

The statement GLOBAL GLOBAL in the above directives declares that the common block name GLOBAL will be placed in the root segment so that it can be accessed from any subroutine. The program as published does not have a common block named GLOBAL. This statement is provided in case the user needs an additional common block for some temporary modification. The user can define the common block GLOBAL as needed, and the above directives will insure its proper placement in the segmentation structure.

The following directives implement essentially the same structure on the CDC NOS 1.4 operating system. The routines not mentioned explicitly and all of the common blocks are automatically assigned by the system.

```
*         MMLE3 SEGMENTATION DIRECTIVES FOR NOS.   12 SEPT 80   RICH MAINE.
*
          COMMON
          TREE  MMLE3-(GETSET,NOPLOTS,THPLOT)
NOPLOTS   TREE GET-(INTREE,DOTREE)
INTREE    TREE TITLES-(EDIT,MATSET)
DOTREE    TREE NEWTON-(GTREE,NGTREE)
GTREE     TREE GIRL-(KALMAN,REAT)
NGTREE    TREE SINV-(UPDATE,SUMOUT)
*
GETSET    INCLUDE VARDEF,WTIN
GET       INCLUDE GETLAB,SET1,SET2,SPIT,ZOT1,ZOT2
MATSET    INCLUDE ONCE,MTLOAD,THDATA,MATDEF,COMPAT,ALLOW
UPDATE    INCLUDE APRADD,BIAS,DFACT,FLIMIT,MVMULT,RESIDS
KALMAN    INCLUDE GRADK
REAT      INCLUDE DIM2,GRAD,GRADIC,INIT,OBSERV,SPIDIM,VMADD,MAKEVW,THOUT
*
          LEVEL
*
          TREE NAMIN=
NAMIN=    INCLUDE IDIGIT,ROWCOL
*
          TREE ADD
ADD       INCLUDE ADD,ADDPAR,DMULT,GETP,GETPAR,IDENT1,INV,MAKE,MULT
ADD       INCLUDE MULT1,SMULT,SUB,SUMULT,SYM,TRANSP,UNSET,ZMULT,ZOT
*
          TREE AXES
AXES      INCLUDE AXES,LINES,PLTDAT,SCALE2,SYMBL4,TITPLT
AXES      INCLUDE FACTOR,NUMBER,SYMBOL
*
          TREE EIGENG-(ELMHES,HQR,HQR2)
ELMHES    INCLUDE BALANC,ELTRAN
HQR2      INCLUDE BALBAK
*
          END
```

If MMLE3 is to be run without segmentation or overlay, several small changes will reduce the amount of core required. First, subroutine WTIN should be removed from the MMLE3 program and run as a separate job; alternately, the dimensions in WTIN could be decreased. Second, the dimensions in subroutine THPLOT could be decreased, at the expense of an increased number of disk reads. Third, some of the large common blocks from different segments could be equivalenced. Fourth, subroutine HQR2 could be modified to include HQR as a subset.

## 2.4 EISPACK Routines

The MMLE3 program uses the subroutines described in reference 6 for obtaining the eigenvalues and eigenvectors of matrices. This subroutine package (EISPACK) is widely used and well documented. The EISPACK routines used by MMLE3 are

BALANC, BALBAK, ELMHES, ELTRAN, HQR, and HQR2 from release 2 of EISPACK. Listings of these EISPACK routines are included in supplement 1, in case the user's installation does not have the EISPACK release 2 library available. The listings include brief descriptions of the functions and usage of the routines.

The EISPACK routine BALANC contains a machine-dependent variable RADIX. This variable should be defined at card 64 to be the base of the machine floating point representation. For CDC machines, this value is 2 (as shown in the listing in supplement 1). On IBM machines, RADIX should be 16. RADIX is used to insure that the arithmetic in subroutine BALANC will be exact, with no rounding error.

Subroutines HQR and HQR2 contain a machine-dependent variable MACHEP. This variable should be defined at cards HQR.63 and HQR2.87 to be the smallest positive, floating point number which, when added to 1, gives a result not equal to 1. For CDC machines this value is $2^{-47}$, as shown in the listings in supplement 1. Different values would be used for other machines. MACHEP is used in convergence tests.

## 2.5 Plotting

High-resolution time history plots are needed to adequately evaluate the results from MMLE3. Line printer plots are usually inadequate. The MMLE3 program uses CalComp plotter software (ref. 4). If a CalComp plotter is not available, the routines calling the CalComp software must be changed. The only CalComp subroutines called are PLOTS, PLOT, FACTOR, SYMBOL, and NUMBER. The calls to the CalComp software are in the main program, subroutine THPLOT and subroutines AXES, LINES, PLTDAT, and SYMBL4 called by THPLOT.

If the plotting subroutines are rewritten, it is important to recall that MMLE3 stores labels four characters per word for machine generality. Therefore, each word must be treated separately. Subroutine SYMBL4 is an example of this type of treatment. On a computer that stores four characters per word, SYMBL4 is not needed because a direct call to SYMBOL will work as well. However, the direct call to SYMBOL would not be transportable to machines with more than four characters per word.

The functions of the CalComp subroutines will be described here so that the user can adapt the program to other plotting software.

### 2.5.1 PLOTS (BUF, NBUF, UPLOT)

Subroutine PLOTS initializes the plotting software. BUF is a scratch storage area used by the plotting software, and NBUF is the length of the BUF vector in words. Some CalComp implementations do not require a user-allocated buffer, in which case BUF need not be dimensioned. UPLOT is the FORTRAN device number assigned for the plot file. Some systems ignore UPLOT and assign a system-specified file name.

### 2.5.2 PLOT (X, Y, IPEN)

Subroutine PLOT moves the plotter pen to the position (X, Y) referenced to the current origin. IPEN should be ±2, ±3, or 999. If the magnitude of IPEN is 2, the pen is down during movement, thus drawing a line. If the magnitude of IPEN is 3, the pen is up during movement. If IPEN is negative, the origin is redefined to be the pen position after the move. The value of 999 for IPEN is a special call to close the plot file.

### 2.5.3 FACTOR (FACT)

Subroutine FACTOR enlarges or reduces subsequent plots. The size of subsequent plots is FACT times the "normal" size. Calls to FACTOR are not cumulative; e.g., a call with FACT=.25 followed by a call with FACT=.5 makes subsequent plots one-half of the original size, rather than one-eighth. The program assumes that the original plot size takes all values in inches.

### 2.5.4 SYMBOL (X, Y, HGT, I, ANGLE, N)

Subroutine SYMBOL draws text or symbols. There are two branches in subroutine SYMBOL, depending on the last argument, N.

If N is positive, N characters taken from the vector I (in A format) are drawn. X and Y are the starting coordinates of the lower left-hand corner of the first character, and ANGLE is the angle at which they will be drawn. If X or Y is 999, the characters are drawn starting at the previous pen location in the corresponding axis. HGT is the height of the characters drawn.

If N is negative, a single symbol is drawn, specified by the integer I. The list of symbols is given in reference 4. The interpretation of X, Y, HGT, and ANGLE is the same as when N is positive. If N is -1, the pen is up during the move to location (X, Y). If N is -2 or less, the pen is down. The symbols for I values from 0 to 13 are drawn centered at the location (X, Y); all other symbols treat (X, Y) as the lower left-hand corner of the symbol.

The value N=0 is not used by the MMLE3 program.

### 2.5.5 NUMBER (X, Y, HGT, A, ANGLE, N)

Subroutine NUMBER draws the value of a floating point number. X, Y, HGT, and ANGLE are identical to the corresponding arguments in subroutine SYMBOL. A is the floating point number to be drawn. N controls the format used. If N is greater than or equal to 0, N digits after the decimal point will be drawn. If N is -1, the integer part of A is drawn with no decimal point. Values of N less than -1 are not used by MMLE3. All numbers drawn are rounded rather than truncated values.

## 2.6 Date and Time

Calls to the DATE and TIME subroutines are provided to help in identifying printed output and plots. Naturally, these subroutines are machine specific. If the system does not supply these functions, dummy routines which return blanks can be used. The calls to these routines occur in subroutines TITLES and PLTDAT. Any other useful identifying information may be substituted in these places, if desired.

## 2.7 Assembly Language Routines

The MMLE3 program is coded entirely in FORTRAN. No assembly language routines are included in the listings of supplement 1. The program spends a considerable portion of its CPU time in three matrix multiplication routines—MULT, SUMULT, and ZMULT. These routines are quite short and simple. It is, therefore, worth considering the use of assembly language replacements for these three routines in order to decrease the program run time. On a CDC Cyber 70/73, assembly language (COMPASS) versions of these subroutines run in about one-half of the CPU time required for the FORTRAN routines. This results in a decrease in the overall program run time of about 20 percent. Assembly language replacements for other subroutines are not generally worth the effort.

Listings of the COMPASS routines are shown below. The macros ENTRYP and CALL used in these routines are also listed.

```
          IDENT MULT
*
          COMMENT MATRIX MULTIPLICATION C=A*B
*
          USE DATA
 MAX      BSS 1B
 II       BSS 1B
 JJ       BSS 1B
 MIX      BSS 1B
 J        BSS 1B
 KK       BSS 1B
*
          ENTRYP MULT
*
* SAVE FORMAL PARAMETERS
*
          SA2   A1+1B
          SA3   A2+1B
          BX6   X1
          BX7   X2
          SA6   ARGL1
          SA7   ARGL2
          BX6   X3
          SA6   ARGL3
          CALL GET,(O,MAX,II,JJ),ARGL1
          CALL GET,(O,MIX,J,KK),ARGL2
          CALL SET1,(O,MAX,II,KK),ARGL3
* COMPARE VALUES OF J AND JJ
          SA3   J
          SA4   JJ
          IX0   X4-X3
          ZR    X0,GOTO
          CALL ABEND
*
* INITIALIZE REGISTERS FOR LOOPS
*
 GOTO     SA1   ARGL1           A
          SA2   ARGL2           B
          SA3   ARGL3           C
          SB1   -1              -1
          SB4   B1              -1
          SB7   X4              JJ
          SA4   MAX
          SB2   X4              MAX
          SA4   MIX
          SB3   X4              MIX
          SA4   II
          SB5   X4+B1           II-1
          SA4   KK
          SX0   X4              KK
 LOOPIK   SB4   B4-B1           I=I+1
          SX7   X1+B4
          SB6   B0              J=0
          SX6   B0
 LOOPJ    SA4   X7              A
          SA5   X2+B6           B
          FX4   X4*X5           A*B
          FX6   X6+X4           +C
          SX7   X7+B2           STEP A
          SB6   B6-B1           J=J+1
          LT    B6,B7,LOOPJ
          NX6   X6
          SA6   X3+B4           =C
          LT    B4,B5,LOOPIK
          SB4   B1              I=0
          SX0   X0+B1           K=K+1
          SX2   X2+B3           STEP B
          SX3   X3+B2           STEP C
          NZ    X0,LOOPIK
          EQ RETURN
*
          END
```

```
          IDENT SUMULT
*
          COMMENT MATRIX MULT  C=(A*)*B + C LOWER TRIANGULAR
*         I LOOP RUNS BACKWARDS FROM II TO 1 TO SAVE A B REGISTER
*
          USE DATA
 MAX      BSS   1B
 KK       BSS   1B
 II       BSS   1B
 MIX      BSS   1B
 I        BSS   1B
 J        BSS   1B
 BLOC     BSS   1B
*
          ENTRYP SUMULT
*
* SAVE FORMAL PARAMETERS
*
          SA2   A1+1B
          SA3   A2+1B
          BX6   X1
          BX7   X2
          SA6   ALOC
          SA7   BLOC
          BX6   X3
          SA6   CLOC
          CALL GET,(0,MAX,KK,II),ALOC
          CALL GET,(0,MIX,I,J),CLOC
*
* INITIALIZE REGISTERS FOR LOOP
*
          SA1   MAX             X1=MAX
          SA3   MIX
          SA4   KK
          SA5   II
          SB7   X4              B7=KK
          IX6   X3*X5
          SB4   X6              B4=MIX*II
          SB3   X3              B3=MIX
          SB1   -1              B1=-1
          SA3   BLOC
          IX3   X3-X1           X3=ADDR(B)-MAX
          SB2   X3              B2=X3
          SA2   CLOC
          SA4   ALOC
          SX5   X5+B1              (II-1)
          IX2   X2+X5           X2=ADDR(C(II,1))
          IX0   X5*X1
          IX0   X0+X4           X0=ADDR(A(1,II))
          SB5   0
 LOOPIJ   SA5   X2+B5           C(I,J)
          BX6   X5
          SB6   B0
          SB2   X1+B2           STEP B COLUMN
 LOOPK    SA4   X0+B6           A(K,I)
          SA5   B2+B6           B(K,J)
          FX5   X4*X5
          FX6   X5+X6           +C
          NX6   X6
          SB6   B6-B1           K=K+1
          LT    B6,B7,LOOPK
          SA6   X2+B5           =C(I,J)
          SB5   B5+B3           STEP C COLUMN
          LT    B5,B4,LOOPIJ
          SB4   B4-B3           REDUCE J LIMIT
          SB5   B0
          IX0   X0-X1           STEP A COLUMN BACKWARDS
          SB2   X3
          SX2   X2-1            STEP C ROW BACKWARDS
          LT    B0,B4,LOOPIJ
          EQ RETURN
*
          END
```

```
          IDENT ZMULT
*
          COMMENT C=C +A*B TAKES ADVANTAGE OF STRUCTURE OF A
*
          USE DATA
 MAX      BSS   1B
 II       BSS   1B
 JJ       BSS   1B
 MIX      BSS   1B
 J        BSS   1B
 KK       BSS   1B
 SAVEC    BSS   1B
 ONE      DATA  1.0
*
          ENTRYP ZMULT
*
* SAVE FORMAL PARAMETERS
*
          SA2   A1+1B
          SA3   A2+1B
          BX6   X1
          BX7   X2
          SA6   ARGL1
          SA7   ARGL2
          BX6   X3
          SA6   SAVEC
          CALL GET,(0,MAX,II,JJ),ARGL1
          CALL GET,(0,MIX,J,KK),ARGL2
* COMPARE VALUES OF J AND JJ
          SA3   J
          SA4   JJ
          IX0   X4-X3
          ZR    X0,GOTO
          CALL ABEND
*
* INITIALIZE REGISTERS FOR LOOPS
*
 GOTO     SB1   -1              B1=-1
          SA2   MAX
          SB2   X2              B2=MAX
          SA4   KK
          IX4   X2*X4
          SB7   X4              B7=KK*MAX
          SA5   JJ
          SX5   X5+B1
          SA3   ARGL2
          IX7   X3+X5           X7=LOC(B(JJ,1))
          IX5   X2*X5
          SB5   X5              B5=(JJ-1)*MAX
          SA5   ,I
          SA2   X5+B1           A2=II-1
          SB6   B5+A2           B6=(JJ-1)*MAX + (II-1)
          SA3   SAVEC
          SB3   A2
          SX2   X3+B3           X2=LOC(C(II,1))
          BX0   X2              X0=X2
          SA3   MIX
          SB3   X3              B3=MIX
          SA1   ARGL1
          SA1   X1              A1=LOC(A(1,1))
          SA4   ONE
          NX1   X4
*
 LOOPIJ   SB4   B0              INITIALIZE IK
          SA3   A1+B6           AIJ
          ZR    X3,S40
          SA5   X7              INITIALIZE B (JK = JJ)
          IX4   X3-X1
          NZ    X4,LOOP2
*
 LOOP1    SA4   X2+B4           C(IK) FROM MEMORY
          FX6   X4+X5            C + B
          NX6   X6
          SA6   X2+B4           C TO MEMORY
          SB4   B4+B2           IK=IK+MAX
          SA5   A5+B3           JK=JK+MIX  AND B(JK) FROM MEMORY
          LT,B4,B7,LOOP1
*
 S40      SX2   X2+B1           DECREMENT ROW OF C
          SB6   B6+B1           DECREMENT ROW OF A
          GE    B6,B5,LOOPIJ
```

```
*
          S85    B5-B2          DECREMENT COLUMN OF A
          SB6    A2+B5
          SX7    X7+B1          DECREMENT ROW OF B
          BX2    X0             RESET ROW OF C
          GE     B5,B0,LOOPIJ
          EQ RETURN
*
 LOOP2    SA4    X2+B4          C(IK) FROM MEMORY
          FX6    X3*X5          A*B
          FX6    X6+X4           +C
          NX6    X6
          SA6    X2+B4          C TO MEMORY
          SB4    B4+B2          IK=IK+MAX
          SA5    A5+B3          JK=JK+MIX  AND B(JK) FROM MEMORY
          LT,B4,B7,LOOP2
          EQ     S40
*
          END
```

```
                    IDENT MACROS
                    STEXT
          *
          ENTRYP    MACRO NAME
                    ENTRY NAME
                    USE 0
          TRACE.    VFD 42/7L_NAME
                    VFD 18/NAME
          SAVEA0    BSS 1
          RETURN    SA5 SAVEA0
                    SA0 X5
          NAME      EQ  *+4000008
                    SX6 A0
                    SA6 SAVEA0
          ENTRYP.   EQU NAME
          NCALLS.   SET 0
                    ENDM
          *
          CALL      MACRO SUB,ARGL,ARGNAM
                    LOCAL NONAM
          NAMED     IFC NE,*ARGNAM**
                    SA1 ARGNAM
          ARGLDEF   IF -DEF,ARGNAM
                    USE ARGLISTS
          ARGNAM    BSS 0
          NAMED     ELSE
          ARGLDEF   IFC NE,*ARGL**
                    SA1 NONAM
                    USE ARGLISTS
          NONAM     BSS 0
          NAMED     ENDIF
                    IRP ARGL
                    VFD 60/ARGL
                    IRP
                    BSS7 1
                    USE *
          ARGLDEF   ENDIF
          NCALLS.   SET NCALLS.+1
          +         RJ =X_SUB
          -         VFD 12/NCALLS.,18/TRACE.
                    ENDM
          *
                    END
```

## 2.8 Small Computer Systems

This section discusses the implementation of the MMLE3 program on small computer systems. The program is oriented toward large systems with such software features as NAMELIST input and CalComp plotter routines. Scratch disk space is assumed to be abundant. Finally, the program structure is optimized for segmented or overlaid loading, without which the core requirements would be quite large.

The essential heart of the program, however, is fairly compact. By sacrificing some versatility and convenience features, the program can be reduced to the point that it will run on a minicomputer. Several suggestions for such reductions are made below. The specific reductions chosen will depend on individual requirements and capabilities. Several of the suggestions involve eliminating subroutines. This can be accomplished either by actually removing the subroutine and all calls to it, or by replacing the subroutine with a dummy consisting just of a return.

Several of the suggestions will be recognized as replacing overlaid loading by a less automated equivalent—dividing the program into separate jobs run in the same core.

### 2.8.1 Matrix Dimensions

The most obvious saving is in reducing the maximum matrix dimensions as discussed in section 3.2. The dimensions in the published program are larger than needed for many applications, particularly applications appropriate to minicomputers.

### 2.8.2 Predicted-Derivative Input

The predicted-derivative input routine WTIN is the first candidate for reduction. In a nonoverlaid load, WTIN is extremely wasteful of core. The simplest solution is to eliminate WTIN from the MMLE3 program. WTIN can be run (if it is needed at all) as a completely separate program simply by replacing the subroutine card with a program card and replacing the return with a stop or by writing a driver that does nothing but call WTIN. The only communication between WTIN and the rest of the program is the file UWT created by WTIN. This file can be saved on disk or copied to cards. The UWT file can also be easily created without using WTIN for many applications. Reference 1, section 4.2.2, describes this file.

An alternative to eliminating WTIN is to reduce the size of the 20 by 20 matrix in WTIN and/or store it in common blocks used elsewhere in the program for other purposes.

For larger reductions, the UWT file can be eliminated. Subroutines WTIN, WTDEF, and WTTRAN would then be unneeded, as well as the buffers and disk space for the file.

### 2.8.3 Plotting

Time history plots are by far the most useful "frill" of the MMLE3 program. Indeed, this frill is almost a necessity for analyzing real flight data. The plotting routines and associated system software use a large amount of core in a nonoverlaid environment. At the cost of some inconvenience, the time history plots can be created by a separate job.

To remove the time history plotting from the MMLE3 program, delete the subroutine THPLOT and the call to PLOT at card MMLE3.71. Subroutines AXES, LINES, PLTDAT, SCALE2, SYMBL4, TITPLT, and the CalComp routines are all called from THPLOT, and are thus also eliminated.

Subroutine THPLOT expects the time history data on the scratch file UT2 and information describing the plot options in common blocks COM, HEADING, INTEGR, SIZE, and TOPLOT. The values in blocks FILES and MAXIMS must also be defined. If THPLOT is run as a separate program, it is probably more convenient to redefine the options and information in the common blocks than to communicate it from MMLE3. The time history data of file UT2 can be stored on disk or tape. Alternately, these data could be punched on cards (in which case file UT2 is not needed in MMLE3). If disk or tape storage is unavailable and the number of cards required impractical, the data of file UT2 can be recreated by a driver routine for THPLOT by using the final estimated system matrices. This would require a moderate amount of coding most easily done by cannibalizing subroutine GIRL, throwing out parts used only for the gradients.

### 2.8.4 State Noise Option

If the state noise option is not needed, a significant amount of core can be saved. The state noise option cannot be run as a separate job like the wind-tunnel input or time history plotting, so removing this option constitutes a sacrifice of capability, not just of convenience. The state noise option is removed by eliminating subroutines FADJ, FLIMIT, KALMAN, and GRADK. The routines ADDPAR, GETPAR, GRADP, LYAPCB, MOVE, MULTT, RICATC, SSIMEQ, and TRANSP are called only from the four primary state noise routines (directly or indirectly) and can thus also be eliminated. Common block GRDCOM and cards 71 to 82 should be removed from subroutine GRAD. The routines EIGENG, BALANC, HQR, HQR2, ELMHES, ELTRAN, and BALBAK can be eliminated if the state noise option is removed and the computation of the determinant at cards RESIDS.56 to 62 is eliminated. Several matrices used only with state noise (F, FV, APRF, P, and KGAIN) could be eliminated along with short sections of code in the rest of the program, but such savings are not large.

### 2.8.5 Miscellaneous Files

Several files used by the program can be eliminated to lower disk utilization. Elimination of files UWT and UT2 is discussed in sections 2.8.2 and 2.8.3, respectively. File UTHOUT is an easily sacrificed frill written by routine THOUT. The punch file UPUNCH can also be dispensed with, in which case subroutine OUTPUN would be eliminated.

The file UDATA, read by subroutine READTH, is not used if the time history is read from cards. Alternately, the input time history file can be prepared beforehand, with times selected out and data corrections made, and attached directly as file UT1. In this case most of subroutine THDATA can be eliminated; only the averaging and the call to user routine AVERAG need remain. If absolutely no disk or tape storage is available, the time histories can be stored in core, but this uses a large amount of core even if one is very selective about the number of signals used. If there is a capability for having only single file on disk or tape, that file should be UT1.

### 2.8.6 HQR and HQR2

Subroutine HQR is the same as HQR2 without the eigenvector computation. Therefore, HQR can be eliminated by adding an argument to HQR2 to control skipping the eigenvector computations. The change is easy and saves a worthwhile amount of core if segmentation is not used. The two routines are separated in MMLE3 because it was desirable to use the EISPACK routines as published in reference 6, and because there is no penalty for the code duplication if segmented loading is used.

If the state noise option is removed, the routines HQR and HQR2 will normally also be removed and this question will be moot.

### 2.8.7 Minimum Program

If MMLE3 is to be run on a small minicomputer, it may be necessary to make further reductions than those described above. This section describes the reduction of MMLE3 to about the minimum usable size.

The actual derivative estimation is performed by subroutine NEWTON and the subroutines called by NEWTON. The rest of the program can therefore be dispensed with, except for the minimum needed to define quantities used by NEWTON. In particular, subroutines WTIN, TITLES, EDIT, MATSET, MTLOAD, THDATA, MATDEF, COMPAT, ALLOW, SUMOUT, THPLOT, and subroutines called only through these routines can be eliminated. Section 4 describes the general relationships of these routines. Figure 1 can also be helpful in checking which routines and common blocks can be eliminated in this process. Some type of minimal input routine will be needed to define the common block parameters used by NEWTON. In addition to the basic matrices and options, the data in common block DETERM must be defined. The unmodified MMLE3 program reads the data in a more convenient format, and then subroutine ALLOW creates the lists in block DETERM based on the input. This makes the program easier to use, but uses significantly more memory than reading the data directly in the format required by NEWTON.

A minimal size MMLE3 would also incorporate the previously discussed modification to eliminate the state noise option to reduce the size of the part of the program under subroutine NEWTON. A few more words could also be saved by eliminating the separate storage of the M suffix matrices in common block MATRAT. The M suffix matrices could be equivalenced to the matrices in block DIMMAT instead of stored separately.

# 3.0 MATRIX STORAGE

This section discusses the matrix storage conventions used by the MMLE3 program. Methods of changing maximum matrix dimensions are also shown.

## 3.1 Conventions

The MMLE3 program must be able to work with matrices of varying size without recompilation. The FORTRAN language is not well equipped to handle variable size matrices. A set of matrix storage conventions has been adopted to partially alleviate this problem. Figure 2 illustrates the conventions discussed in this section.

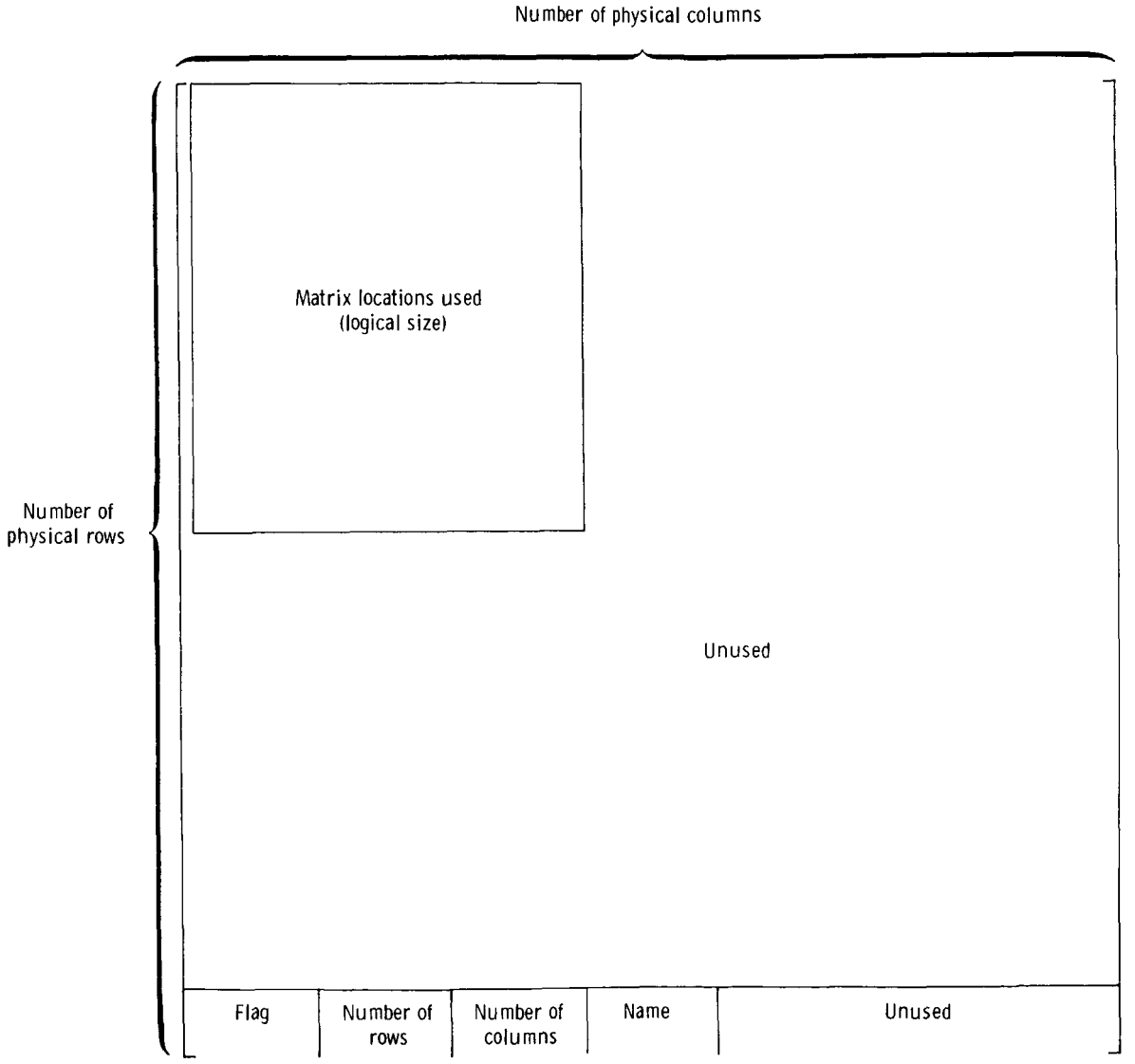



*Figure 2. Matrix storage conventions.*

The numbers in the FORTRAN dimension statement are referred to in this report as the physical dimensions of a matrix. These values control the amount of space allocated for the matrix by the FORTRAN compiler, and thus the largest usable dimensions of the matrix. Changes of the physical dimensions of a matrix require recompilation and are discussed in section 3.2.

Although the physical dimensions of a matrix are fixed, the program can store a matrix smaller than the physical dimensions as a partition of the physical matrix, ignoring the remaining locations. The size of this partition in use, referred to as the logical size of the matrix, may vary during program execution. No special conventions have been adopted for vectors (unless they are stored as a matrix with one logical column or row). The following conventions are used in MMLE3 to keep track of the physical and logical dimensions of matrices.

The last physical row of each matrix is reserved for information about the matrix; therefore, the physical number of rows of a matrix must always be at least one more than the logical number of rows. The physical number of columns of every matrix should be at least 4, because 4 locations in the last physical row are used.

The first location in the last physical row is used as a flag to find the row. The value used as a flag is defined in subroutine GETSET. Currently this value is the real number equivalent to the characters "TEST" left-justified and blank-filled. (On a CDC 60-bit computer, this value is approximately $3.149 \times 10^{92}$.) The flag value should be one that will not occur in any of the matrices stored. Note that the chances of any randomly picked value occurring in any given location are about 1 in $4 \times 10^{9}$ on a 32-bit computer and 1 in $10^{18}$ on a 60-bit computer. The odds in practice are even more unlikely for very large values such as the one used here, because the units used for most physical problems tend to keep values within a couple of orders of magnitude of 1. On a CDC computer, the value "negative indefinite" is a particularly good choice, because it can never be the result of an arithmetic operation. (Subroutines GET and GETP would have to be slightly modified if "negative indefinite" were used.)

The second and third locations in the last physical row are the number of logical rows and columns, respectively. These values are stored as real numbers. The fourth location in the last physical row contains the name of the matrix in Hollerith format. This name is used when the matrix is printed or punched. The name DONT is reserved for a special convention. The matrix print or punch routines (SPIT and PLOP) will not output matrices with a name of DONT. This convention simplifies control of which matrices will be output for a given case.

The functions of all the matrix manipulation routines are described in appendix A, but a few of the most important will be briefly described here. GETSET is an initialization routine that must be called before any of the other matrix routines. Routines SET, SET1, and SET2 set elements in the last physical row of a matrix. GET and GETP retrieve information about the physical and logical size of a matrix. UNSET deletes the flag from the first element of the last physical column of a matrix (this allows the same space to be reused as if it were a matrix with different physical dimensions). ABEND is an error routine; it intentionally causes an end-of-file error on the card reader file in order to get an error traceback.

## 3.2 Changing Maximum Dimensions

The matrix storage conventions allow the logical dimensions of a matrix to be changed within limits without recompilation. However, if a system to be analyzed exceeds the physical dimensions of a matrix, the program must be recompiled with larger dimensions. Conversely, if most of the analysis at an installation is done on very small systems, it may be prudent to recompile the program with smaller dimensions to avoid wasting core.

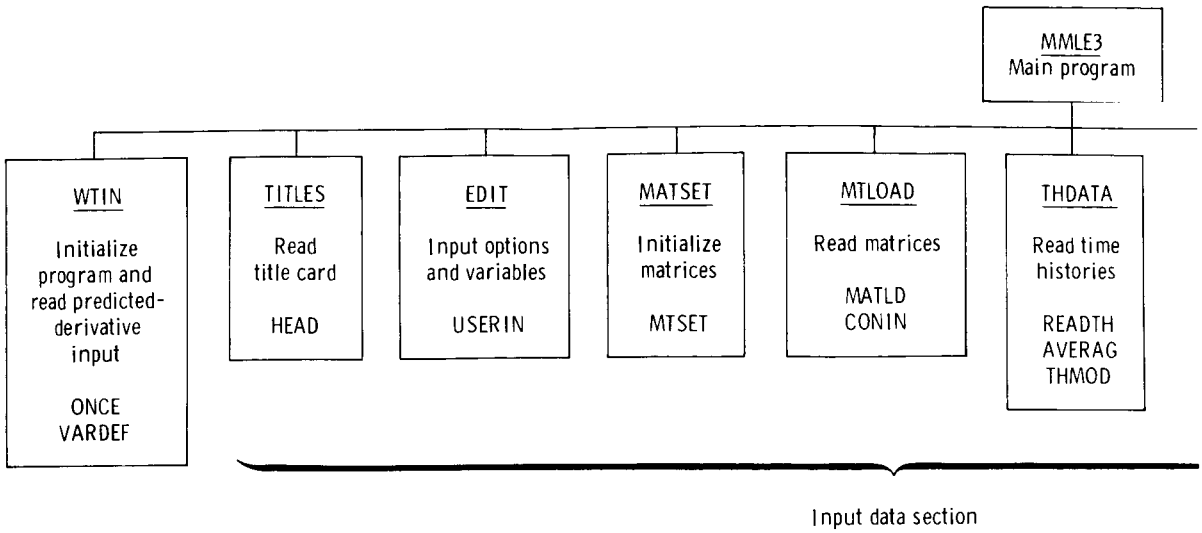
Changing the physical dimensions in a program as large as MMLE3 can be a major task. The coding of the MMLE3 program was specifically designed to simplify this task. Every card that must be changed in order to change the physical dimensions is in a common deck (see sec. 1). This convention minimizes the number of places to check for changes.

In order to simplify the task further, program COMPUN was written to punch out a complete set of the common decks with altered dimensions. The only input to COMPUN is a NAMELIST defining the desired matrix sizes. A listing of this program is shown in appendix C. The punched output from COMPUN can be used in the COMSUB program (appendix B) to create the actual FORTRAN code. Alternately, CDC UPDATE (ref. 2) can be used in place of COMSUB. Users of UPDATE will find it convenient to change COMPUN so that it punches out a correction set. This is done by changing each *COMDECK card in the data for COMPUN to an appropriate delete card; an ident card will also be needed at the beginning, and resequence cards may be desirable at the end (ref. 2).

# 4.0 PROGRAM STRUCTURE

This section gives a brief overview of the structure of the MMLE3 program. Because of the large number of subroutines, it is difficult to understand the operation of MMLE3 by studying the individual subroutines. A guide to how the subroutines fit together as a coherent whole is needed. With this guide, the user should be able to determine which subroutines are involved in any particular task; appendix A and supplement 1 can then be consulted to find details of how the task is accomplished.

Figure 3 constitutes a structural diagram of the major routines of MMLE3. Each box represents a task performed by one subroutine or a closely related group of subroutines. The primary subroutine name is underlined at the top of the box, and a brief description of the task follows. Important secondary subroutines called in the performance of each task are listed at the bottom of the box. Minor subroutines such as those for matrix manipulation are not included in the figure. The matrix subroutines are called throughout the program.



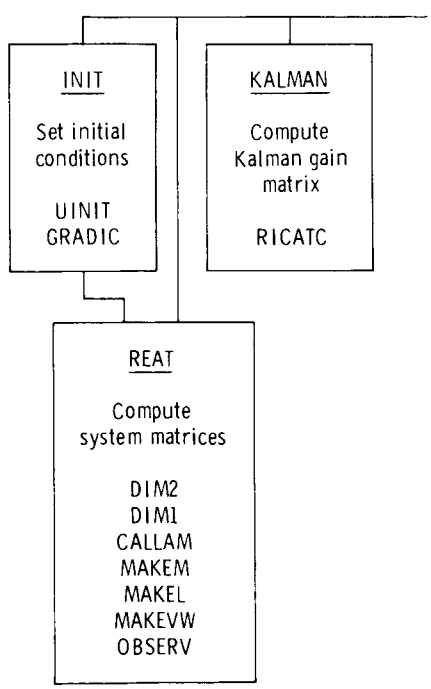
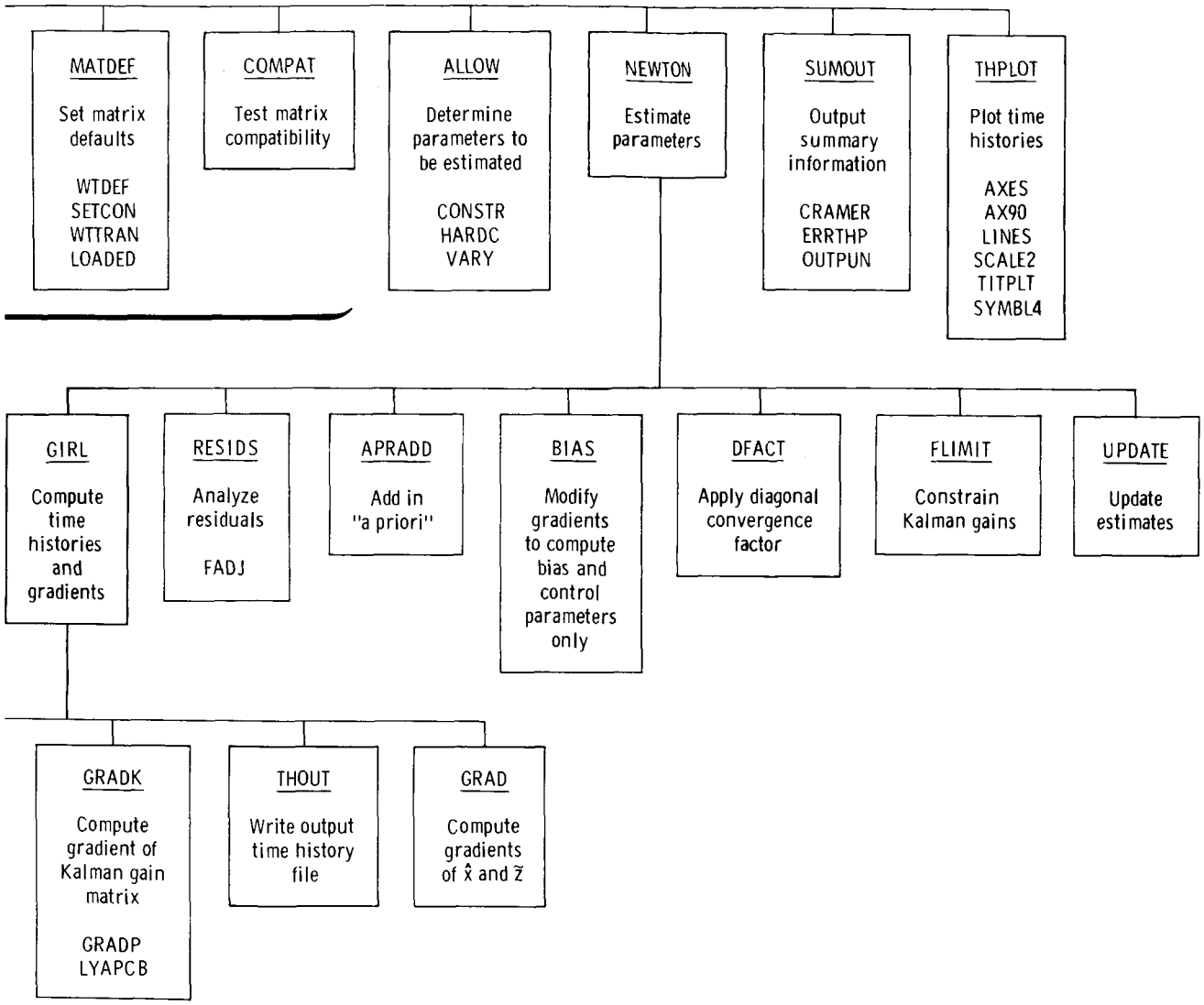




*Figure 3. Structure of*




*the MMLE3 program.*

4.0

*Dryden Flight Research Center*
*National Aeronautics and Space Administration*
*Edwards, Calif., November 28, 1979*

# APPENDIX A

## DESCRIPTIONS OF SUBROUTINES AND COMMON DECKS

### CONTENTS

# APPENDIX A

## DESCRIPTIONS OF SUBROUTINES AND COMMON DECKS

This appendix contains descriptions of the function and operation of each of the subroutines of the MMLE3 program. The variables in each of the common decks are also described. The user routines described are the standard aircraft routines (see ref. 1, sec. 4).

Supplement 1 contains microfiche listings of the subroutines and common decks; it should be consulted in conjunction with the descriptions in this appendix. The common decks are listed first. Next, all of the *CALL cards are listed separately. The separation of these cards from the FORTRAN listings is inconvenient, as cross-checking between the list of the *CALL cards and the FORTRAN listing is sometimes necessary, but a more convenient format of the listings is not easily obtained. The following pages contain lists of the UPDATE correction idents and deck names, completing the listings produced by UPDATE. The largest portion of supplement 1 consists of the FORTRAN listings and reference maps. The EISPACK routines are listed separately at the end of supplement 1, preceded by an UPDATE deck list for these routines.

### A.1 Common Decks

The common decks will be described in the order that they are shown on the listing of supplement 1. The common decks of the basic program are described first, followed by those of the standard aircraft routines. Following each description is a listing of the common deck.

#### A.1.1 Basic Program

The first common deck of the basic program is common deck HISTORY; the remaining common decks are in alphabetical order.

A.1.1.1 HISTORY. - Common deck HISTORY contains only comment cards. This common deck is intended to give a history of program modifications. For each modification to the program, one card briefly identifying the modification should be added to this common deck. Common deck HISTORY is called only from the main program.

```
*COMDECK HISTORY                                          HISTORY    1
C     MODIFICATION HISTORY:                               HISTORY    2
C     END OF MODIFICATIONS.                               HISTORY    3
```

A.1.1.2 AMCOM. - Common deck AMCOM contains the vector BCONST. This vector is defined by subroutine CALLAM; it contains the dimensionalization ratios from the M suffix matrices. For dependent variables in hard constraints, the corresponding element of BCONST is the constraint ratio times the dimensionalization ratio from the M suffix matrix.

```
*COMDECK AMCOM                                                    AMCOM      1
      COMMON /AMCOM/ BCONST( 50)                                  AMCOM      2
```

A.1.1.3 AVGCOM. - Common deck AVGCOM contains the average values of the measured states (ZAVG), controls (UAVG), and extra signals (EXAVG). The averages are computed for the entire, dimensioned lengths of these vectors, regardless of the vector size used in the system equations. The averages are taken over all of the maneuvers if multiple maneuvers are used. Standard deviations, maxima, and minima are also stored in AVGCOM. The relative position of the variables in this common block should not be disturbed, since subroutines THDATA and DIM1 depend on this order, treating ZAVG, UAVG, and EXAVG as a single, concatenated vector.

```
*COMDECK AVGCOM                                                   AVGCOM     1
      COMMON /AVGCOM/ ZAVG( 08),UAVG( 04),EXAVG( 20),ZSIG( 08),   AVGCOM     2
     -     USIG( 04),EXSIG( 20),ZMINM( 08),UMINM( 04),EXMINM( 20), AVGCOM    3
     -     ZMAXM( 08),UMAXM( 04),EXMAXM( 20)                     AVGCOM     4
```

A.1.1.4 BIASES. - Common deck BIASES contains the bias vector UOFF and YOFF defined by subroutine INIT. The role of these biases is discussed in reference 1, section 3.1.

```
*COMDECK BIASES                                                   BIASES     1
      COMMON /BIASES/ UOFF( 04),YOFF( 08)                         BIASES     2
```

A.1.1.5 BILIN. - Common deck BILIN contains the measured time history data for one time point. The observations (Z), controls (U), and extra signals (EXTRA) are obtained from the time history data file. The bias vector (ONES) is computed in subroutine GIRL (lines 61, 62, and 67) as described in reference 1, section 3.1. The logical variable TIMVAR is described in reference 1, section 3.3.8(20). Average values from common block AVGCOM are placed in Z, U, and EXTRA by subroutine DIM1 when USEAVG is TRUE. Subroutine GIRL controls USEAVG. These average values are used in the computation of the Kalman gain matrix and its gradients. The relative position of Z, U, and EXTRA in the common block should not be disturbed, or the average values will not be placed correctly.

```
*COMDECK BILIN                                                    BILIN      1
      COMMON /BILIN/ USEAVG,TIMVAR,Z( 08),U( 04),EXTRA( 20),ONES( 04) BILIN  2
      LOGICAL USEAVG,TIMVAR                                       BILIN      3
```

A.1.1.6 COM. - Common deck COM contains information about the time points used for a case. NCASE is the number of maneuvers being analyzed (ref. 1, sec. 3.3.8(1)). NPTS contains the number of time points in each maneuver, and NPTT is the total number of time points for all of the maneuvers. ITMSTS contains the total time in milliseconds of the first time point in each maneuver.

```
*COMDECK COM                                                          COM         1
      COMMON /COM/ NCASE,NPTT,NPTS(15),ITMSTS(15)                     COM         2
```

A.1.1.7 CRMAT. - Common deck CRMAT contains the Cramér-Rao bounds for the nondimensional matrices.

```
*COMDECK CRMAT                                                        CRMAT       1
      COMMON /CRMAT/ AC( 08, 07),BC( 08, 04),SC( 08, 04),RC( 08, 07),  CRMAT       2
     -     CC( 09, 07),DC( 09, 04),HC( 09, 04),EC( 09, 07),FC( 08, 07)  CRMAT     3
```

A.1.1.8 DETERM. - Common deck DETERM contains information describing the unknowns to be determined. The first half of subroutine ALLOW defines the variables in this common deck. NVAR is the total number of unknown locations. It equals the number of independent unknowns plus the number of constraints; thus, some locations may be counted more than once if they are dependent variables in more than one constraint. The remaining variables in the common block are vectors of length NVAR. IMAT contains the matrix number. The matrices (ref. 1, sec. 3.3.11(1)) corresponding to the numbers in IMAT are 1 = AN, 2 = BN, 3 = SN, 4 = RN, 5 = CN, 6 = DN, 7 = HN, 8 = EN, 9 = FN, and 10 = initial condition. IROW and ICOL contain the row and column numbers, respectively (for unknown initial conditions, the column number is ignored). ILOC gives the location of the associated gradient variable, and ACONST is the constraint ratio. Each independent variable will be associated with a unique gradient variable and the constraint ratio will be defined as 1. Dependent variables in hard constraints will be associated with the same gradient variable as the corresponding independent variable; the appropriate constraint ratio will be used. The information in common block DETERM is printed in the output under the heading "LOCATION INDICES."

```
*COMDECK DETERM                                                       DETERM      1
      COMMON /DETERM/ NVAR,IMAT( 50),IROW( 50),ICOL( 50),ILOC( 50),   DETERM      2
     -     ACONST( 50)                                                DETERM      3
```

A.1.1.9 DIMMAT. - Common deck DIMMAT contains the dimensional system matrices. They are defined in subroutine DIM1.

```
*COMDECK DIMMAT                                                       DIMMAT      1
      COMMON /DIMMAT/ A ( 08, 07),B ( 08, 04),S ( 08, 04),R ( 08, 07),  DIMMAT     2
     -     C ( 09, 07),D ( 09, 04),H ( 09, 04),E ( 09, 07)            DIMMAT      3
```

A.1.1.10 DUMCOM. - Common deck DUMCOM contains three matrices used for scratch storage in several subroutines.

```
*COMDECK DUMCOM                                                    DUMCOM    1
      COMMON /DUMCOM/ DUM( 08, 07),DUM2( 08, 07),DUM3( 08, 07)     DUMCOM    2
```

A.1.1.11 DUMVEC. - Common deck DUMVEC contains the vectors DUMX and DUM2 used for scratch storage.

```
*COMDECK DUMVEC                                                    DUMVEC    1
      COMMON /DUMVEC/ DUMX( 07),DUM2X( 07)                         DUMVEC    2
```

A.1.1.12 ECOM. - Common deck ECOM contains matrices of the state equation multiplied through by $R^{-1}$. The specific variables are RI = $R^{-1}$, RIA = $R^{-1}A$, RIB = $R^{-1}B$, and RIS = $R^{-1}S$. These matrices are defined by subroutine DIM2.

```
*COMDECK ECOM                                                      ECOM      1
      COMMON /ECOM/ RIA( 08, 07),RIB( 08, 04),RIS( 08, 04),RI( 08, 07)   ECOM      2
```

A.1.1.13 ERLIST. - Common deck ERLIST contains information on the convergence of the cost functional. NITER is the current interation number, and ERRVEC is a vector containing all of the previous values of the cost functional. The logical variable BLOWUP is set if the error becomes unreasonably large, indicating probable divergence. This causes iteration to stop prematurely. The dimension of ERRVEC can be changed without affecting any other program dimensions; this dimension limits the allowable number of iterations.

```
*COMDECK ERLIST                                                    ERLIST    1
      COMMON /ERLIST/ BLOWUP,NITER,ERRVEC(50)                      ERLIST    2
      LOGICAL BLOWUP                                               ERLIST    3
```

A.1.1.14 FCOM. - Common deck FCOM contains the state noise power spectral density matrix, F.

```
*COMDECK FCOM                                                      FCOM      1
      COMMON /FCOM/ F( 08, 07)                                     FCOM      2
```

A.1.1.15 FILES. - Common deck FILES contains the variables used for I/O unit numbers. These variables are discussed in reference 1, section 3.2. Their values are defined by subroutine VARDEF. If these values are changed, the program card (cards MMLE3.2 to 6) must also be changed.

```
*COMDECK FILES                                                     FILES     1
      COMMON /FILES/ UCARD,UPUNCH,UPRINT,UDATA,UT1,UT2,UTHOUT,UWT,UPLOT   FILES     2
      INTEGER UCARD,UPUNCH,UPRINT,UDATA,UT1,UT2,UTHOUT,UWT,UPLOT   FILES     3
```

A.1.1.16 GICOM. - Common deck GICOM contains the GGI matrix and related information. RSQ and FRSQ are the sample covariances of the raw and filtered residuals (fit errors), respectively. WRSQ and WFRSQ are the diagonal weighted fit errors. FREQCR, ITG, RLXG, and DIAGG are the input variables (ref. 1, sec. 3.3.8(23) to (25)) that control the residual filter and the $G$ determination. If the residuals are not filtered (FREQCR equals 0), FRSQ will be the same as RSQ. ERRFLT is the total filtered error sum, and SGNLS is the number of weighted signals. FC1 and FC2 are the constants computed to implement the residual filter.

```
*COMDECK GICOM                                                          GICOM    1
      COMMON /GICOM/ ITG,DIAGG,FREQCR,RLXG,FC1,FC2,ERRFLT,SGNLS,        GICOM    2
     -     GGI( 09, 08),RSQ( 09, 08),FRSQ( 09, 08),WRSQ( 08),WFRSQ( 08)  GICOM    3
      LOGICAL DIAGG                                                     GICOM    4
```

A.1.1.17 GRADS. - Common deck GRADS contains the gradients of the states and observations. GRADX is the gradient of the state and GRADY is the gradient of the observation. The residual is stored as an augmented column of GRADY. GRAD1 is a scratch matrix used in the computation of both gradients. The matrices are also used for scratch storage in subroutine FLIMIT.

```
*COMDECK GRADS                                                          GRADS    1
      COMMON /GRADS/ GRADX( 08, 35),GRADY( 09, 35),GRAD1( 09, 35)       GRADS    2
```

A.1.1.18 GRAD$. - Common deck GRAD$ contains vectors needed only in subroutine GRAD. XT12 is the average of the state at the beginning and end of a sample interval. XDT2 is the derivative of the state at the end of the interval; XDT12 is the average of the state derivatives at the beginning and end of the sample interval. The XDT's are computed ignoring state noise.

```
*COMDECK GRAD$                                                          GRAD$    1
      DIMENSION XDT2( 07),XDT12( 07),XT12( 07)                          GRAD$    2
```

A.1.1.19 GRDCOM. - Common deck GRDCOM contains the triply dimensioned array DK. The third index of the array corresponds to the list of unknowns that affect the $K$ matrix. For each of these unknowns, DK(•, •, $i$) contains the gradient of $K$ with respect to that unknown. DK is defined in subroutine GRADK.

```
*COMDECK GRDCOM                                                         GRDCOM   1
      COMMON /GRDCOM/ DK( 08, 08, 15)                                   GRDCOM   2
```

A.1.1.20 GRSIZE. - Common deck GRSIZE contains information about the size of the gradient vectors. JKMM1 is the number of independent unknowns. NK is the number of independent unknowns affecting the $K$ matrix.

```
*COMDECK GRSIZE                                                         GRSIZE   1
      COMMON /GRSIZE/ JKMM1,NK                                          GRSIZE   2
```

A.1.1.21 HEADNG. - Common deck HEADNG contains labels and titles. SIGLAB, XLAB, CONLAB, and EXLAB contain the labels for the observations, states, controls, and extra signals, respectively. Two words are allowed for each label. TITLE contains the title card for the case. ADATE and ATIME contain the date and time if available to the program. The relative position of SIGLAB, XLAB, CONLAB, and EXLAB should not be changed, as subroutine THPLOT depends on this relationship (card THPLOT.157).

```
*COMDECK HEADNG                                                    HEADNG    1
      COMMON /HEADNG/ TITLE(20),ADATE,ATIME,                       HEADNG    2
     -     SIGLAB(2, 08),XLAB(2, 07),CONLAB(2, 04),EXLAB(2, 20)    HEADNG    3
```

A.1.1.22 ICOND. - Common deck ICOND contains information about the initial conditions. USERIC and VARIC are input variables discussed in reference 1, section 3.3.8(26) and (27). VARICS is the Boolean sum of the elements of VARIC. DXIC is the initial condition increment estimated when elements of VARIC are TRUE.

```
*COMDECK ICOND                                                     ICOND     1
      COMMON /ICOND/ USERIC,VARICS,VARIC( 07),DXIC( 07)            ICOND     2
      LOGICAL USERIC,VARICS,VARIC                                  ICOND     3
```

A.1.1.23 INMAT. - Common deck INMAT contains information from a matrix header card during matrix input. ALAB is the matrix name, and IM is the corresponding matrix number defined by function MATNO. II and JJ are the numbers of rows and columns, respectively.

```
*COMDECK INMAT                                                     INMAT     1
      COMMON /INMAT/ ALAB,II,JJ,IM                                 INMAT     2
```

A.1.1.24 INOPT. - Common deck INOPT contains the logical variables CARD and TAPE, described in reference 1, section 3.3.8(2).

```
*COMDECK INOPT                                                     INOPT     1
      COMMON /INOPT/ CARD,TAPE                                     INOPT     2
      LOGICAL CARD,TAPE                                            INOPT     3
```

A.1.1.25 INORD. - Common deck INORD contains information about the order of signals on the time history data file. All of the variables are described in reference 1, section 3.3.8(6) and (7). The relative position of ZCHAN, UCHAN, and EXCHAN should not be changed, since subroutine THDATA depends on this relationship.

```
*COMDECK INORD                                                     INORD     1
      COMMON /INORD/ NREC,ZCHAN( 08),UCHAN( 04),EXCHAN( 20)        INORD     2
      INTEGER ZCHAN,UCHAN,EXCHAN                                   INORD     3
```

A.1.1.26 INTEGR. - Common deck INTEGR contains data required for the integration routine EAT. DT is the sample interval of the data (after any thinning). NEAT is an input variable described in reference 1, section 3.3.8(15).

```
*COMDECK INTEGR                                                    INTEGR      1
      COMMON /INTEGR/ DT,NEAT                                      INTEGR      2
```

A.1.1.27 KCOM. - Common deck KCOM contains the Kalman gain matrix, KGAIN, and the Riccati covariance matrix, P.

```
*COMDECK KCOM                                                      KCOM        1
      COMMON /KCOM/ P( 36, 07),KGAIN( 08, 08)                      KCOM        2
      REAL KGAIN                                                   KCOM        3
```

A.1.1.28 MAPCOM. - Common deck MAPCOM contains internal location maps. These maps are created by the second half of subroutine ALLOW and are used only in the state noise algorithm. Each vector in MAPCOM maps from a source list to a destination list. Each map vector in MAPCOM is the same length as the source list; each element in the map corresponds to an element in the source list. The value of each map element indicates the position in the destination list with which the corresponding source element is associated. A value of 0 for any map element indicates that the corresponding source element is not associated with any element of the destination list.

The variable names of the maps are all five characters, the first three of which are "MAP." The fourth and fifth characters indicate the source and destination lists, respectively. The letters used for the fourth and fifth characters are U, G, and K. U represents the complete list of unknowns, including independent unknowns plus constraints; the length of this list is NVAR (common block DETERM). G represents the complete list of gradient elements; the length of this list is JKMM1 (common block GRSIZE). Each independent unknown will correspond to one gradient element. K represents the list of gradient elements which affect the *K* matrix; the length of this list is NK (common block GRSIZE).

```
*COMDECK MAPCOM                                                    MAPCOM      1
      COMMON /MAPCOM/ MAPUK( 50),MAPKG( 15)                        MAPCOM      2
```

A.1.1.29 MATIN. - Common deck MATIN contains matrices that are only used in the input section of the program. The information from these matrices is put into other forms for use later in the program. The matrices in this common block are the V suffix, APR prefix, and hard constraint (HARD) matrices. All of these matrices are described in reference 1, section 3.3.11(3), (4), and (6), respectively.

```
*COMDECK MATIN                                                     MATIN       1
      COMMON /MATIN/  AV( 08, 07),BV( 08, 04),SV( 08, 04),RV( 08, 07),  MATIN  2
     -     CV( 09, 07),DV( 09, 04),HV( 09, 04),EV( 09, 07),FV( 08, 07),  MATIN  3
     -     APRA( 08, 07),APRB( 08, 04),APRS( 08, 04),APRR( 08, 07),      MATIN  4
     -     APRC( 09, 07),APRD( 09, 04),APRH( 09, 04),APRE( 09, 07),      MATIN  5
     -     APRF( 08, 07),HARD( 36,7)                                     MATIN  6
```

A.1.1.30 MATRAT. - Common deck MATRAT contains the M suffix matrices.

```
*COMDECK MATRAT                                                    MATRAT    1
      COMMON /MATRAT/ AM( 08, 07),BM( 08, 04),SM( 08, 04),RM( 08, 07),  MATRAT    2
     -     CM( 09, 07),DM( 09, 04),HM( 09, 04),EM( 09, 07)             MATRAT    3
```

A.1.1.31 MATLAB. - Common deck MATLAB contains the list of matrix labels and read flags. NMATS is the length of these lists (currently 31). LAB is the list of matrix labels. INFLAG is the list of read flags. Each element of INFLAG is 1 if the corresponding matrix has been read from cards; the element is 0 if the matrix has not been read. INFLAG is initialized to 0, and NMATS and LAB are defined in subroutine MATSET. INFLAG is then altered by cards MTLOAD.25 and CONIN.28.

```
*COMDECK MATLAB                                                    MATLAB    1
      COMMON /MATLAB/ NMATS,LAB(31),INFLAG(31)                     MATLAB    2
      REAL LAB                                                     MATLAB    3
```

A.1.1.32 MATRIX. - Common deck MATRIX contains the N suffix matrices, described in reference 1, section 3.3.11(1).

```
*COMDECK MATRIX                                                    MATRIX    1
      COMMON /MATRIX/ AN( 08, 07),BN( 08, 04),SN( 08, 04),RN( 08, 07),  MATRIX    2
     -     CN( 09, 07),DN( 09, 04),HN( 09, 04),EN( 09, 07)             MATRIX    3
```

A.1.1.33 MAXCON. - Common deck MAXCON contains the maximum dimensions MAXHRD and MAXSFT for the constraint matrices, HARD and SOFT (ref. 1, sec. 3.3.11(6) and (7), respectively). The variables MAXHRD and MAXSFT are defined by common deck VARDEF$. Maximum matrix dimensions are discussed in section 3 and in reference 1, section 2.

```
*COMDECK MAXCON                                                    MAXCON    1
      COMMON /MAXCON/ MAXHRD,MAXSFT                                MAXCON    2
```

A.1.1.34 MAXIM. - Common deck MAXIM contains some maximum matrix dimensions. NI, MAXTV, and MAXKV are defined by common deck VARDEF$. MAXX1 and MAXZ1 are defined as MAXX + 1 and MAXZ + 1 in subroutine VARDEF. Maximum matrix dimensions are discussed in section 3 and in reference 1, section 2.

```
*COMDECK MAXIM                                                     MAXIM     1
      COMMON /MAXIM/ MAXX1,MAXZ1,NI,MAXTV,MAXKV                    MAXIM     2
```

A.1.1.35 MAXIMS. - Common deck MAXIMS contains some maximum matrix dimensions. MAXX, MAXZ, MAXU, MAXB, and LEX are defined by common deck VARDEF$. LORD is defined as MAXZ + MAXU + LEX in subroutine VARDEF. Maximum matrix dimensions are discussed in section 3 and in reference 1, section 2.

```
*COMDECK MAXIMS                                                    MAXIMS    1
      COMMON /MAXIMS/ MAXX,MAXZ,MAXU,MAXB,LEX,LORD                 MAXIMS    2
```

A.1.1.36 MODCOM. - Common deck MODCOM contains the logical variable UMOD. UMOD is TRUE if user routines are used; otherwise, it is FALSE. UMOD is defined by the main program; the input to control this is described in reference 1, section 3.3.2.

```
*COMDECK MODCOM                                                    MODCOM    1
      COMMON /MODCOM/ UMOD                                         MODCOM    2
      LOGICAL UMOD                                                 MODCOM    3
```

A.1.1.37 OBSRV. - Common deck OBSRV contains matrices used for the computed observations and their gradients: ERIAC = $ER^{-1}A + C$, ERIBD = $ER^{-1}B + D$, ERISH = $ER^{-1}S + H$, ERI = $ER^{-1}$. These matrices are defined by subroutine DIM2.

```
*COMDECK OBSRV                                                     OBSRV     1
      COMMON /OBSRV/ ERIAC( 09, 07),ERIBD( 09, 04),ERISH( 09, 04),OBSRV     2
     -     ERI( 09, 07)                                          OBSRV     3
```

A.1.1.38 OUTOPT. - Common deck OUTOPT contains variables controlling output options. All of these variables are described in reference 1, section 3.3.8.

```
*COMDECK OUTOPT                                                    OUTOPT    1
      COMMON /OUTOPT/ PRINTX,PRINTY,PRINTD,PLOTEM,PUNCH,TEST,PLTMAX,OUTOPT     2
     -     ERRTH                                                   OUTOPT    3
      LOGICAL PRINTX,PRINTY,PRINTD,PLOTEM,PUNCH,TEST,ERRTH         OUTOPT    4
```

A.1.1.39 PBCOM. - Common deck PBCOM contains PB, the vector of changes in the coefficient estimates. The call to MVMULT at card NEWTON.66 defines PB. It may be modified by subroutine FLIMIT.

```
*COMDECK PBCOM                                                     PBCOM     1
      COMMON /PBCOM/ PB( 35)                                       PBCOM     2
```

A.1.1.40 PHICOM. - Common deck PHICOM contains the transition matrix and several products involving its integral. The transition matrix, PHI, is the exponential of $R^{-1}A\Delta t$. Call the integral of the transition matrix, $\psi$. Then, PSIB = $\psi R^{-1}B$, PSIS = $\psi R^{-1}S$, and PSI = $\psi R^{-1}$. All of these matrices are defined by subroutine REAT.

```
*COMDECK PHICOM                                                    PHICOM    1
      COMMON /PHICOM/ PHI( 08, 07),PSI( 08, 07),PSIB( 08, 04),     PHICOM    2
     -     PSIS( 08, 04)                                         PHICOM    3
```

A.1.1.41 RECRD. - Common deck RECRD contains one record from the time history data file. EOFTH can be set to TRUE by user routine READTH to indicate an end-of-file. T is the time in integer hours, minutes, seconds, and milliseconds. RECORD is the data for that time. The dimension of RECORD can be changed without affecting any other program dimensions.

```
*COMDECK RECRD                                                    RECRD    1
      COMMON /RECRD/ EOFTH,T(4),RECORD(100)                       RECRD    2
      LOGICAL EOFTH                                               RECRD    3
      INTEGER T                                                   RECRD    4
```

A.1.1.42 RICCOM. - Common deck RICCOM contains matrices used to compute the Kalman gain matrix and its derivatives. RIF = $R^{-1}F$, RIFRIF = $R^{-1}F(R^{-1}F)^*$, CTG = $C^*(GG^*)^{-1}$, and RIAP = $R^{-1}AP$. DUMXZ and DUMZX are scratch storage.

```
*COMDECK RICCOM                                                   RICCOM   1
      COMMON /RICCOM/ DUMXZ( 08, 08),DUMZX( 09, 07),              RICCOM   2
     -     RIAP( 08, 07),CTG( 08, 08),RIF( 08, 07),RIFRIF( 08, 07)RICCOM   3
```

A.1.1.43 SIZE. - Common deck SIZE contains the system vector sizes. MX is the length of the state vector, MZ the observation vector, MU the control vector, and MB the bias vector. These lengths are defined by subroutine COMPAT. They are discussed in reference 1, section 3.3.8(11) to (14).

```
*COMDECK SIZE                                                     SIZE     1
      COMMON /SIZE/ MX,MZ,MU,MB                                   SIZE     2
```

A.1.1.44 SOFCOM. - Common deck SOFCOM contains the matrix of soft constraints, SOFT.

```
*COMDECK SOFCOM                                                   SOFCOM   1
      COMMON /SOFCOM/ SOFT( 11,7)                                 SOFCOM   2
```

A.1.1.45 SUMCOM. - Common deck SUMCOM contains the SUM matrix, the second gradient of the cost functional. Only the lower triangular and diagonal parts of this symmetric matrix are stored. The first gradient is augmented as a last row or column. JKM is the logical dimension of the SUM matrix, i.e., the length of the gradient vector JKMM1 (common block GRSIZE) plus 1. In subroutine APRADD, the upper triangle of the matrix is used to form the *a priori* terms. In subroutine KALMAN, the SUM matrix is used for scratch storage. After the last iteration, the Cramér-Rao bounds and correlations are computed in the SUM matrix.

```
*COMDECK SUMCOM                                                   SUMCOM   1
      COMMON /SUMCOM/ JKM,SUM( 35, 35)                            SUMCOM   2
```

A.1.1.46 SUMSAV. - Common deck SUMSAV contains information about the *a priori* penalty function. WAPR and ITAPR are described in reference 1, section 3.3.8(22). DIAGON is the weighting vector of squared elements selected from the APR prefix matrices (ref. 1, sec. 3.3.11(4)). APRDIF is the vector of differences between the estimates and the *a priori* values. DIAGON is defined and APRDIF is initialized by subroutines ALLOW and VARY.

```
*COMDECK SUMSAV                                                    SUMSAV    1
      COMMON /SUMSAV/ DIAGON( 35),APRDIF( 35),WAPR,ITAPR           SUMSAV    2
```

A.1.1.47 TAPPOS. - Common deck TAPPOS contains information about the position of the time history data file. ITM is the last time read in total milliseconds; it is initialized to 0 in the main program. REW is used to request that subroutine READTH rewind the time history data file. REW is set to TRUE on the first point of a maneuver if the maneuver start time is less than or equal to the last time read. At all other times, REW will be FALSE. User subroutine READTH is responsible for checking REW and manipulating the data file as desired.

```
*COMDECK TAPPOS                                                    TAPPOS    1
      COMMON /TAPPOS/ ITM,REW                                      TAPPOS    2
      LOGICAL REW                                                  TAPPOS    3
```

A.1.1.48 THPLOT$. - Common deck THPLOT$ contains variables used only in subroutine THPLOT. If the program is run without segmentation or overlay, it may be desirable to shorten these vectors and store them in common blocks not used during the plotting (SUMCOM is the largest such block). Time from the maneuver start is stored in the vector TIME. Measured and computed observations are stored in X and XX, respectively. NCH is the number of observation time histories stored simultaneously, and NTPLT is 2 plus the maximum number of time points plotted (see sec. 3.2). XXX is equivalenced to X and XX; it is used to store up to 2 X NCH state, control, or extra signal time histories. Z, ZZ, and DC are used to read in each point of the time histories. VMINS and VMAXS contain the minimum and maximum values of the signals plotted. IPLT is a vector used to indicate which of the states, controls, and extra signals are to be plotted.

```
*COMDECK THPLOT$                                                   THPLOT$   1
      DIMENSION Z( 08),ZZ( 08),DC( 31),IPLT( 31),VMINS( 06),VMAXS( 06),  THPLOT$   2
     -     TIME(1202),XXX(1202, 06),X(1202, 03),XX(1202, 03)       THPLOT$   3
      EQUIVALENCE (X(1,1),XXX(1,1)),(XX(1,1),XXX(1, 04))           THPLOT$   4
C                                                                  THPLOT$   5
      NTPLT=1202                                                   THPLOT$   6
      NCH= 03                                                      THPLOT$   7
```

A.1.1.49 TODATA. - Common deck TODATA contains information needed to read the time history data file (channel numbers are in block INORD). STC and ETC are the requested maneuver start and end times in total milliseconds. All of the other variables are input variables described in reference 1, section 3.3.8(3), (8), (9), (29), and (32). The relative positions of ZBIAS, UBIAS, and EXBIAS in the common

block should not be changed, since subroutine THDATA depends on this relationship. The same applies to ZSCALE, USCALE, and EXSCAL.

```
*COMDECK TODATA                                                    TODATA    1
      COMMON /TODATA/ STC(15),ETC(15),THIN,PRINTI,MAXREC,          TODATA    2
     -     ZBIAS( 08),UBIAS( 04),EXBIAS( 20),ZSCALE( 08),USCALE( 04),  TODATA  3
     -     EXSCAL( 20)                                           TODATA    4
      INTEGER THIN,STC,ETC                                         TODATA    5
      LOGICAL PRINTI                                               TODATA    6
```

A.1.1.50 TOGIRL. - Common deck TOGIRL contains input variables used to control convergence. SNOISE is TRUE if the state noise algorithm is used; it is defined by subroutine ALLOW. The remaining variables are described in reference 1, section 3.3.8(16) to (19), and (21). Variables that control *G* determination and *a priori* are in common blocks GICOM and SUMSAV, respectively.

```
*COMDECK TOGIRL                                                    TOGIRL    1
      COMMON /TOGIRL/ BOUND,ERRMAX,FULL1,NOITER,DFAC,ITDFAC,SNOISE TOGIRL    2
      LOGICAL FULL1,SNOISE                                         TOGIRL    3
```

A.1.1.51 TOGRAD. - Common deck TOGRAD contains system vectors used in computing the time histories and gradients. The suffix 1 indicates a value at the beginning of the sample interval; the suffix 2 indicates a value at the end of the sample interval; the suffix 12 indicates the average of the values at the beginning and end of the sample interval. Names without suffixes indicate the end of the sample interval. XT is the predicted state, XH the corrected state, U the control, V and W the known forcing functions in the state and observation equations, Y the predicted observation, Z the measured observation, ZMY the residual, and ZMYFLT the filtered residual.

```
*COMDECK TOGRAD                                                    TOGRAD    1
      COMMON /TOGRAD/ XT1( 07),XT2( 07),XH2( 07),                  TOGRAD    2
     -     V1( 07),V2( 07),V12( 07),U1( 04),U2( 04),U12( 04),      TOGRAD    3
     -     Y( 08),ZMY2( 08),ZMYFLT( 08),W( 08)                     TOGRAD    4
```

A.1.1.52 TOPLOT. - Common deck TOPLOT contains input variables used to control plot scales and signals plotted. All of the variables except for RATIO are described in reference 1, section 3.3.8(35), (36), (37), and (39) to (44). RATIO is PLTFAC/2 if INCH (ref. 1, sec. 3.3.8(38)) is TRUE and PLTFAC/2.54 if INCH is FALSE. The relative positions of XMAX, UMAX, and EXMAX should not be changed, since subroutine THPLOT depends on this relationship. The same is true for XMIN, UMIN, and EXMIN.

```
*COMDECK TOPLOT                                                    TOPLOT    1
      COMMON /TOPLOT/ZMAX( 08),ZMIN( 08),XMAX( 07),UMAX( 04),EXMAX( 20),  TOPLOT  2
     -     XMIN( 07),UMIN( 04),EXMIN( 20),XPLOT( 07),NUPLT,NEXPLT,  TOPLOT    3
     -     TIMESC,RATIO,PAPER                                      TOPLOT    4
      LOGICAL XPLOT                                                TOPLOT    5
```

A.1.1.53 VARDEF$. - Common deck VARDEF$ defines the values of the physical matrix dimensions. This common deck is called only in subroutine VARDEF. Matrix physical dimensions are discussed in section 3 and in reference 1, section 2.

```
*COMDECK VARDEF$                                                    VARDEF$     1
      MAXX = 07                                                     VARDEF$     2
      MAXZ = 08                                                     VARDEF$     3
      MAXU = 04                                                     VARDEF$     4
      MAXB = 04                                                     VARDEF$     5
      LEX = 20                                                      VARDEF$     6
      NT = 35                                                       VARDEF$     7
      MAXTV = 50                                                    VARDEF$     8
      MAXKV = 15                                                    VARDEF$     9
      MAXHRD = 36                                                   VARDEF$    10
      MAXSET = 11                                                   VARDEF$    11
```

A.1.1.54 XSUMS. - Common deck XSUMS contains the averages and standard deviations of the corrected states for the last iteration. Cards 37 to 39 and 116 to 118 of subroutine GIRL accumulate the sums in XSUM and the sums of the squares in X2SUM. Then cards 36 to 42 of SUMOUT compute and print the averages in XSUM and standard deviations in X2SUM.

```
*COMDECK XSUMS                                                      XSUMS       1
      COMMON /XSUMS/ XSUM( 07),X2SUM( 07)                           XSUMS       2
```

## A.1.2 Standard Aircraft Routines

The common decks of the standard aircraft routines are described below in alphabetical order.

A.1.2.1 FLCOND. - Common deck FLCOND contains variables describing the flight condition. All of the variables except G are input variables described in reference 1, section 4.3.3(10) and (15) to (21). G is the acceleration of gravity, 32.172 feet/second$^2$ or 9.80665 meters/second$^2$, depending on METRIC (ref. 1, sec. 4.3.3(3)).

```
*COMDECK FLCOND                                                     FLCOND      1
      COMMON /FLCOND/ QBAR,V,THETA,PHI,ALPHA,MACH,PARAM,CG,G        FLCOND      2
      REAL MACH                                                     FLCOND      3
```

A.1.2.2 GFDEFS. - Common deck GFDEFS contains the default values for F and GGI defined by subroutine ONCE. FLON and FLAT are the longitudinal and lateral-directional defaults for F. GGILON and GGILAT are the corresponding defaults for GGI.

```
*COMDECK GFDEFS                                                     GFDEFS      1
      COMMON /GFDEFS/ GGILAT( 09, 08),GGILON( 09, 08),FLAT( 08, 07),GFDEFS      2
     -     FLON( 08, 07)                                          GFDEFS      3
```

A.1.2.3 GRAV. - Common deck GRAV contains the derivatives of the gravity terms in the $\dot{\alpha}$ and $\dot{\beta}$ equations. DGDT is the derivative of the gravity term in $\dot{\alpha}$ with

respect to $\theta$. DGDP is the derivative of the gravity term in $\dot{\beta}$ with respect to $\varphi$. These quantities are defined by subroutine MAKEL and subsequently used in both MAKEL and MAKEVW for the linearization of the gravity terms.

```
*COMDECK GRAV                                                          GRAV    1
      COMMON /GRAV/ DGDT,DGDP                                          GRAV    2
```

A.1.2.4 INERTS. - Common deck INERTS contains aircraft mass and geometry data. All of the variables except MASS and WTCG are input variables described in reference 1, section 4.3.3(2), (4), (5), (6), (8), and (9). MASS is the weight (ref. 1, sec. 4.3.3(7)) divided by the acceleration of gravity. WTCG is the reference center of gravity of the predicted derivatives. If a predicted-derivative file is not used (ref. 1, sec. 3.3.4), WTCG is undefined.

```
*COMDECK INERTS                                                        INERTS  1
      COMMON /INERTS/ IX,IY,IZ,IXZ,IXE,MASS,AREA,CHORD,SPAN,WTCG,SHIFT INERTS  2
      REAL IX,IY,IZ,IXZ,IXE,MASS                                       INERTS  3
      LOGICAL SHIFT                                                    INERTS  4
```

A.1.2.5 INSTR. - Common deck INSTR contains instrument positions and corrections. All of the variables except DCGFT are input variables described in reference 1, section 4.3.3(11) to (14). If SHIFT is TRUE, DCGFT is the distance of the flight center of gravity forward of the reference center of gravity in feet or meters. If SHIFT is FALSE or if there are no predicted data, DCGFT is 0.

```
*COMDECK INSTR                                                         INSTR   1
      COMMON /INSTR/ KALF,KB,XALF,XB,XAN,XAX,XAY,YALF,YB,YAN,YAX,YAY,  INSTR   2
     -     ZALF,ZB,ZAN,ZAX,ZAY,DCGFT                                   INSTR   3
      REAL KALF,KB                                                     INSTR   4
```

A.1.2.6 LONLAT. - Common deck LONLAT contains the logical variables LONG and LATR, described in reference 1, section 4.3.3(1). The program forces LATR to be .NOT.LONG.

```
*COMDECK LONLAT                                                        LONLAT  1
      COMMON /LONLAT/ LONG,LATR                                        LONLAT  2
      LOGICAL LONG,LATR                                                LONLAT  3
```

A.1.2.7 UVCOM. - Common deck UVCOM contains the vector UVAR described in reference 1, section 4.3.3(22).

```
*COMDECK UVCOM                                                         UVCOM   1
      COMMON /UVCOM/ UVAR( 04)                                         UVCOM   2
      INTEGER UVAR                                                     UVCOM   3
```

## A.2 Subroutines

The subroutines will be described in the order that they appear on the listing of supplement 1. The main program and the subroutines of the basic program are described first. Then follow the general utility routines for matrix manipulation, plotting, and time conversion. The standard aircraft user routines are described next, followed by the EISPACK routines. The flow of most of the routines is so simple that flow charts would be superfluous. Flow charts are given for those few subroutines for which they are useful. Refer to section A.1 for descriptions of the variables in common blocks.

### A.2.1 Basic Program

The major routines of the basic program are listed in the order of their use. Major routines are defined as the main program, including all subroutines called directly from the main program. Subroutine GIRL is so important that it is considered to be a major routine even though it is called from subroutine NEWTON, instead of from the main program. Each major routine is followed by the associated minor routines, listed in alphabetical order. The subroutine descriptions all refer to the cards at which the subroutines are called.

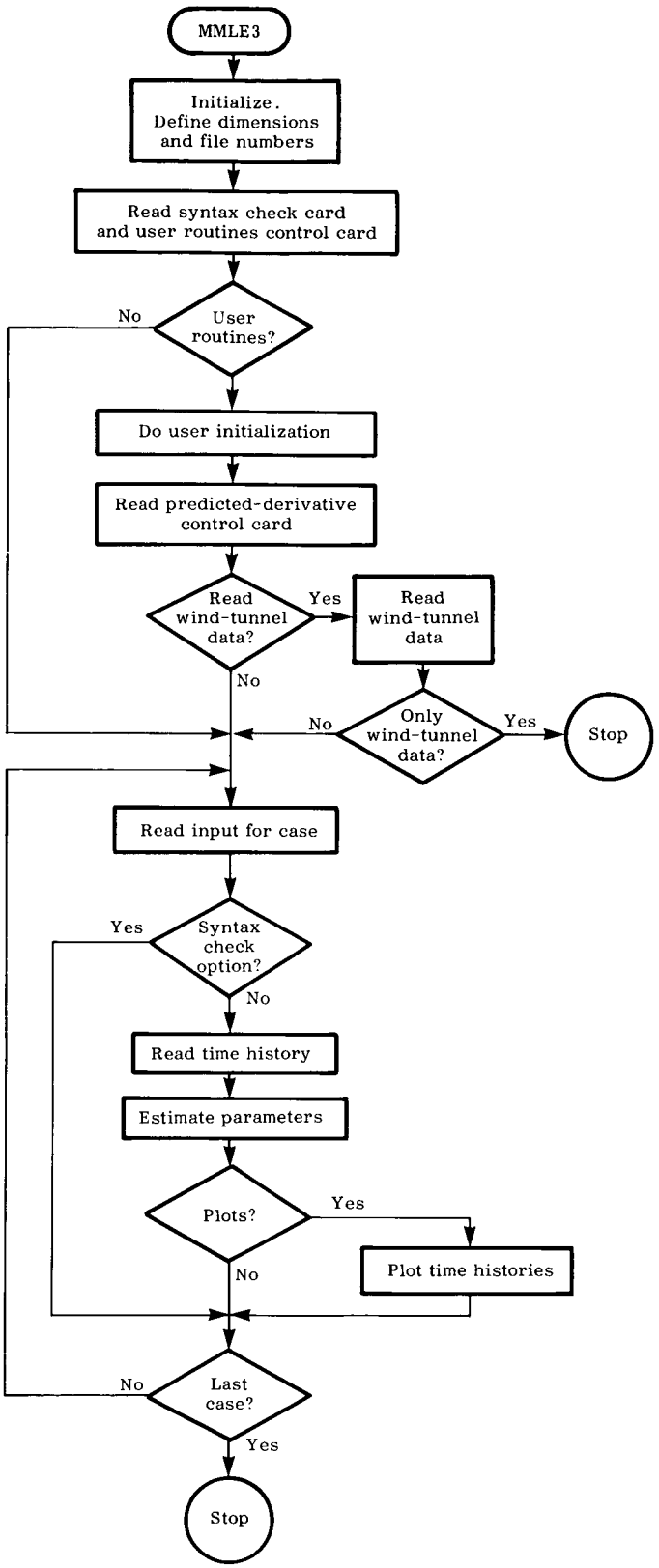
A.2.1.1 MMLE3. - MMLE3 is the main program. It contains the only call to common deck HISTORY, which describes the modification history of the program. The program card defines the files and buffer sizes. The program card and the variable definitions in common deck VARDEF$ must be changed in order to change file numbers. Some systems may not allow a program card, in which case it should be deleted.

The program first calls VARDEF to initialize file numbers and matrix physical dimensions. The variable ITM (last time read on the input time history file) is then initialized to 0. The variable PLTOPN indicates whether the plot file has been opened; it is initialized to FALSE.

Cards 37 to 43 read the syntax check card (ref. 1, sec. 3.3.1) and the user routines control card (ref. 1, sec. 3.3.2) and define the variable UMOD. The rest of the section is skipped if UMOD is FALSE. If UMOD is TRUE, subroutine ONCE is called for any user initialization, and the predicted-derivative control card (ref. 1, sec. 3.3.4) is read to define WTFILE. Depending on the predicted-derivative control card, user routine WTIN is then called to read predicted-derivative data.

Cards 56 to 70 loop until all cases have been analyzed (indicated by the variable LAST returned from subroutine MTLOAD). If the plot file was opened, subroutine PLOT is called after termination of the loop in order to close it.

A.2.1.2 VARDEF. - Subroutine VARDEF defines the variables describing the I/O file numbers and the matrix physical dimensions. Common deck VARDEF$ is included to define the basic matrix physical dimensions. Other dimensions are then computed from the basic ones. VARDEF calls subroutine GETSET to initialize the matrix routines and define the maximum physical dimension allowed. VARDEF is called at card MMLE3.31.


MMLE3
Initialize. Define dimensions and file numbers
Read syntax check card and user routines control card
User routines?
No
Do user initialization
Read predicted-derivative control card
Read wind-tunnel data?
Yes
Read wind-tunnel data
No
Only wind-tunnel data?
No
Yes
Stop
Read input for case
Syntax check option?
Yes
No
Read time history
Estimate parameters
Plots?
Yes
Plot time histories
No
Last case?
No
Yes
Stop

A.2.1.3 TITLES. - Subroutine TITLES reads the title card for a case (ref. 1, sec. 3.3.6). It also calls the DATE and TIME routines (sec. 2.6) to find the date and time of the run for identifying the printout. Subroutine HEAD is then called to print the page heading. TITLES is called at card MMLE3.56.

A.2.1.3.1 HEAD. - Subroutine HEAD prints a page heading, consisting of the title, date, and time. HEAD is called from several different routines.

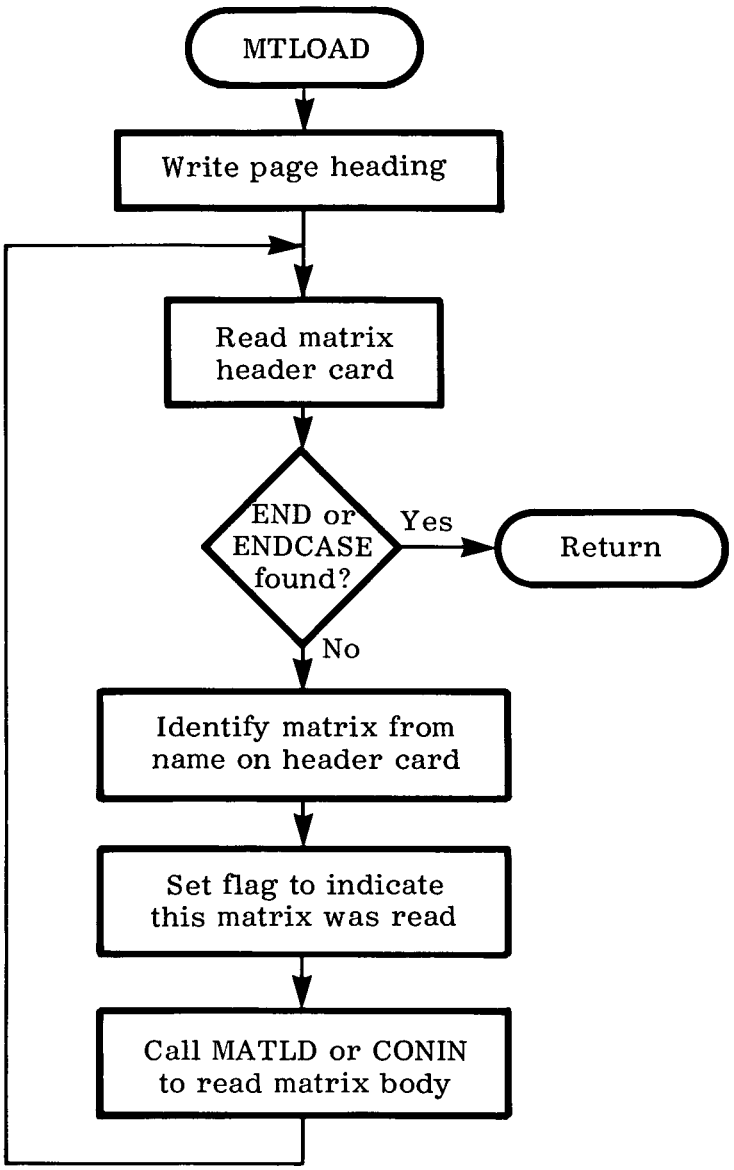
A.2.1.4 EDIT (WTFILE). - Subroutine EDIT reads the NAMELIST INPUT, the signal labels, and the time cards (ref. 1, secs. 3.3.8 to 3.3.10). The first section of code defines basic program default values. The user routine USERIN is called at card 117 to read any input for the user routines (ref. 1, sec. 3.3.7). The argument, WTFILE, is passed to subroutine USERIN to inform USERIN whether predicted-derivative data are available. USERIN may modify the basic program defaults. The next section of code reads the NAMELIST INPUT and makes some consistency checks between the options. The next sections print out scalar variables and options, read signal labels, and print vector variables and options. The last section of code reads and prints the requested maneuver times. EDIT is called at card MMLE3.57.

A.2.1.5 MATSET. - Subroutine MATSET initializes the input matrices as required by the standard matrix routines and defines their defaults. The N and V suffix and APR prefix matrices are initialized to 0 (except for RN, which is initialized to identity). Each element of the M suffix matrices is initialized to 1. The GGI matrix is initialized to 0, and the hard constraint (HARD) and soft constraint (SOFT) matrices are initialized to indicate no constraints. The matrix labels and input flags in common block MATLAB are also defined by MATSET. MATSET is called at card MMLE3.58.

A.2.1.5.1 MTSET (AN, AV, APRA, AM, IM, II, JJ). - Subroutine MTSET is used by MATSET to initialize a group of related matrices. IM is a code indicating which group of matrices is being initialized. II and JJ are the logical matrix dimensions to be used. For the matrices related to F (IM equals 9), an FM matrix is not defined, so the code initializing the M suffix matrices is skipped.

A.2.1.6 MTLOAD (LAST). - Subroutine MTLOAD controls the matrix input for each case. It reads the matrix header cards and determines which matrix is being read. It then sets the input flag (INFLAG) for that matrix and calls MATLD or CONIN to read the matrix body into the appropriate locations. Common block INMAT is used to pass information from the header card to MATLD and CONIN. Subroutine MTLOAD also detects the endcase card (ref. 1, sec. 3.3.12), which signals the end of the matrix input. The variable LAST is defined based on the endcase card and passed back to the main program as an argument. This variable flags the last case of a run. MTLOAD is called at card MMLE3.59.

A.2.1.6.1 CONIN (CON). - Subroutine CONIN reads the body of a constraint matrix input into the argument, CON, and prints out the matrix. Depending on the matrix header card, the matrix input flag (INFLAG) is reset to 0, allowing the constraints read in to supplement rather than replace any default constraints. This is discussed in section A.2.3.6 and in reference 1, section 3.3.11. CONIN is called at cards MTLOAD.85 and 87.


MTLOAD
Write page heading
Read matrix header card
END or ENDCASE found?
Yes
Return
No
Identify matrix from name on header card
Set flag to indicate this matrix was read
Call MATLD or CONIN to read matrix body

A.2.1.6.2 LOADED (AN). - Logical function LOADED determines whether a given matrix has been read in from cards. The function LOADED is intended for use by user routine MATDEF to determine whether the matrix defaults are used. The matrix itself is used as an argument, and TRUE is returned if the matrix was read from cards. LOADED first calls GETLAB to extract the matrix name from the matrix, and calls MATNO to find the corresponding matrix number. The vector of input flags (INFLAG) is then checked to see if that matrix was read. LOADED will return the value FALSE for constraint matrices that should supplement any default constraints rather than replace them. This is because subroutine CONIN has reset the corresponding input flag to 0. LOADED is called many times in the standard aircraft routines MATDEF and WTDEF.

A.2.1.6.3 MATNO (ALAB). - Function MATNO returns a matrix number, given its name as an argument. MATNO searches the list of names in common block MATLAB to find a matching name. The value returned is the index of the matching name. An error message is printed if the name is not found in the list. MATNO is called from several different routines.

A.2.1.7 THDATA. - Subroutine THDATA controls reading and processing of the input time history file UDATA. Subroutine READTH is called to do the actual manipulation of the data file so that the input format can be easily changed. THDATA handles the time searching, data scaling, printing, averaging, and associated tasks. For each maneuver (ref. 1, sec. 3.3.8(1)), THDATA defines the variable REW based on the last time point read and the requested start time. It then enters a search loop from cards 48 to 55 for the start time. The actual start time used for each maneuver is printed and stored for use by subroutine THPLOT.

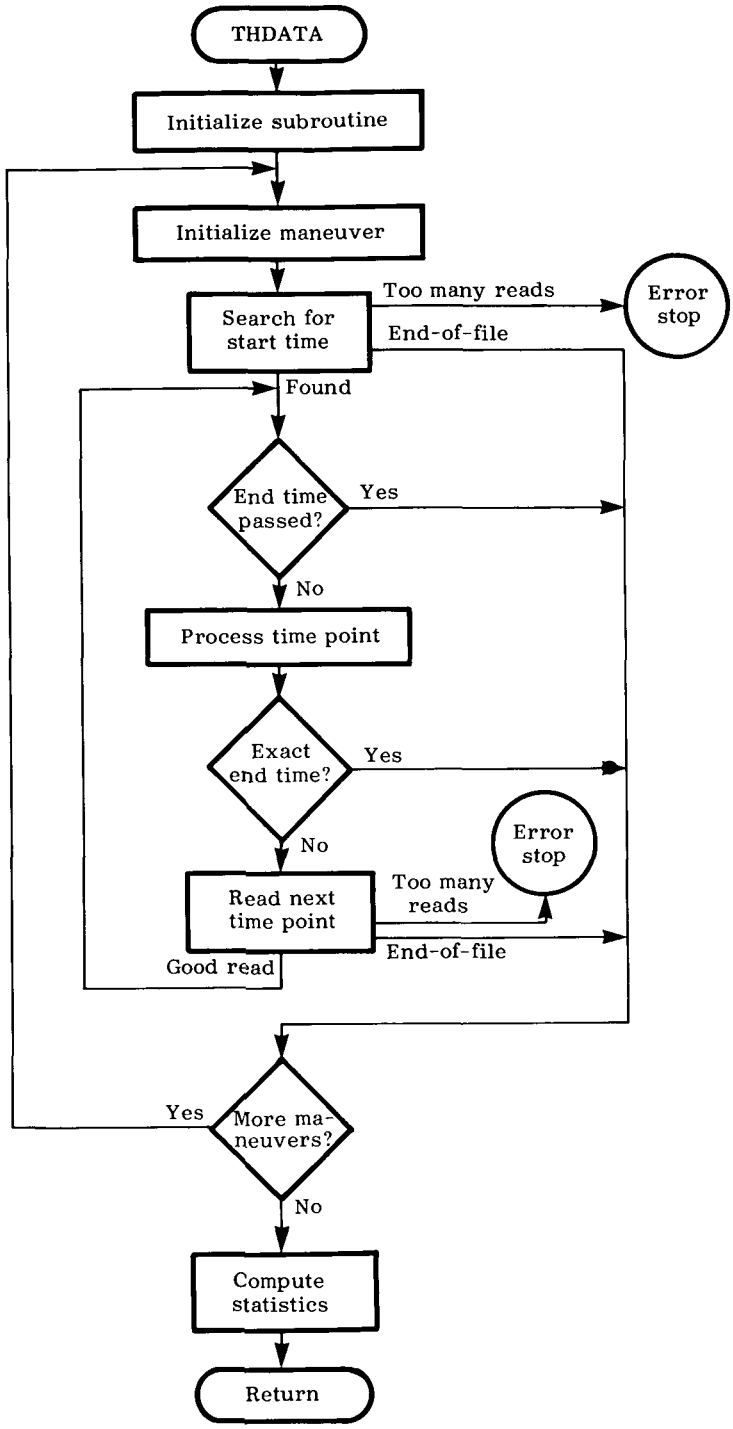
Cards 58 to 91 then process data until the maneuver stop time is found. The data are thinned if desired. At each time point, the requested data channels are extracted from the input record, and scale factors and biases are applied. Subroutine THMOD is called to modify or correct the data as desired. The data are then written to the scratch file UT1 and printed if desired. Based on the first and second thinned time points of the first maneuver, THDATA computes and prints the sampling rate of the data as described in reference 1, section 3.3.8(4).

After all of the maneuvers have been read, THDATA computes and prints the averages over the entire case of the observations, controls, and extra signals. Standard deviations, minima, and maxima are also computed, but not printed. User routine AVERAG is then called to allow the user access to these averages.

Subroutine THDATA stops with an error message if no time points are found in a requested interval. Another error check limits the number of calls to user routine READTH for each case; this is to guard against possible infinite loops caused by logic errors or omissions in READTH. End-of-file checks are made if user routine READTH has defined the variable EOFTH. End-of-file is not considered an error in itself, but may result in an error if no time points are found in an interval or if a following interval is requested.

THDATA is called at card MMLE3.61.

A.2.1.8 COMPAT. - Subroutine COMPAT sets the logical matrix sizes to compatible values and checks dimension limits. Cards 24 to 48 determine the matrix sizes




to be used as described in reference 1, section 3.3.8(11) to (14) and check these sizes against the dimension limits. Cards 50 to 78 set the appropriate logical sizes to be used for each of the matrices. Card 79 calls SYM to check GGI for symmetry. The starting nondimensional matrices and the F and GGI matrices are then printed out. COMPAT is called at card MMLE3.63.

A.2.1.9 ALLOW. - Subroutine ALLOW determines what parameters are allowed to vary. It also defines maps between different subsets of the parameters. This information is obtained from the V suffix matrices and constraint matrices. It is output from ALLOW in the DETERM and MAPCOM common blocks. ALLOW can be viewed as translating information from the input format (organized for ease of use) to the internal format (organized for compactness and efficiency).

ALLOW first initializes several vectors. It then calls VARY to define the independent unknowns in the N suffix matrices and the F matrix. Cards 41 to 50 add variable initial conditions to the list of independent unknowns as requested by the input vector VARIC. ALLOW then checks the dimension limit on the number of independent unknowns.

Subroutine CONSTR is called for preliminary processing of the hard and soft constraint matrices. Hard constraint processing is then completed by the call to HARDC. The remainder of the soft constraint processing is done later in subroutine APRADD.

Cards 59 to 63 determine whether the state noise algorithm will be used. Then cards 71 to 91 define the maps in common block MAPCOM that are used in the state noise algorithm. Cards 93 to 97 print the results from subroutine ALLOW.

ALLOW is called at card MMLE3.64.

A.2.1.9.1 CONSTR (CON). - Subroutine CONSTR does preliminary processing of the soft or hard constraint matrices. The matrix to be processed is the argument. The matrix names read in to define the constraints are translated into matrix numbers (see sec. A.1.1.8) by calls to subroutine MATNO. If constraint ratios were not specified on input, they are defined from the ratios of the starting values. Error messages are provided for unallowed matrix numbers and ill-defined constraint ratios. CONSTR is called at cards ALLOW.55 and 57.

A.2.1.9.2 GVALVE (IM, IR, IC). - Function GVALVE returns the present value of any coefficient in the N suffix or F matrices. Input arguments are the matrix number (see sec. A.1.1.8), row, and column (IM, IR, and IC, respectively). GVALVE is called at cards CONSTR.27 and 32.

A.2.1.9.3 HARDC. - Subroutine HARDC implements the hard constraints. Before HARDC is called, the lists in common block DETERM must have been defined for the independent unknowns. Subroutine CONSTR must have been called for preliminary processing of the hard constraint matrix. HARDC extends the list in common block DETERM to include the hard constraints. For each hard constraint, subroutine LOCATE is called to locate the independent variable of the constraint in common block DETERM. If the independent variable is found, the constraint information is added to the lists in DETERM. The variable NVAR, specifying the length of the lists in DETERM, is set to the number of independent unknowns during the execution of HARDC. This is so that the search in subroutine LOCATE will be restricted to the independent unknowns. At the end of subroutine HARDC, NVAR is set to the number of independent unknowns plus the number of active hard constraints. An error message is provided for exceeding the dimension limits in common block DETERM. HARDC is called at card ALLOW.56.

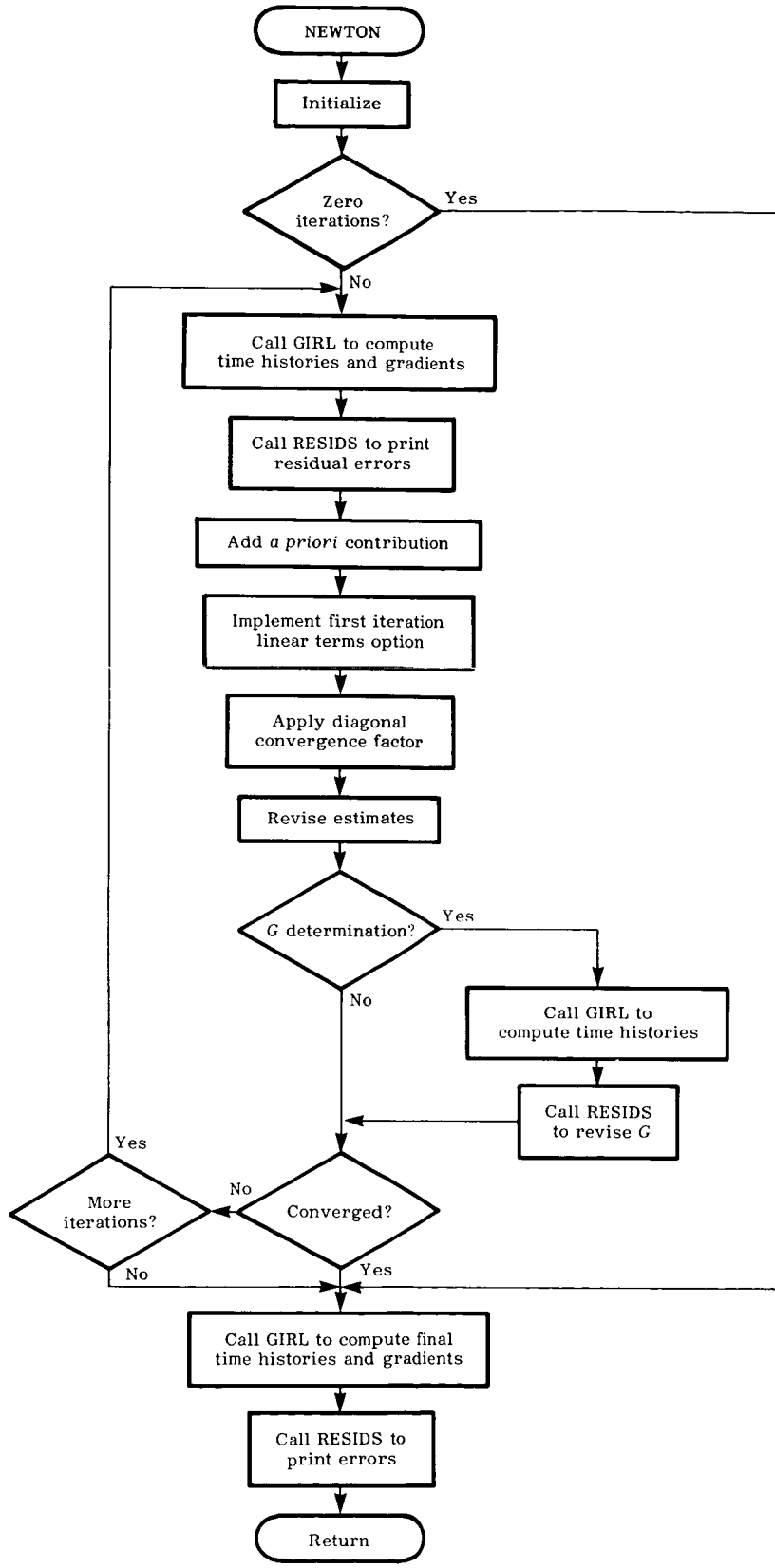
A.2.1.9.4 LOCATE (IM, IR, IC). - Function LOCATE returns the index of a variable in the list of unknowns. The input arguments are the matrix number, row, and column (IM, IR, and IC, respectively). If the given variable is not found in the lists in common block DETERM, the value 0 is returned. LOCATE is called at cards HARDC.19 and APRADD.20 and 21.

A.2.1.9.5 SETCON (CON, AI, IR, IC, AJ, JR, JC, FACT, IFZERO). - Subroutine SETCON defines a single default constraint in the hard or soft constraint matrices. SETCON is intended for use in user routine MATDEF. The first argument is the hard or soft constraint matrix (HARD or SOFT). AI, IR, and IC specify the location of the dependent variable in terms of matrix, row, and column, respectively. For convenience of use, the matrix itself (rather than the matrix name or number) is used for the second argument. Similarly, AJ, JR, and JC specify the matrix, row, and column of the independent variable. FACT is the constraint ratio. The last argument, IFZERO, is relevant only if the constraint ratio given is 0. If the constraint ratio given is 0 and IFZERO is FALSE, the constraint will be ignored as irrelevant. This is the outcome usually desired if the constraint ratio is a calculated quantity which can validly be 0. If the constraint ratio given is 0 and IFZERO is TRUE, the constraint will be retained. Subsequent processing by subroutine CONSTR will define the constraint ratio as the ratio of the starting values. SETCON is called several times in the standard aircraft routine MATDEF.

A.2.1.9.6 VARY (AN, AV, APRA, AR). - Subroutine VARY determines what coefficients in a given matrix are independently varying. The four arguments are the nondimensional starting matrix (AN) and its associated variation (AV), *a priori* weighting (APRA), and *a priori* value (AR) matrices. The corresponding V suffix matrix is searched for nonzero elements. For each such element, the matrix number, row, and column are stored in common block DETERM; corresponding *a priori* information is stored in common block SUMSAV. If there are no independent unknowns in the matrix, the matrix name will be defined as DONT; this will prevent printing the matrix every iteration. Subroutine VARY distinguishes between the starting values (AN) and the *a priori* values (AR), although the MMLE3 program does not currently preserve this distinction. VARY is called at cards ALLOW.32 to 40.

A.2.1.10 NEWTON. - Subroutine NEWTON controls the iteration for obtaining the maximum likelihood estimates. NEWTON is called at card MMLE3.65. The following are important variables used in the iteration control:

| Variable Name | Description |
|---|---|
| ITA | Number of iterations remaining until *a priori* is turned off plus one. When ITA is 0, *a priori* will either remain on or is already off. DFAC multiplication (ref. 1, sec. 3.3.8(21)) and *G* determination (ref. 1, sec. 3.3.8(24)) cannot start until ITA is 0. If convergence is achieved while ITA is nonzero, then ITA is set to 0, *a priori* is turned off, and iteration continues. |
| ITD | Number of iterations remaining with DFAC multiplication. DFAC multiplication does not start until ITA is 0. DFAC multiplication is not done the first iteration, regardless of ITA, unless FULL1 is TRUE. The convergence test is disabled while DFAC is used. *G* determination cannot start until ITD is 0. |


NEWTON
Initialize
Zero iterations?
Yes
No
Call GIRL to compute time histories and gradients
Call RESIDS to print residual errors
Add *a priori* contribution
Implement first iteration linear terms option
Apply diagonal convergence factor
Revise estimates
*G* determination?
Yes
No
Call GIRL to compute time histories
Call RESIDS to revise *G*
Yes
More iterations?
No
Converged?
No
Yes
Call GIRL to compute final time histories and gradients
Call RESIDS to print errors
Return

ITGGI — Number of iterations until *G* determination starts. If ITGGI is 0, either *G* determination has already started or will not be used (depending on ITG). ITGGI does not start counting iterations until ITA and ITD are 0. If convergence is achieved while ITGGI is nonzero and ITA is 0, then ITGGI is set to 0, and *G* determination starts.

CONVRG — Convergence indication. CONVRG is set to TRUE when the cost functional converges within the limit specified by BOUND. CONVRG is used to turn off *a priori*, turn on *G* determination, or stop iteration, depending on ITA and ITGGI.

The iteration loop is skipped by cards 23 to 29 if NOITER is 0 or if there are no unknowns. Cards 30 to 33 initialize iteration control variables. Cards 35 to 92 are the iteration loop.

GIRL is called at card 37 to compute the time history and the gradients of the cost functional. RESIDS is then called at card 39 to compute and print the residual powers and related quantities. Cards 42 to 44 determine if convergence has occurred; this determination may subsequently be changed, depending on the options in effect.

Cards 46 to 53 control the *a priori* option. If ITA (initialized at cards 30 and 31) is nonzero, it is decremented by 1 each iteration. When ITA reaches 0 or convergence occurs, *a priori* is turned off by setting WAPR and ITA to 0; the convergence flag is turned off so that iteration can proceed with *a priori* off. On subsequent iterations, ITA is 0, so this logic is skipped and *a priori* remains off. If ITA is initially 0, the code to turn off *a priori* will never be executed; the *a priori* weighting will thus remain at its initial value, which could be zero or nonzero.

Cards 55 and 56 control the call to BIAS to determine linear unknowns only. Note that the definitions of the convergence flag CONVRG on cards 42 and 86 check FULL1 or FULLIT to insure that an iteration on which BIAS is called will never be judged to have converged. Also card 58 checks FULLIT so that the call to DFACT will be skipped on iterations that BIAS is called. Finally, the call to FLIMIT at card 68 is not needed on iterations with BIAS because none of the linear unknowns affect the Kalman gain.

Cards 58 to 62 control the call to DFACT, which implements the diagonal convergence factor option. DFACT is not called until ITA is 0. DFACT is also not called on iterations for which BIAS was called. For each iteration that DFACT is called, ITD is decremented by 1; the convergence flag is forced to FALSE because convergence with DFACT is very slow and would often set the convergence flag before truly converging. After ITD reaches 0 (or if it starts at 0), subsequent iterations do not call DFACT.

Cards 64 and 65 invert the second gradient matrix and fill in its symmetric, upper triangular portion. Then cards 66 and 67 compute and print the changes in the parameter estimates. Card 68 calls FLIMIT if required to implement the inequality constraints described in reference 1, section 1.2.3. Cards 69 and 70 then revise and print the parameter estimates.

Cards 72 to 86 do the *G* determination if requested by a nonzero value of ITG. This code is not entered until both ITA and ITD are 0. After ITA and ITD are 0, cards 74 to 76 decrement ITGGI by 1 each iteration until ITGGI reaches 0 or convergence is obtained; either of these conditions triggers the start of *G* determination. The call to GIRL at card 82 computes the time history (no gradients are computed at this call). RESIDS is then called at card 85 to revise GGI based on the residuals. The convergence is tested at card 86. When *G* determination is active, each iteration has two steps: First, cards 37 to 70 revise all of the estimates except for GGI; and second, cards 82 and 85 revise the estimate of GGI.

Cards 89 to 91 exit the iteration loop if final convergence has been attained. Card 94 prints a warning message if the iteration limit is reached without attaining convergence.

GIRL is called at card 98 to compute the final time history and gradients (the second gradient will be required to compute the Cramér-Rao bounds). RESIDS, called at card 100, computes and prints the final iteration residual powers and related quantities.

A.2.1.10.1 APRADD. - Subroutine APRADD adds *a priori* terms to the first and second gradients. Soft constraints are implemented by APRADD as off-diagonal *a priori* terms. The upper triangular part of the SUM matrix is used to form the *a priori* weighting matrix; the diagonal elements of this matrix are stored in row JKM of SUM. APRADD is called at card NEWTON.53.

A.2.1.10.2 BIAS. - Subroutine BIAS causes only bias and control terms (linear terms) to be estimated in a particular iteration. The logical variable FULLIT controls the call to BIAS at card NEWTON.56. FULLIT is defined in turn at card NEWTON.55 depending on FULL1 and the iteration number. This results in BIAS being called for the first iteration unless FULL1 is TRUE.

A.2.1.10.3 DFACT. - Subroutine DFACT implements the diagonal convergence factor option (ref. 1, sec. 3.3.8(21)). It multiplies the diagonal elements of the second gradient by DFAC. The calling of subroutine DFACT is controlled by the input variable ITDFAC (ref. 1, sec. 3.3.8(21)). Cards NEWTON.58 to 62 implement the logic to call DFACT.

A.2.1.10.4 FADJ (AVG). - Subroutine FADJ adjusts *F* during *G* determination. The intent is to keep the Kalman gain matrix, *K*, unchanged as closely as reasonable. On entry, it is assumed that WFRSQ contains the diagonal elements of $\sqrt{GGI_{old}/GGI_{new}}$ and that the diagonal FRSQ contains the old GGI matrix elements; both of these quantities were computed in subroutine RESIDS. The input argument, AVG, is assumed to contain the logarithmic average of the elements in WFRSQ. The algorithm used is to multiply each row of *F* by the ratio of the corresponding diagonal element of $\frac{\text{ERIAC}^* \, GGI_{old} \, \text{DIAG(WFRSQ)} \, \text{ERIAC}}{\text{ERIAC}^* \, GGI_{old} \, \text{ERIAC}}$ where only diagonal elements of $GGI_{old}$ are used and DIAG(WFRSQ) is the diagonal matrix formed from WFRSQ. If any element of the above numerator is 0, the corresponding row of *F* is instead multiplied by AVG. Only independently varying elements of *F* will be changed by subroutine FADJ as controlled by the loop from cards 33 to 41. FADJ is called at card RESIDS.95.

A.2.1.10.5 FLIMIT. - Subroutine FLIMIT constrains certain diagonal elements of the closed loop gain $K(ER^{-1}A + C)$ to be less than or equal to 1. The algorithm and reasons for this constraint are discussed in reference 1, sections 1.1.2 and 1.2.3. GRADX, GRADY, and GRAD1 are used for scratch storage in this routine. The subroutine first forms the gradients of these diagonal elements in GRADX. It then determines which diagonal elements of $F$ are varying, using DUM2X to flag such elements. Constraints are only made on diagonal elements of $K(ER^{-1}A + C)$ corresponding to unknown diagonal elements of $F$. Cards 64 to 76 compute the linearized extrapolation of the diagonal elements of $K(ER^{-1}A + C)$, adding the current value plus the gradient times PB (the vector of proposed coefficient changes). The rows of GRADX corresponding to constraints that are satisfied (or the elements that are not constrained) are deleted, and the remaining rows are compressed into the first JJ rows of GRADX. Corresponding elements of DUMX are filled with the amount by which the constraint is exceeded. If no constraints are exceeded (JJ equals 0) the subroutine is done. Otherwise, cards 79 to 89 modify PB to lie on the constraint boundary, approximated by local linearization. FLIMIT is called at card NEWTON.68.

A.2.1.10.6 RESIDS (GIIT). - Subroutine RESIDS does computations based on the sample residual power. It first computes the filtered and unfiltered sample residual powers from the accumulated sums at cards 21 to 27. Cards 28 to 37 then eliminate the effect of any unweighted signals; this is needed so that such residuals do not affect the inverse and determinant of the residual power. Next, cards 40 to 54 compute and print the weighted errors for the filtered and unfiltered residuals. The cost functional (unfiltered weighted error sum) is placed in ERRVEC(NITER) and the filtered weighted error sum is placed in ERRFLT. WRSQ and WFRSQ are the diagonal elements of the unfiltered and filtered weighted errors, respectively. Cards 56 to 62 compute and print the log determinant of the unfiltered residual power using the product of the eigenvalues.

The remainder of the subroutine does the $G$ determination. The subroutine argument, GIIT, determines at card 66 whether this code is bypassed. The old GGI matrix is saved in FRSQ. Cards 68 to 72 move the unfiltered sample residual power matrix (or its diagonal elements only) into the GGI matrix and then invert it. This is the preliminary value of the new GGI matrix. Cards 73 to 92 then apply relaxation to further revise GGI; cards 80 to 91 compute WFRSQ as will be required by subroutine FADJ. After symmetrizing and printing the new GGI matrix at cards 93 and 94, subroutine FADJ is called at card 95 if the state noise algorithm is used. FADJ will adjust F to compensate for the GGI change.

RESIDS is called at cards NEWTON.39, 85, and 100.

A.2.1.10.7 SPITEM. - Subroutine SPITEM prints the N suffix matrices and the F matrix. Subroutine VARY controls which matrices are printed by changing some of the matrix names to DONT. Subroutine SPIT will ignore any matrices which have DONT as a name. The call to SPITEM at COMPAT.82 occurs before VARY is called; therefore, all of the matrices will still have their proper names and thus will be printed. The call to SPITEM at NEWTON.70 is subsequent to VARY; therefore, some (or all) of the printouts may be omitted.

A.2.1.10.8 UPDATE. - Subroutine UPDATE updates the parameter estimates. PB contains the vector of parameter changes to be made. Cards 43 and 44 keep track of the total change from the *a priori* value for use in subroutine APRADD. UPDATE is called at card NEWTON.69.

A.2.1.11 GIRL (DOGRAD, LASTIT). - Subroutine GIRL computes time histories and, optionally, gradients. The argument DOGRAD controls whether gradients are computed; the argument LASTIT controls various output options used only on the last iteration. Cards 31 to 63 do initialization before entering the case and time loops. Average dimensional matrices are computed by REAT or DIM2 if needed. If the time varying option is not used or if test output is requested, the average dimensional matrices and transition matrices are required, so REAT is called. If REAT is not called and the state noise algorithm is used, DIM2 is called to compute the matrices required for computation of the Kalman gain matrix and its gradient by subroutines KALMAN and GRADK. Cards 50 to 56 test if GGI is diagonal. Cards 66 to 76 initialize each maneuver and do any output required for the first time point of the maneuver. Subroutine INIT is called to define the time history initial condition, and GRADIC defines the gradient initial condition.

The time loop goes from cards 78 to 134. The measured data are read from scratch file UT1 and, if TIMVAR is TRUE, REAT is called to recompute the dimensional matrices at each time point. The predicted response is computed at cards 82 to 95. The filtered and unfiltered residuals are then computed and summed. The unfiltered residual is also stored as an augmented column to the GRADY matrix for convenience in computing the cost functional gradients. If the state noise algorithm is used, the corrected responses are then computed. The gradients are computed at cards 123 to 129. Subroutine GRAD computes the gradient of the predicted response at one time point and stores it in the matrix GRADY. The call to SUMULT at card 129 accumulates the contribution from the time point to the first and second gradients of the cost functional.

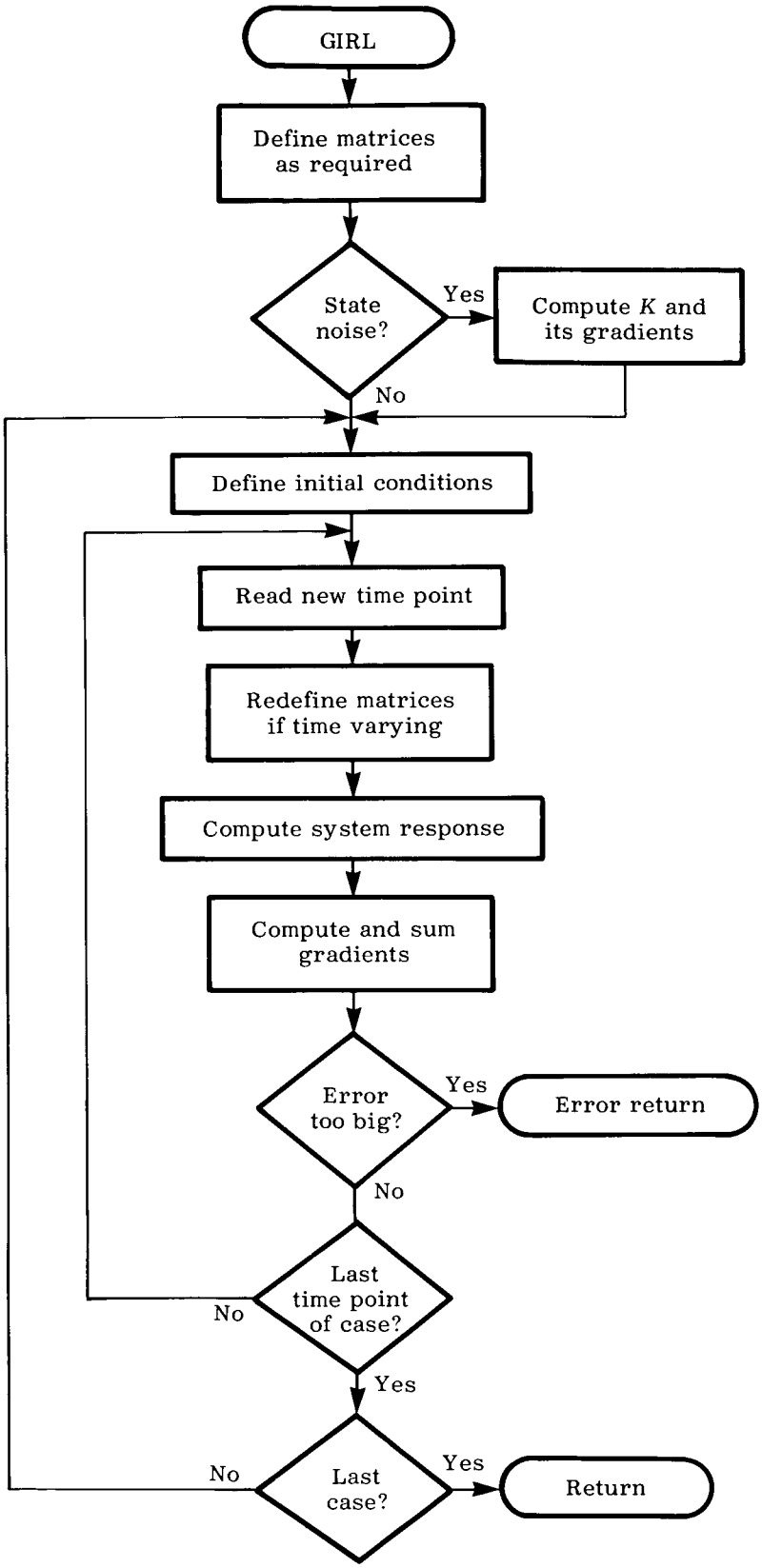
After the case and time loops, cards 137 and 138 move the first gradient of the cost functional from row JKM to column JKM of the SUM matrix, as required by the rest of the program. DIM2 is called to recompute the average dimensional matrices if required for subroutine FLIMIT.

GIRL is called by cards NEWTON.37, 82, and 98.

A.2.1.11.1 CALLAM. - Subroutine CALLAM stores the dimensionalization ratios in common block AMCOM. It calls user routine MAKEM if appropriate to compute the M suffix matrices. CALLAM is called from card DIM1.18.

A.2.1.11.2 DIM1. - Subroutine DIM1 computes the basic dimensional system matrices. If USEAVG is TRUE, it moves average values into common block BILIN so that average dimensional matrices will be computed. CALLAM computes the M suffix matrices, and MAKEL adds in the L suffix matrix contributions. DIM1 is called from card DIM2.11.

A.2.1.11.3 DIM2. - Subroutine DIM2 computes dimensional system matrices and expressions. It calls DIM1 to compute the dimensional matrices. All of the matrix expressions in the common blocks ECOM and OBSRV are then evaluated. DIM2 is called from cards REAT.13, INIT.26, and GIRL.47.


GIRL
Define matrices as required
State noise?
Yes
Compute K and its gradients
No
Define initial conditions
Read new time point
Redefine matrices if time varying
Compute system response
Compute and sum gradients
Error too big?
Yes
Error return
No
Last time point of case?
No
Yes
Last case?
No
Yes
Return

A.2.1.11.4 GRAD. - Subroutine GRAD computes the gradient of the predicted response for one time point. Cards 23 to 26 compute the variables in common deck GRAD$, used later in the subroutine. The gradients of the predicted response are then computed using the equations given in reference 1, section 3.1. It is assumed that, on entry, GRADX contains the gradient of the corrected response at the previous time point. Cards 30 to 34 zero GRADY and multiply GRADX by the transition matrix. GRAD1 is used for scratch storage. The loop from cards 35 to 66 adds terms to GRADX and GRADY for each unknown. Then card 67 adds the $(ER^{-1}A + C)$ GRADX term into GRADY.

At this point GRADX and GRADY contain the gradients of the predicted state and observation. If the state noise algorithm is used, cards 72 to 81 add $-K\ \nabla\tilde{z} + (\nabla K)(z - \tilde{z})$ to the gradient of the predicted $x$ to obtain the gradient of the corrected $x$. This is needed for the next call to subroutine GRAD; only GRADY is required as an external output from GRAD. If the state noise algorithm is not used, the corrected and predicted states are identical, so the last section of code is skipped.

GRAD is called from card GIRL.123.

A.2.1.11.5 GRADIC. - Subroutine GRADIC initializes the gradient of $x$ at the beginning of each maneuver. All elements of the gradient are 0, except those corresponding to variable initial conditions. GRADIC is called from card GIRL.73.

A.2.1.11.6 GRADK. - Subroutine GRADK computes the gradient of the Kalman gain matrix. The results are stored in the triply dimensioned array DK in common block GRDCOM. GRADK assumes that subroutine KALMAN was previously called to compute the Riccati covariance matrix, P, and the Kalman gain matrix, KGAIN.

GRADK first computes RIAP as RIA X P, and calls GRADP. RIAP is used both in GRADP and GRADK. On return from GRADP, the gradient of $P$ is stored in DK. Cards 25 to 28 multiply $\nabla P$ by CTG = $(ER^{-1}A + C)$*GGI (computed in subroutine KALMAN). Cards 30 to 54 then add in the elements of $\nabla P(ER^{-1}A + C)$*GGI to complete the gradient of $K$. Finally, DK is printed if the TEST option is on.

GRADK is called from card GIRL.57.

A.2.1.11.7 GRADP. - Subroutine GRADP computes the gradient of the Riccati covariance matrix and stores the result in the array DK. It assumes that the matrices RIAP = $R^{-1}AP$, RIF = $R^{-1}F$, and RIFRIF = $R^{-1}F(R^{-1}F)^*$ have been computed by the subroutines GRADK and KALMAN. RIFRIF will be destroyed in GRADP. As discussed in reference 1, section 1.6, the gradient of the Riccati covariance matrix is computed as the solution to a group of Lyapunov equations.

Cards 24 to 58 compute one-half of the constant terms in the Lyapunov equations and store the results in DK. To get the full constant terms, the values stored in DK would be added to their transposes. Cards 60 and 61 compute the coefficient of $\nabla P$ in the Lyapunov equations and store the result in DUM. Cards 65 to 74 perform the real eigenvector decomposition of DUM. RIFRIF becomes the block-diagonalized DUM matrix, DUM2 is the matrix of eigenvectors transposed, and DUM3 is the inverse of DUM2.

The loop from cards 76 to 88 solves one of the Lyapunov equations for each pass through the loop. Cards 77 to 82 compute the constant term of the block-diagonalized Lyapunov equation. Then LYAPCB solves this block-diagonalized equation. Cards 84 to 87 transform this solution back into the solution of the original Lyapunov equation.

GRADP is called from card GRADK.23.

A.2.1.11.8 INIT (IT, XT1, U1, Y, V, W). - Subroutine INIT sets the time history initial conditions and biases. The arguments, IT, XT1, U1, Y, V, and W, are the total time in milliseconds, state, biased control, computed observation, state equation forcing function, and observation equation forcing function, respectively, all at the initial time point of a maneuver. Card 16 reads the measured data and stores them in common block BILIN. Cards 18 to 25 define the default state, observation bias, and control appropriately for perturbation equations. If TIMVAR is TRUE, DIM2 is called to compute dimensional matrices at the initial time point. The V and W vectors are computed for use in defining the computed observations. Card 28 calls subroutine UINIT to redefine the state, observation bias, and control as desired by the user. Cards 30 to 32 compute the control bias as the difference between the model control and the measured control at the initial point. Variable initial condition increments are added and OBSERV is called to compute the initial computed observation. INIT is called from card GIRL.68.

A.2.1.11.9 KALMAN. - Subroutine KALMAN computes the Kalman gain matrix. The matrices RIF, RIFRIF, and CTG computed here are used subsequently in subroutines GRADK and GRADP. Cards 20 to 22 define the physical dimensions of P, KGAIN, and CTG. Then cards 23 to 28 compute the matrices of the continuous-time Riccati equation as follows:

$$\text{RIFRIF} = R^{-1}F(R^{-1}F)^*$$

$$\text{DUM} = (ER^{-1}A + C)^*(GG^*)^{-1}(ER^{-1}A + C)\,\frac{1}{\Delta t}$$

Cards 32 to 35 destroy the physical dimension pointer of SUM and define two scratch matrices in the space occupied by the SUM matrix. RICATC is then called to solve the continuous Riccati equation. DUM2 and the two scratch matrices defined by SUM are used for scratch storage matrices in RICATC. The last two arguments of the call to RICATC are columns of SUM used as scratch vectors in RICATC. The scratch usage of the SUM matrix here assumes that the physical number of rows of the SUM matrix is at least two times the physical dimension of the state vector plus one, i.e., NI = 2 X MAXX + 1 (see sec. 3). Card 43 multiplies the Riccati covariance matrix by $(ER^{-1}A + C)^*(GG^*)^{-1}$ to obtain the Kalman gain matrix.

KALMAN is called at card GIRL.48.

A.2.1.11.10 OBSERV (X, U, ONES, V, W, Y). - Subroutine OBSERV expands the observation equation. X, U, ONES, V, and W are input arguments giving the state, control, bias, state equation forcing function, and observation equation forcing function, respectively. The result is stored in Y. OBSERV is called at GIRL.96 and INIT.36.

A.2.1.11.11 REAT. - Subroutine REAT computes dimensional matrices and transition matrices. It first calls subroutine DIM2 to compute the basic dimensional system matrices and matrix expressions. Then EAT is called to compute the transition matrix, PHI, and its integral, DUM. PSI, PSIB, and PSIS are expressions involving the integral of the transition matrix. REAT is called at cards GIRL.44 and 80.

A.2.1.11.12 SPIDIM. - Subroutine SPIDIM prints the dimensional system matrices, matrix expressions, and transition matrices. Subroutine REAT must be called before SPIDIM in order to define all of the matrices printed. SPIDIM is called at card GIRL.45.

A.2.1.12 SUMOUT. - Subroutine SUMOUT handles various output to summarize a case (not including plotted output). The variable BLOWUP, defined by subroutines NEWTON and GIRL, is first examined to see if the case has diverged past program limits. If so, most summary output is skipped, plotting is turned off, and measured time histories are printed if desired. The rest of the subroutine is only executed if BLOWUP is FALSE.

Subroutine CRAMER is called at card 18 to compute Cramér-Rao bounds and parameter correlations. OUTPUN is called at card 19, depending on PUNCH, to punch coefficient estimates as required. Both CRAMER and OUTPUN are bypassed if NOITER is 0. Cards 21 to 34 normalize the residual covariance matrix to obtain the residual correlations and print them out. The unnormalized covariance matrix is saved in FRSQ during this computation. Cards 36 to 42 compute and print the averages and standard deviations of the corrected state estimates. The summary of the convergence of the cost functional is then printed. Finally, the plotting error limit is checked; if it is exceeded, plotting is turned off and, optionally, measured time histories are printed.

SUMOUT is called from card MMLE3.66.

A.2.1.12.1 CRAMER. - Subroutine CRAMER computes the Cramér-Rao bounds and estimated parameter correlations. The Fisher information matrix must be in SUM when CRAMER is called. First the matrices in common block CRMAT are initialized to 0. Then the SUM matrix is inverted. The loop from cards 28 to 53 places the Cramér-Rao bounds in the appropriate locations in the matrices in CRMAT. This loop assumes that the independent unknowns are the first JKMM1 unknowns in the list in common block DETERM. These matrices are then printed. Cards 64 to 75 normalize the inverse of the information matrix to obtain the estimated correlations, which are then printed. CRAMER is called from card SUMOUT.18.

A.2.1.12.2 CRSET (AC, AN, ALAB). - Subroutine CRSET initializes one of the matrices of Cramér-Rao bounds. The argument ALAB is used for the name of the matrix AC unless the name of AN is DONT. If the name of AN is DONT, the name of AC is also set to DONT; this will prevent AC from being printed. CRSET is called at cards CRAMER.16 to 24.

A.2.1.12.3 ERRTHP. - Subroutine ERRTHP prints the measured time history. It is called at card SUMOUT.48 if the cost functional exceeds allowable limits. The data printed are from scratch file UT1, which has already had scale factors, biases, and modifications from subroutine THMOD applied.

A.2.1.13 THPLOT (PLTOPN). - Subroutine THPLOT plots time history data. The data to be plotted are obtained from the scratch file UT2. The argument, PLTOPN, controls whether the plot file is to be opened by the call to PLOTS at card 23. If PLTOPN is TRUE, the file is already open, so the call to PLOTS is skipped. Cards 34 to 44 put the title, start time, maneuver number, and other identifying information on the plot. Cards 46 to 56 determine the time scale and thinning used. Then cards 59 to 65 form the time vector and draw the time axis and label.

The observations are read into core and plotted up to NCH at a time by cards 67 to 110. Cards 68 and 69 determine how many observations are left to plot (if any). Then cards 71 to 74 position the file UT2 at the beginning of the maneuver being plotted. (NIP is the total number of time points in the previous maneuvers.) Cards 76 to 88 read NCHAN measured and computed observations into the X and XX arrays, thinning as needed. The minimum and maximum values are computed during the reading. Cards 92 to 96 determine the scale to be used for a plot. Cards 97 and 98 then plot the label and axis. The time histories are plotted by cards 100 to 105.

Cards 113 to 126 form the IPLT vector which lists the states, controls, and extra signals to be plotted. XPLOT, NUPLT, and NEXPLT define this information. The states, controls, and extra signals are treated together as one concatenated vector throughout subroutine THPLOT. States, controls, and extra signals are read into core and plotted up to 2 X NCH at a time by cards 128 to 168. This code is similar to that used for the observation plots at cards 67 to 110, with the following differences: First, the IPLT vector is used at cards 146 and 151 to select the channels; second, only one line is drawn per plot instead of two; and third, the signal minima and maxima are initialized to 0 at cards 137 to 139 so that automatic scaling will always include 0.

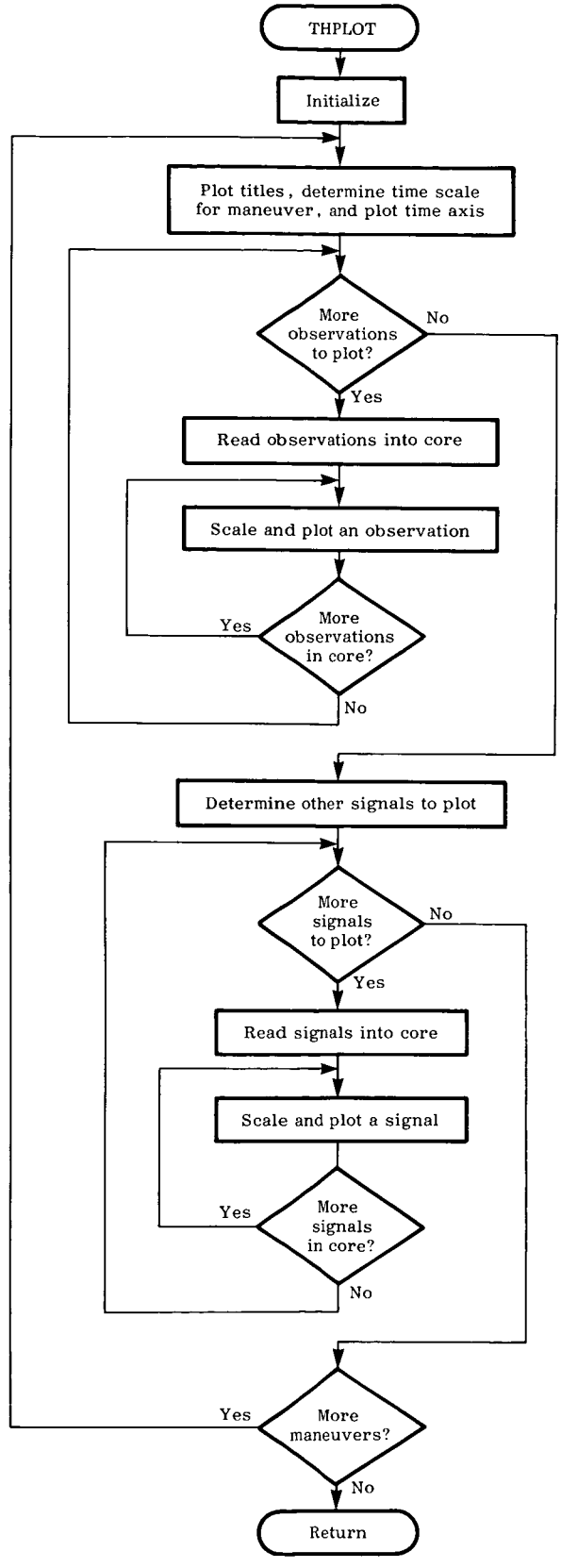
THPLOT is called at card MMLE3.68.

## A.2.2 Utility Subroutines

The utility subroutines perform general tasks, such as plotting and matrix manipulation, that are not specific to the MMLE3 program. The subroutines are discussed in alphabetical order.

A.2.2.1 ABEND. - Subroutine ABEND is an error exit subroutine. It reads 60,000 cards from the input file in order to force an end-of-file error. This intentional error is intended to get an error traceback from the system (both IBM and CDC systems give traceback in response to an end-of-file error). If an end-of-file error does not occur (only realistically possible if the wrong file is connected to unit UCARD (ref. 1, sec. 3.2)), the subroutine stops. The return statement is never executed, but is required by some computers.

A.2.2.2 ADD (A, B, C). - Subroutine ADD adds the matrices A and B, placing the result in C. An error exit is taken if the matrix sizes are not the same. The physical dimension of the C matrix is assumed equal to that of the A matrix.


THPLOT
Initialize
Plot titles, determine time scale for maneuver, and plot time axis
More observations to plot?
No
Yes
Read observations into core
Scale and plot an observation
More observations in core?
Yes
No
Determine other signals to plot
More signals to plot?
No
Yes
Read signals into core
Scale and plot a signal
More signals in core?
Yes
No
More maneuvers?
Yes
No
Return

A.2.2.3 ADDPAR (A, B, IB0, JB0). - Subroutine ADDPAR adds a matrix A into a partition of a matrix B starting at B(IB0, JB0). An error exit is taken if the partition involved exceeds the logical limits of the B matrix.

A.2.2.4 AXES (XPAGE, YPAGE, AXANG, AXLEN, FIRSTV, SCALE, ANNOT, HGT). Subroutine AXES draws an axis and annotation; no axis label is included. XPAGE and YPAGE are the coordinates of the beginning of the axis. AXANG is the angle of the axis from horizontal, and AXLEN is the length. FIRSTV is the beginning value for the annotation, and SCALE is the change per unit distance along the axis. ANNOT controls the placement and orientation of the annotation. Annotation can be placed on either the clockwise or counterclockwise side of the axis, and it can be parallel or perpendicular to the axis, according to the following table.

| ANNOT | Position | Orientation |
|---|---|---|
| 0 | Counterclockwise side | Parallel |
| 90 | Counterclockwise side | 90° |
| -90 | Clockwise side | 90° |
| ±180 | Clockwise side | Parallel |
| Other | Clockwise side | Parallel |

HGT is the height of the annotation. The variable TICDST, defined at card 13, is the distance between tick marks.

Cards 23 to 42 determine the scaling exponent, NEX, and the number of digits placed left of the decimal point, NDECL. Cards 20 to 22 and 44 to 50 determine the number of digits to the right of the decimal point, NDECIM, and the total number of digits including decimal point and sign, NUMLEN. Up to five digits are allowed before a scaling exponent is used. Cards 52 to 70 compute starting positions and sign increments for the annotation. XN and YN are the starting pen positions for the annotation. DXNPOS and DYNPOS are increments added to the pen position if the annotated value is positive. Cards 72 to 83 draw the annotation values. Then cards 85 to 95 draw the scaling exponent if needed. Cards 97 to 110 draw the axis line and tick marks. Negative TICLEN puts the ticks on the side opposite the annotation. If the ticks and annotation are to be on the same side, TICLEN should be made positive.

A.2.2.5 DIGIT (I). - Function DIGIT returns the character representation of an integer input argument, I. The input argument must lie in the range 0 to 20 or an error message will be printed and an error exit taken. The range of legal values can easily be modified. The ENCODE statement found on many large systems performs the task of subroutine DIGIT with much more versatility. Subroutine DIGIT is used, however, because ENCODE is nonstandard and, therefore, not available on some systems.

A.2.2.6 DMULT (A, B, C). - Subroutine DMULT multiplies a diagonal matrix A times a general matrix B and places the result in C. An error message is printed if the dimensions of A and B are inconsistent for multiplication or if A is not square. Off-diagonal elements of A are ignored if present. The physical dimension of C is set equal to that of B.

A.2.2.7 EAT (A, TT, PHI, APHI, A2, A3, NEAT). - Subroutine EAT computes the matrix exponential $e^{At}$ and its integral. Series expansion with time scaling is used (ref. 7). Ten terms are used for the series expansion. A is the matrix which is to be exponentiated, TT is the time interval, and NEAT is the power of 2 used for time scaling. On return, PHI is the matrix exponential $e^{At}$, and APHI is its integral $\int_0^t e^{As} ds$. A2 and A3 are scratch matrices.

Cards 9 to 20 compute the time-scaled matrix exponential and its integral. Cards 22 to 32 then double the time interval NEAT times.

A.2.2.8 EIGENG (A, Z, WR, WI, FV1, VECTS). - Subroutine EIGENG computes the eigenvalues and, optionally, normalized eigenvectors of a real general matrix. EIGENG uses the EISPACK (ref. 6) routines BALANC, ELMHES, HQR, ELTRAN, HQR2, and BALBAK. These routines use the *QR* algorithm after balancing and elementary transformation to upper Hessenberg form (ref. 8). A is the input matrix, which is destroyed. VECTS is an input argument, set to TRUE if eigenvectors and eigenvalues are desired, FALSE if only eigenvalues are desired. Z is the matrix of eigenvalues if VECTS is TRUE, otherwise Z is not used and need not be dimensioned in the calling subroutine. The vectors are stored in the same order as the eigenvalues. For complex eigenvectors, the real part is stored in one column, followed by the imaginary part (corresponding to positive imaginary part of eigenvalue) in the next column. The complement eigenvector is not stored. WR and WI contain the real and imaginary parts of the eigenvalues. The eigenvalues are not ordered, except that complex conjugate pairs are adjacent, with the positive imaginary part first. FV1 is a scratch vector the same length as WR and WI needed only if VECTS is TRUE. Error messages are printed if A is not square or if the *QR* algorithm fails.

Cards 27 to 39 call the recommended sequence of EISPACK routines for the eigenvalues only or eigenvalues and eigenvectors. WI is used for scratch storage and communication between ELMHES and ELTRAN. WR is used for scratch storage in BALANC; if VECTS is TRUE, WR is communicated to BALBAK via FV1. IS1 and IS2 are used for communication between the EISPACK routines. Cards 41 to 63 normalize the eigenvectors if eigenvectors are requested.

A.2.2.9 GET (A, MAX, II, JJ). - Subroutine GET finds the physical and logical dimensions of a matrix, A. On return, MAX is the physical number of rows, II and JJ are the logical number of rows and columns, respectively.

Cards 12 to 19 search for the physical dimension flag stored in the first column of the last row of A. In order to speed this search, a table of previously used physical dimensions (MAXS) is maintained. The locations of this table are checked first; then, locations 1 to MAXMAX are checked. Error messages are printed if the table of physical dimensions used becomes too long (current limit is 10). Cards 28 and 29 find the logical number of columns and rows.

Subroutine GETSET must be called before the first call to GET in order to define TEST and MAXMAX and to initialize NUMBER to 0.

A.2.2.10 GETLAB (A). - Function GETLAB returns the name of the matrix given as an argument, A. If the name location is 0 (common when the name was never defined), the characters NONE are returned for the name.

A.2.2.11 GETP (A, MAX). - Subroutine GETP finds the physical dimension of a matrix, A. MAX is returned as the physical dimension. If the physical dimension flag is not found, 0 is returned for MAX. Note that this is different from subroutine GET, which stops with an error message if the physical dimension flag is not found.

A.2.2.12 GETPAR (A, B, IA0, JA0, IIB, JJB). - Subroutine GETPAR moves the IIB by JJB partition of a matrix A starting at A(IA0, JA0) to a matrix B. A and B may occupy the same location in storage. If the physical dimension of B was previously defined, it is used as defined; otherwise, the physical dimension of B is assumed the same as that of A. An error exit is taken if the specified partition exceeds the logical dimensions of A.

A.2.2.13 GETSET (MAX). - Subroutine GETSET is the initialization routine for the matrix manipulation routines. It must be called before any of the other matrix routines are used. The argument, MAX, is the largest matrix physical dimension that will be allowed. An error exit is taken if MAX is less than 2 or greater than 101. GETSET defines the value, 4HTEST, used for the physical dimension pointers and initializes the list of physical dimensions to 0. GETSET defines the I/O unit numbers used by the matrix routines for the card reader (1), card punch (2), and line printer (3), if they were not previously defined. A warning message is printed if GETSET defines these values. If the unit numbers are defined in GETSET and are inconsistent with the unit numbers that should be used, errors will generally result.

A.2.2.14 IDENT1 (A, MAX, II). - Subroutine IDENT1 initializes a matrix to an identity. The first argument is the matrix, A; the second argument is the physical size of the matrix, MAX; the third argument is the logical number of rows, II. Since the matrix will be square, the number of columns is not needed. Subroutine ZOT1 is called to set up the matrix and initialize it to 0. The diagonal elements are then set to 1.

A.2.2.15 IDIGIT (A). - Function IDIGIT returns the integer value corresponding to a single-digit, Hollerith character argument, A. A blank is treated as 0; any other nonnumeric character returns a value of -1. Arguments containing more than one digit will not be recognized; thus, they will also return a -1 value. The FORTRAN DECODE statement performs the function of IDIGIT with more generality but is not available on many systems.

A.2.2.16 IHMSMS (ITM, T). - Subroutine IHMSMS converts time in total milliseconds to hours, minutes, seconds, and milliseconds. The first argument is an integer with time in total milliseconds, ITM. The second argument, T, is a four-word integer output vector with the hours, minutes, seconds, and milliseconds.

A.2.2.17 INV (A). - Subroutine INV inverts a general square matrix, A, in place. The routine is relatively unsophisticated and will not perform well on ill-conditioned matrices. The algorithm used is Gauss elimination (ref. 8) with no pivoting. An error exit is taken if the matrix is not square.

A.2.2.18 LINES (X, Y, NPT, ISKIP, JSKIP, HGT, L). - Subroutine LINES plots solid or dashed lines through a vector of points. X and Y are the vectors of the X and Y values to be plotted; NPT is the number of points in these vectors. The locations X(NPT + 1) and Y(NPT + 1) specify the X and Y values at the origin; X(NPT + 2) and Y(NPT + 2) specify the X and Y scales in units per centimeter. ISKIP is a skipping parameter. The absolute value of ISKIP is a thinning parameter for the data. If ISKIP is 1, all of the data are used; if 2, every second point is used, etc. If ISKIP is positive, the data are plotted starting at the beginning of the X and Y vectors; if negative, starting at the end. JSKIP controls dashing and symbols. If JSKIP is 0, a solid line with no symbols will be drawn. If JSKIP is positive, a solid line will be drawn with symbols every JSKIPth point. If JSKIP is negative, a dashed line will be drawn with dashes JSKIP - 1 intervals long and spaces 1 interval long. A symbol will be put at the beginning of each dash; therefore, JSKIP = -1 results in symbols only since the dashes are of length 0. HGT is the symbol height, and L is the CalComp symbol number (ref. 4).

A.2.2.19 LYAPCB (P, A, C). - Subroutine LYAPCB solves a continuous-time block-diagonal steady-state Lyapunov equation. The form of the equation is

$$AP + PA^* = C$$

where the matrix A is block diagonal with 2 by 2 maximum blocks; the 2 by 2 blocks must be skew symmetric, and both diagonal elements of the blocks must be equal. C must be symmetric. The first argument, P, is the symmetric solution matrix to the equation. The second and third arguments, A and C, are square input matrices. There are no checks on the structure of A and C.

Because of the block-diagonal structure of A, the problem separates into 1 by 1, 1 by 2, and 2 by 2 partitions. Each such partition is a linear equation in 1, 2, or 4 elements of the P matrix. The solutions to these partitions are coded explicitly, taking advantage of the properties of A.

A.2.2.20 MAKE (X, Y). - Subroutine MAKE copies a matrix. The first argument is the output matrix, X; the second argument is the input matrix, Y. The physical dimensions of the input and output matrices are assumed to be the same (see subroutine MOVE otherwise).

A.2.2.21 MATLD (A). - Subroutine MATLD reads the body of a matrix, A, from the card reader file. The matrix name, number of rows, and number of columns must be in common block INMAT when MATLD is called. The physical dimension of the matrix must have been defined previously. The matrix is read one row to a card in 8F10 format. If the number of columns is 0, the matrix is assumed to be diagonal, and the diagonal elements are read from one card. After reading the matrix, MATLD calls subroutine SPIT to print it out.

A.2.2.22 MIL (T). - Function MIL converts time in hours, minutes, seconds, and milliseconds to total time in milliseconds. The input argument, T, is a four-word integer vector.

A.2.2.23 MOVE (A, B). - Subroutine MOVE moves a matrix A into a matrix B. The physical dimensions of A and B must be defined before the call and may be

different. MOVE differs from MAKE in the order of the arguments and the treatment of the physical dimensions.

A.2.2.24 MULT (A, B, C). - Subroutine MULT multiplies a matrix A times a matrix B and places the result in C. The physical dimension of C is assumed equal to that of A. An error exit is taken if the number of columns of A is not equal to the number of rows of B. The CPU time spent in MULT is significant; therefore, it may prove worthwhile to use assembly language versions of MULT.

A.2.2.25 MULTT (A, B, C). - Subroutine MULTT multiplies a matrix A times a matrix B* and places the result in C. The physical dimension of C is assumed equal to that of A. An error exit is taken if the number of columns of A is not equal to the number of columns of B.

A.2.2.26 MVMULT (A, B, C). - Subroutine MVMULT multiplies a matrix A times a vector B and places the result in matrix C.

A.2.2.27 PLOP (X). - Subroutine PLOP punches a matrix, X, in standard matrix format. If the matrix name is DONT, the subroutine is bypassed.

A.2.2.28 PLTDAT (X, Y). - Subroutine PLTDAT puts the date and time on a plot for plot identification. The arguments are the X and Y position at which the date and time are to be placed. Subroutine PLTDAT calls the machine-specific subroutines DATE and TIME to obtain the information from the system clock.

A.2.2.29 REDUCE (A, MAX, N). - Subroutine REDUCE factors a positive definite symmetric matrix using Cholesky's decomposition (ref. 8) and inverts the factors. The arguments are the matrix, A; the physical dimension, MAX; and the logical dimension, N. The matrix is replaced by the factorization on exit. The matrix is factored into the form $L^{-1}DL^{*-1}$, where L is lower triangular with unity diagonal elements and D is diagonal. On exit, the lower triangular part of the matrix contains L (except for the diagonal elements, which are not stored), and the diagonal contains D. The strict upper triangle is not used.

A.2.2.30 RICATC (P, A, B, C, DUM, H, E, WR, WI, FV1). - Subroutine RICATC solves the continuous-time steady-state matrix Riccati equation $AP + PA^* + B - PCP = 0$. The first argument is the symmetric solution matrix, P. The second to fourth arguments are the square input matrices, A, B, and C. (B and C must be symmetric.) The fifth argument is a dummy matrix, DUM, which is the same size as A, B, and C. The sixth and seventh arguments, H and E, are dummy matrices of physical dimension at least 2N + 1 by 2N, where N is the logical number of rows at A. The last three arguments, WR, WI, and FV1, are scratch 2N-vectors.

Potter's method (ref. 9) is used for the solution. Error messages are printed and the program stops if the Hamiltonian has a 0 eigenvalue or if exactly half of its eigenvalues do not have negative real parts.

A.2.2.31 ROWCOL (IR, IC, STRING). - Subroutine ROWCOL picks a matrix row and column number from a string of five characters. The first two arguments, IR and IC, are the integer row and column number outputs, respectively. The third

argument, STRING, is an input five-word vector containing one Hollerith character in each word. The row and column numbers may use one or two characters and must be separated by a comma. If less than five characters are required, any nonnumeric character, except a blank, can be used as a terminator. Subroutine ROWCOL is generally used when matrix locations are being read from cards.

A.2.2.32 SCALE2 (XMIN, XMAX, S, AMIN, SCALE). - Subroutine SCALE2 determines reasonable plotting scales. The first two arguments, XMIN and XMAX, are the minimum and maximum data values, respectively. S is the axis length in centimeters or half inches. AMIN and SCALE are output arguments for the minimum value for the axis and the scale in units per centimeter or per half inch. If the data minimum is greater than or equal to the maximum, the scale will be set to -999, and the minimum value for the axis will not be defined. Scales of one, two, or five times a power of 10 units per centimeter or per half inch will be used by SCALE2. The scale and minimum value will also be chosen so that the value 0 would appear a multiple of 2 centimeters or half inches from the beginning of the axis (though it need not lie within the range of the axis). If both positive and negative values are included, the axis length should be at least 4 centimeters or half inches to insure that all values will fit within the range of the axis. The use of centimeter or half inch units depends on the program's call to FACTOR (sec. 2.5). SCALE2 is not directly affected.

A.2.2.33 SET (A, II, JJ). - Subroutine SET defines the logical dimensions of a matrix. The physical dimension must have been previously defined. The three arguments are the matrix, A; the logical number of rows, II; and the number of columns, JJ. An error message is printed and an error exit taken if the logical number of rows is greater than or equal to the physical dimension.

A.2.2.34 SET1 (A, MAX, II, JJ). - Subroutine SET1 defines the physical and logical dimensions of a matrix. The four arguments are the matrix, A; the physical number of rows, MAX; the logical number of rows, II; and the logical number of columns, JJ. An error message is printed and an error exit taken if the logical number of rows is greater than or equal to the physical dimension.

A.2.2.35 SET2 (A, MAX, II, JJ, ALAB). - Subroutine SET2 defines the physical and logical dimensions of a matrix and its name. SET2 is identical to SET1 with the addition of a last argument, ALAB, for the matrix name.

A.2.2.36 SINV (A). - Subroutine SINV inverts a positive definite symmetric matrix, A, in place. The strict upper triangle of the matrix is ignored. SINV calls subroutine REDUCE to compute the inverse of the Cholesky factorization (ref. 8). The inverse factors are then multiplied to obtain the inverse of the full matrix.

A.2.2.37 SMULT (G, A, B). - Subroutine SMULT multiplies a matrix A by a scalar G and places the result in B. A and B may occupy the same storage locations; they must always have the same physical dimensions.

A.2.2.38 SPIT (A). - Subroutine SPIT prints a matrix, A. If the matrix name is DONT, the subroutine is bypassed. Matrices with more than 10 columns are printed in blocks of 10 or less columns at a time.

A.2.2.39 SSIMEQ (A, B, X). - Subroutine SSIMEQ solves the symmetric linear system AX = B. A is a symmetric matrix, B is a vector, and X is the solution vector. Subroutine REDUCE is called to obtain the inverse of the Cholesky factorization (ref. 8) of A. Then X is computed by multiplying B by the inverse factors. The lower triangle and diagonal of A contain the factorizations from subroutine REDUCE on return. The strict upper triangle of A is ignored.

A.2.2.40 SUB (A, B, C). - Subroutine SUB subtracts a matrix B from a matrix A and places the result in C. An error exit is taken if the physical or logical dimensions of A and B are not the same. The physical dimension of C is assumed equal to that of A and B.

A.2.2.41 SUMULT (A, B, C). - Subroutine SUMULT replaces the matrix C by A*B + C. Only the lower triangular and diagonal parts of C are computed. The A and B matrices are assumed to have the same physical and logical dimensions; there is no check to verify this however. The physical dimension of C may differ from that of A and B. A significant amount of CPU time is spent in SUMULT, so it may prove worthwhile to write an assembly language version.

A.2.2.42 SYM (A, PRNT). - Subroutine SYM checks a matrix, A, for symmetry. If the matrix is not square, an error exit is taken. If the matrix is square, but not symmetric, it is symmetrized using the lower triangular part. The logical variable PRNT controls a warning message. If PRNT is TRUE, a warning message is printed when A is square but not symmetric. If PRNT is FALSE, the symmetrization is done without comment.

A.2.2.43 SYMBL4 (X, Y, HGT, TITLE, ANGLE, NCHAR). - Subroutine SYMBL4 performs the same function as the standard call to the CalComp routine SYMBOL (sec. 2.5), except that the data are assumed to be stored only four characters per word. NCHAR characters of alphanumeric data, stored four characters per word in the array TITLE, are plotted at an angle of ANGLE degrees from the horizontal. NCHAR is limited to 400; otherwise, only the first 400 characters will be plotted. The starting position for the plot is given by X and Y. Following the usual CalComp convention, X and/or Y may be 999 to indicate that the plot starts at the previous pen position. HGT is the height of the symbols in centimeters.

A.2.2.44 TRANSP (A, B). - Subroutine TRANSP transposes a matrix A and places the result in matrix B. A and B may occupy the same storage locations. The physical dimensions of A and B may be different. If the physical dimension of B was not previously defined, it is assumed equal to that of A.

Cards 16 to 34 handle the largest square partition of A. This part must be handled specially since A and B may occupy the same storage locations. Cards 37 to 45 handle the remaining columns of A if A had more columns than rows. Cards 47 to 54 handle the remaining rows if A had more rows than columns.

A.2.2.45 UNSET (A). - Subroutine UNSET deletes the physical dimension pointer from a matrix, A, if it is present. This has the effect of making the physical dimension of A undefined. UNSET is used if the space in a matrix is to be used for a temporary scratch matrix with a different physical dimension. It is also used afterward, preparatory to redefining the correct physical dimension for the matrix.

A.2.2.46 VMADD (A, X, B, U, S, ONES, C, V, Y). - Subroutine VMADD forms the expression AX + BU + S ONES + CV and stores the result in Y. A, B, S, and C are matrices which may have different physical dimensions and logical number of columns. An error exit is taken if the logical number of rows of A, B, S, and C are not all the same. X, U, ONES, V, and Y are vectors.

A.2.2.47 ZMULT (A, B, C). - Subroutine ZMULT computes the matrix expression C = C + AB. An error exit is taken if the logical number of columns of A does not equal the number of rows of B. The physical dimensions of A and C are assumed equal. ZMULT takes advantage of 0's and 1's in the A matrix to save time. Therefore, if A is sparse, ZMULT is quite efficient. For general A matrices, subroutine MULT may be more efficient. A significant amount of CPU time is spent in ZMULT, so it may prove worthwhile to use an assembly language version.

A.2.2.48 ZOT (A). - Subroutine ZOT zeros a matrix, A. The physical and logical dimensions of A are assumed to have been previously defined.

A.2.2.49 ZOT1 (A, MAX, II, JJ). - Subroutine ZOT1 defines the physical and logical dimensions of a matrix, A, and then zeros the matrix. MAX is the physical number of rows. II and JJ are the logical number of rows and columns, respectively.

A.2.2.50 ZOT2 (A, MAX, II, JJ, ALAB). - Subroutine ZOT2 defines the physical and logical dimensions and the name of a matrix, A, and then zeros the matrix. MAX is the physical number of rows. II and JJ are the logical number of rows and columns, respectively. ALAB is the matrix name.

## A.2.3 Standard Aircraft Routines

This section describes the standard aircraft user routines. The subroutines are listed in alphabetical order. Each subroutine is first described in terms of its general function in the program. The communication to the basic program is described, and the place from which the subroutine is called is referenced. This description is intended to guide the user in coding the new set of user routines for a different problem. After the general description, the standard aircraft version of each subroutine is discussed. Subroutine WTDEF is called from MATDEF, and subroutines INTERP and WTTRAN are called from WTDEF. WTDEF, INTERP, and WTTRAN are never called directly from the basic program; therefore, they do not have general functional descriptions independent of the standard aircraft routines—they are merely parts of the standard aircraft routine's implementation of subroutine MATDEF.

A.2.3.1 AVERAG. - Subroutine AVERAG provides the user routines convenient access to the time history averages. AVERAG is called at card THDATA.111. This call occurs after USERIN and before MATDEF. AVERAG is therefore convenient for defining variables needed for MATDEF, but not read in by USERIN. The averages are available in common block AVGCOM.

The standard aircraft routine AVERAG obtains QBAR, V, ALPHA, THETA, PHI, and MACH if they were not read by subroutine USERIN. The values 0 and 999, depending on the variable, are indications that USERIN did not read these variables.

The channels for ALPHA, THETA, and PHI depend on whether the case is longitudinal or lateral-directional. The ALPHA computation includes the corrections for upwash and instrument position. Cards 19 and 20 define the distances that the moment derivatives and instrument positions must be shifted to reference them to the flight center of gravity. This computation has no direct connection with the time history averages in the subroutine supplied; it could just as well have been done in subroutine USERIN. It is envisioned, however, that the user might desire to make a program modification to compute the flight center of gravity as a function of some extra channel averages. Therefore, no computations are done with the center of gravity until this point so that such a modification can be readily made.

A.2.3.2 INTERP (ALPHA, NABP, ABP, IA, JA, FIA, FJA). - Subroutine INTERP computes indices and factors for linear interpolation. Subroutine INTERP is a standard aircraft routine called only from the standard aircraft routine WTDEF; therefore, it does not have a general function independent of the standard aircraft routines.

ABP is a vector of length NABP containing the break point values for the independent variable. The values in ABP are assumed to be monotone strictly increasing; no check is made to verify this. ALPHA is the value of the independent variable to which data are to be interpolated. On return, IA and JA are the indices to be used for interpolation, and FIA and FJA are the interpolating factors. To interpolate a vector, DATA, containing dependent variable values corresponding to the independent variable values in the ABP, the expression FIA * DATA(IA) + FJA * DATA(JA) would be used following the call to subroutine INTERP.

If the value of ALPHA lies outside the range of the ABP values, subroutine INTERP limits the output to the range given; it does not attempt to extrapolate outside of the range. Consequently, INTERP will work correctly when NABP is 1.

A.2.3.3 MAKEL. - Subroutine MAKEL adds in the contribution from the L suffix matrices to the dimensional matrices. It is called from card DIM1.37. On entry, common block DIMMAT contains the dimensional matrices except for the contributions from the L suffix matrices. Subroutine MAKEL should compute the L suffix matrices and add the contributions to the dimensional matrices. The L suffix matrices need not be stored separately. Common block BILIN contains the logical variable TIMVAR (ref. 1, sec. 3.3.8(20)) and the measured observations, controls, and extra signals. The L suffix matrices can be functions of the measured quantities in common block BILIN. The L suffix matrices should not be dependent on the unknown coefficients, directly or indirectly (in particular they should not depend on the computed time histories).

The standard aircraft routine MAKEL computes the matrices described in reference 1, section 4.1.3. Cards 18 to 34 define the variables VT, ALPR, THETR, and PHIR. If TIMVAR is TRUE, measured values from common block BILIN are used to define these variables. The statement function ALPHAC corrects the measured angle of attack for upwash and angular rates. If TIMVAR is FALSE, values from common block FLCOND are used; these values can have come either from average measured values or from NAMELIST USER (ref. 1, sec. 4.3.3). Cards 35 to 39 compute trigonometric functions of the angles. The lateral-directional L suffix matrix terms are added by cards 43 to 48, and the longitudinal ones by cards 51 to 54. The variables DGDP and DGDT defined at cards 45 and 53 are passed through common block GRAV to subroutine MAKEVW.

A.2.3.4 MAKEM. - Subroutine MAKEM computes the dimensionalization ratios, M suffix matrices. The values computed should be stored in common block MATRAT. MAKEM is called from card CALLAM.11. On entry, the M suffix matrices in MATRAT will be filled with 1's. Common block BILIN contains the logical variable TIMVAR (ref. 1, sec. 3.3.8(20)) and the measured observations, controls, and extra signals. The M suffix matrices can be functions of the measured quantities in common block BILIN. The M suffix matrices should not be dependent on the unknown coefficients, directly or indirectly (in particular they should not depend on the computed time histories).

The standard aircraft routine MAKEM computes the matrices described in reference 1, section 4.1.3. The dynamic pressure, QT, and velocity, VT, are obtained from common block BILIN or FLCOND, depending on TIMVAR. The values in FLCOND can have come either from average measured values or from NAMELIST USER (ref. 1, sec. 4.3.3).

A.2.3.5 MAKEVW (VB, WB, FIRST). - Subroutine MAKEVW computes the known forcing functions $v(t)$ and $w(t)$. It is called at cards INIT.27 and GIRL.88. The formal parameter names used for $v(t)$ and $w(t)$ are VB and WB to avoid confusion with velocity and weight in the standard aircraft routines. The logical variable FIRST is TRUE on the first time point of each maneuver so that the subroutine can do any reinitialization needed. The value of FIRST should not be changed by the subroutine. Subroutine MAKEVW will be called at each time point regardless of the variable TIMVAR (ref. 1, sec. 3.3.8(20)). The VB and WB vectors can be functions of the measured quantities in common block BILIN. They can also be functions of the L suffix and M suffix matrices computed by user routines MAKEL and MAKEM, because the appropriate L suffix and M suffix matrices (time-varying or not) will have been defined before the call to subroutine MAKEVW. The VB and WB vectors should not be dependent on the unknown coefficients, directly or indirectly (in particular they should not depend on the computed time histories).

The standard aircraft routine MAKEVW computes the vectors described in reference 1, section 4.1.3. The time-varying velocity from common block BILIN is used if TIMVAR is TRUE; otherwise, the constant average velocity from common block FLCOND is used. The angles of attack and sideslip used in the longitudinal equations are the measured values corrected for upwash and instrument position. Cards 37 and 64 use the variables DGDP and DGDT defined by user routine MAKEL. The terms on these two cards represent part of the linearizations of the gravity terms about the measured values. If the $\dot{\theta}$ or $\dot{\phi}$ equations are not integrated (determined by the value of MX, the length of the state vector) the gravity terms must be evaluated at the measured values instead of being linearized; therefore, the terms on cards 37 and 64 are omitted.

A.2.3.6 MATDEF (WTFILE). - Subroutine MATDEF defines input and matrix defaults. MATDEF is called at card MMLE3.62. Defaults can be defined in MATDEF for any of the matrices that can be read from cards. These matrices are found in common blocks FCOM (F matrix), GICOM (GGI matrix), MATIN (V suffix and APR prefix matrices and constraint matrix HARD), MATRIX (N suffix matrices), and SOFCOM (constraint matrix SOFT). Data from common block AVGCOM can be used to define the defaults. Subroutines USERIN and AVERAG are called before MATDEF, so any quantities defined in these two routines can also be used (plus, of course, quanti-

ties defined in ONCE). The formal parameter WTFILE is a logical variable which indicates whether a predicted-derivative file is available. If WTFILE is TRUE, data may be read from the predicted-derivative file to be used in defining the defaults.

Subroutine MATDEF is called after the matrices have been read in from cards. The logical function LOADED can be used to determine whether a particular matrix was read from cards. The matrix is used as the argument for function LOADED, which returns a value of TRUE if that matrix was read from cards. Normally, the defaults for a matrix will be skipped if LOADED returns TRUE. MATDEF can define elements of a matrix, even if LOADED is TRUE for that matrix; such definitions would constitute overrides of the values read from cards. Subroutine SET can be used to define the logical size of a matrix.

The constraint matrices, HARD and SOFT, require special mention. The function LOADED can return a value of FALSE for these matrices even if the matrices were in fact read from cards. In this case, the default constraints should be used in addition to the constraints read in. The input cards specify whether the constraints read in supplement the default constraints in this manner or replace the default constraints (ref. 1, sec. 3.3.11(6) and (7)). In order to implement the convention described in reference 1, section 3.3.11(6), the LOADED function is simply used in the normal manner; the default constraints are used if and only if LOADED is FALSE. The user should be conscious, however, that constraints read from cards can be present even when LOADED is FALSE. Calls to subroutine SETCON can be used to conveniently define the default constraints. SETCON is described in section A.2.1.9.5.

The final values of the variables MX, MZ, MU, and MB (ref. 1, sec. 3.3.8(11) to (14)) in common block SIZE will not have been computed at the time MATDEF is called because the values may depend on the results from MATDEF. Therefore, these values should not be used by MATDEF unless it duplicates the logic that will compute the final values. The standard aircraft routine MATDEF duplicates the logic that determines MU and MZ at cards 27 to 31, 117, and 118.

The standard aircraft routine MATDEF defines the defaults described in reference 1, section 4.3.5. Card 33 calls WTDEF to define defaults using the predicted-derivative data if available. Cards 35 to 41 define the default HV for longitudinal or lateral-directional cases. Cards 44 to 107 define the rest of the lateral-directional defaults, and cards 110 to 200 define the rest of the longitudinal defaults. The GGI and F defaults are obtained from common block GFDEFS, previously defined by subroutine ONCE. The variables DCGFT, ALPHA, and V were defined by subroutine AVERAG. Most of the remaining variables used in MATDEF were defined by subroutine USERIN.

The cards 116 to 119 require special mention. These cards decide whether to use the axis transformation to obtain the $C_L$ derivatives in the $\alpha$ equation from the $C_N$ and $C_A$ derivatives in the $a_n$ and $a_x$ equations. If both $a_n$ and $a_x$ observations are used, the program does the axis transformation. If $a_x$ is not used, the transformation cannot be done; therefore the low $\alpha$ approximation $C_L = C_N$ is used. This approximation is implemented by setting the variable ALPR ($\alpha$ used for the transformation, in radians) to 0 at card 119 and defining $C_{L_\alpha} = C_{N_\alpha}$ at card 132; the constraints from the

fifth row of CN, DN, and HN will automatically be ignored since the derivatives in these locations will not be unknown. The decision on whether to use the axis transformation or the low $\alpha$ approximation depends on MZ, the length of the observation vector. The program variable MZ will not have been finally defined when MATDEF is called. If the program variable MZ has the value -1, it will subsequently be redefined as the number of rows of the GGI matrix. Therefore, subroutine MATDEF must check both GGI and MZ to determine what the final value of MZ will be. The size of GGI is obtained by the call to subroutine GET at card 117. Note that this call is executed after the longitudinal default GGI matrix is defined at cards 110 and 111.

Cards 203 to 207 override the input or default HV matrices for both longitudinal and lateral-directional cases. These cards delete unknown biases for observations that are not weighted. This forestalls the common trivial error of weighting a signal to 0 and forgetting to delete the signal bias from the HV matrix. The bias is, of course, not identifiable when the signal is not weighted.

A.2.3.7 ONCE. - Subroutine ONCE does any initialization for the user routines. ONCE is called only once at card MMLE3.45 at the beginning of the program, not for each case. Any user input which is to be read only once should be read in subroutine ONCE. Predicted-derivative input does not belong in subroutine ONCE; a separate routine, WTIN, is used for it. Subroutine ONCE is called before any of the other user routines, including user routine WTIN.

The standard aircraft routine ONCE reads defaults for the GGI and F matrices. Cards 19 to 25 define the defaults to be used if none are read in. Card 27 reads the matrix header cards, and card 28 checks for the end card. Cards 30 to 45 determine which matrices are being read in and call subroutine MATLD to read the matrix bodies. Common block INMAT passes the matrix name and size from the header card to subroutine MATLD.

A.2.3.8 OUTPUN. - Subroutine OUTPUN punches out the estimates and related information as desired for derivative plotting or other programs using the estimates. The call to OUTPUN at card SUMOUT.19 is controlled by the NAMELIST variable PUNCH (ref. 1, sec. 3.3.8(45)).

The standard aircraft routine OUTPUN punches the nondimensional matrices and Cramér-Rao bound matrices. Subroutine PLOP is called to punch the matrices. Matrices with no independent unknowns will have DONT stored for their names. Subroutine PLOP will take no action for such matrices. The standard aircraft subroutine OUTPUN also punches averages, standard derivations, minima, and maxima for the measured observations, controls, and extra signals.

A.2.3.9 READTH (INSTAT). - Subroutine READTH reads the input time history data. READTH is called at cards THDATA.48 and 87. One time frame of data should be returned in common block RECRD for each call to subroutine READTH. The time placed in common block RECRD is a four-word integer vector in hours, minutes, seconds, and milliseconds. The logical variable EOFTH in common block RECRD can be set to TRUE to indicate that good data were not placed in common block RECRD because no more data were available. The variable NREC in common block INORD is the number of data channels to be read. NREC can be ignored if desired; it is not used elsewhere in the program.

The argument INSTAT gives information about the status of the time history input. Subroutine READTH should not change the value of INSTAT. INSTAT is 0 on the first call to READTH for a case. A test for 0 can be used to control initialization. INSTAT is 1 when the program is searching for a start time, after the first call to READTH. In a multiple maneuver, there will be a search for the start time for each maneuver of the case after the previous maneuvers have been read. INSTAT is 2 when a start time has been found and data to be used are being read.

The variables ITM and REW in common block TAPPOS give other information on the input status. ITM is the total time in milliseconds of the previously read frame (ITM is initialized to 0 at the beginning of the program). REW indicates when a rewind of the time history data file is advised. Before the first call to READTH for each maneuver, REW is set to TRUE if the requested start time for the maneuver is less than or equal to the previously read time point. On all other calls to READTH, REW will be FALSE. The action to be taken when REW is TRUE is up to the user and depends on the file structure. Usually, this will mean that the desired maneuver has been passed, and the time history data file should be rewound.

The variables CARD and TAPE in common block INOPT are described in reference 1, section 3.3.8(2). These variables are passed to subroutine READTH to indicate the source of time history data; they are not used elsewhere in the program. Subroutine READTH can obtain data from any source; the interpretation of CARD and TAPE is up to the user. One common usage is to have READTH create a simulated time history.

The standard aircraft subroutine READTH reads data from cards or the time history input data file, UDATA, as determined by the variable CARD. The file UDATA can be either a tape or disk file. The file UDATA is rewound whenever REW is TRUE and the input is from file UDATA. The argument INSTAT is not used by the standard aircraft routine.

A.2.3.10 THMOD (FIRST). - Subroutine THMOD modifies the time history data. Scale factor and bias corrections can be made using the NAMELIST variables described in reference 1, section 3.3.8(8) and (9). More complicated data corrections or modifications must be done in subroutine THMOD. THMOD is called once for each time point at card THDATA.74, after scale factor and bias corrections to the data, but before any other operations. The measured time histories printed, plotted, or used internally in the program are the modified time histories resulting from subroutine THMOD. The raw data are not retained.

The logical argument FIRST is TRUE on the first time point of each maneuver; this informs THMOD of time discontinuities that may require reinitialization. The value of FIRST should not be changed by THMOD. The time history data are in common block BILIN. The modified time histories should be placed back in the same locations in common block BILIN.

The standard aircraft subroutine THMOD is a null routine.

A.2.3.11 THOUT (FIRST, IT, X, Y). - Subroutine THOUT writes the output time history file, UTHOUT. THOUT is called once for each time point at cards GIRL.76 and 120. The logical argument FIRST is TRUE on the first time point of each maneuver; this informs THOUT of points where initialization or reinitialization may be necessary.

The value of FIRST should not be changed by THOUT. IT is a four-word vector with time in hours, minutes, seconds, and milliseconds. X and Y are the corrected state estimate and predicted observation, respectively. The measured observations, controls, and extra signals are in common block BILIN. Other time history parameters can be obtained from common block TOGRAD if necessary. The logical vector lengths and physically dimensioned lengths are available in common blocks SIZE and MAXIMS. Subroutine THOUT can write any data desired to file UTHOUT; that file is not used elsewhere in the program.

The standard aircraft routine THOUT writes the time, predicted observation, controls, and extra signals to file UTHOUT. The complete dimensioned lengths of these vectors are written.

A.2.3.12 TITPLT. - Subroutine TITPLT puts user-defined title information on the time history plots. TITPLT is called at card THPLOT.44 at the beginning of the plot for each maneuver. The origin when TITPLT is called is 3 centimeters (or half inches) left of the title and at the bottom edge of the plot. The paper length can be obtained from common block TOPLOT. The subroutine SYMBL4 (sec. A.2.2.43) may prove useful in subroutine TITPLT.

The standard aircraft subroutine TITPLT is a null routine.

A.2.3.13 UINIT (X, YBIAS, UMODEL). - Subroutine UINIT defines the initial condition of the state and the observation and control biases. UINIT is called at card INIT.28. The outputs of UINIT are X, YBIAS, and UMODEL. X is the initial state. YBIAS is a bias added to the computed observations to compare with the measured observations. UMODEL is the initial control of the model. The program will compute a control bias by subtracting the measured and model controls. The primary data input to UINIT is the measured data in common block BILIN.

When UINIT is called, X is 0, YBIAS is equal to the measured observation, and UMODEL is 0. If these values are unchanged, perturbation equations result. A common alternate choice, assuming the initial state can be suitably defined, is to set YBIAS to 0 and UMODEL to the measured control.

The call to UINIT is controlled by the NAMELIST variable USERIC (ref. 1, sec. 3.3.8(26)). If USERIC is FALSE, perturbation equations are used.

The standard aircraft routine UINIT sets YBIAS to 0 and UMODEL to the measured control. The initial states are defined equal to the corresponding initial measured observations; the measured angle of attack and sideslip are corrected for upwash and vane position in order to define the initial angle of attack and sideslip states. The initial measured velocity is used in the position correction if TIMVAR (ref. 1, sec. 3.3.8(20)) is TRUE; otherwise, the average velocity is used.

A.2.3.14 USERIN (WTFILE). - Subroutine USERIN reads input required by the user routines for each case. It also does any user routine initialization for the case. The argument WTFILE is a logical variable that indicates whether a predicted-derivative file is available. The value of WTFILE should not be changed by subroutine USERIN. USERIN is called at card EDIT.117 after EDIT has defined the defaults for NAMELIST INPUT (ref. 1, sec. 3.3.8) and the channel labels (ref. 1, sec. 3.3.9). Therefore subroutine USERIN can change the defaults defined by subroutine EDIT.

The standard aircraft routine USERIN reads the NAMELIST USER described in reference 1, section 4.3.3. It also changes several of the defaults set in EDIT as described in reference 1, section 4.3.6.

Cards 49 and 50 redefine the USERIC and NREC defaults (ref. 1, sec. 3.3.8(6) and (26)) of subroutine EDIT. Then cards 51 to 99 define the defaults for NAMELIST USER. The defaults defined at cards 71 to 90 are redefined at cards 91 to 98 from the predicted-derivative file if such a file is available. WTCG, defined at card 94, is not used unless a predicted-derivative file is available.

The NAMELIST USER is read at card 101. Card 102 forces SHIFT (ref. 1, sec. 4.3.3(2)) to FALSE if no predicted-derivative data are available, and card 103 forces LATR to be consistent with LONG (ref. 1, sec. 4.3.3(1)). Cards 104 to 106 define the acceleration of gravity depending on METRIC and divide the weight by the acceleration of gravity to obtain the mass.

Cards 107 to 154 redefine the subroutine EDIT defaults for the channel numbers and labels. Most of these defaults depend on whether the case is longitudinal or lateral-directional. These cards also process the UVAR defaults (ref. 1, sec. 4.3.3(22)). Since the UVAR defaults depend on LONG, which is not known until after UVAR is read, the treatment of these defaults is somewhat unusual. UVAR was initially (cards 51 and 52) set to -999. Then cards 116, 117, 136, 153, and 154 apply the defaults to any UVAR elements that were not read in (those that are still -999).

Finally, cards 156 to 165 print out all of the information read by USERIN.

A.2.3.15 WTDEF (MUSE). - Subroutine WTDEF obtains derivative estimates from the predicted-derivative file. WTDEF is called only from the standard aircraft routine MATDEF; therefore, it does not have a general function independent of the standard aircraft routines. The argument, MUSE, is the length of the control vector that will be used in the equations.

Cards 18 to 24 position the predicted-derivative file and read the relevant header information from it. Then cards 25 to 27 compute interpolating factors and indices to interpolate the data to the ALPHA, MACH, and PARAM in common block FLCOND. Cards 28 to 32 compute DO-loop limits dependent on the interpolation indices.

Cards 34 to 80 loop to read and interpolate the predicted-derivative tables. Cards 34 to 39 read a derivative header card and decide whether that derivative will be used. A derivative is not used if the type (LONG or LATR) is wrong or if the matrix where that derivative goes has been read from cards. The function LOADED, used in subroutine MATDEF to determine if a matrix has been read, is not convenient to use here since LOADED does not take the matrix name for its argument. Therefore cards 38 and 39 substitute for the LOADED function. Card 38 calls the function MATNO to obtain a matrix number from the matrix name. Then card 39 uses the matrix number as an index for the vector of input flags stored in common block MATLAB to determine if the matrix has been read. If a derivative is not used, the program goes to statement number 290, which skips that derivative table and jumps back to process the next derivative. Cards 40 to 60 read through a predicted-derivative table and interpolate using the interpolating information computed earlier. Up to eight values

from the table are used in the interpolation (two break points each for ALPHA, MACH, and PARAM). Fewer values may be used if only one break point is used for one or more directions. Card 42 interpolates between the two angle of attack break points at the first Mach number and parameter (param) break point. If two Mach number break points are used, card 46 interpolates between the two angle of attack break points at the second Mach number break point and first param break point; then card 47 interpolates between the two Mach number break points. If two param break points are used, cards 49 to 56 repeat the logic of cards 40 to 47 for the second param break point, and card 57 interpolates between the two param break points. Cards 58 to 60 skip any remaining cards in the predicted-derivative table to position the file at the beginning of the next derivative table. Cards 61 to 77 place the interpolated value in the appropriate matrix location as specified on the predicted-derivative header card.

After all of the values from the predicted-derivative file have been placed in the matrices, card 82 calls subroutine WTTRAN to do any required transformation on the data.

A.2.3.16 WTIN. - Subroutine WTIN reads predicted-derivative data from cards and writes a predicted-derivative file. WTIN is called at card MMLE3.53 after user routine ONCE but before any of the other user routines.

The standard aircraft subroutine WTIN reads the data in the form described in reference 1, section 4.3.2 and writes the predicted-derivative file in the form described in reference 1, section 4.2.2. The primary processing involved is to reorder and expand the simplified data tables read in from cards. The expanded and reordered forms are simplest to interpolate for later use, but it is preferable to allow more flexibility in the input formats.

Cards 27 to 30 read and write the title card for the data set. Cards 19 to 21, 31, and 32 define defaults for the variables in NAMELIST WIND, and card 33 reads the NAMELIST. These data are written on the predicted-derivative file by cards 38 to 42. The break points are read from cards and written on the predicted-derivative file by cards 44 to 49. Optional printed output is done by cards 50 to 61.

Cards 63 to 126 constitute the primary operations of the standard aircraft routine WTIN. These cards read a derivative data table from cards, reorder and expand the table in triply dimensioned array BDAT, and write the data to the predicted-derivative file and the line printer file.

This part of the subroutine repeats for each derivative table until an end card is found. Cards 63 and 64 read a derivative header card and test for the end card. Subroutine ROWCOL is called at card 65 to translate the character string "SUB" into integer row and column numbers. This information was read in as a character string to allow more freedom in the input format than the FORTRAN I format specification. Subroutine ROWCOL is described in section A.2.2.

The derivative functional dependence, FS, is described in reference 1, section 4.3.2.6. Cards 66 to 69 set FS to the default, AMP, if blank was read in. The variables I1, J1, K1, I2, J2, and K2 are used to expand the derivative table. I1 to I2 are the indices of the angle of attack break points to which a single input

value will be copied. If the input data are not a function of angle of attack (i.e., none of the FS values are A), I1 will be 1 and I2 will be NABP so that each data point read will be copied to all of the angle of attack break points. If the input data are a function of angle of attack, I1 and I2 will both start at 1, indicating that the first data point read will be copied only to the first angle of attack break point; for subsequent data points, I1 and I2 will be incremented whenever a new angle of attack break point is being processed. Similarly, J1 and J2 relate to Mach number break points, and K1 and K2 relate to param break points.

Incrementing is done in the order specified by FS. First, the dimension specified by FS(1) is incremented, corresponding to different fields on one input card. After all of the break points of the FS(1) dimension are done, the FS(2) dimension is incremented by 1, and the FS(1) break points are redone for the second FS(2) break point. FS(3) is incremented as the outermost loop. The variable INDEX keeps track of which dimension is being incremented. When all of the data have been read in, the expanded array BDAT is written on the predicted-derivative file and optionally printed. The program then loops back to process the next derivative table.

A.2.3.17 WTTRAN (AXIS, MUSE). - Subroutine WTTRAN does various transformations on the data obtained from the predicted-derivative file. WTTRAN is called only from the standard aircraft routine WTDEF; therefore, it does not have a general function independent of the standard aircraft routines.

The standard aircraft routine WTTRAN transforms longitudinal stability axis derivatives to body axes, computes $C_{N_0}$ and $C_{A_0}$ from total $C_N$ and $C_A$, and transforms moment derivatives from reference to flight center of gravity. The argument AXIS should be either "STAB" or "BODY" to indicate the axis system (stability or body, respectively) of the longitudinal data. The argument MUSE is the number of controls to use in the transformation. If the longitudinal data are already in body axes, the transformation from stability to body axes is skipped. The lateral-directional data are always assumed to be in body axes, regardless of AXIS. The logical variable SHIFT in common block INERTS controls the center of gravity transformation. If SHIFT is FALSE, the moment derivatives are assumed to be already referenced to the actual flight center of gravity, so no transformation of them is done; any difference between the values given for the flight and reference center of gravity is ignored.

Just as in subroutine MATDEF, the LOADED function is used to determine if a matrix has been read in from cards. Subroutine WTTRAN does not change any matrix that was read from cards; its transformations are intended only for defining default matrices using the predicted-derivative file.

Cards 20 to 36 do the transformation from stability to body axes for the longitudinal data. The transformation for the angle of attack derivatives on cards 34 to 36 assumes that total $C_N$ and $C_A$ are in HN(4,1) and HN(5,1). Therefore, this code must be after the transformation of $C_L$ and $C_D$ to $C_N$ and $C_A$ (cards 29 to 32) and before the computation of $C_{N_0}$ and $C_{A_0}$ to replace $C_N$ and $C_A$ (cards 39 to 43).

Cards 39 to 43 compute the extended $C_{N_0}$ and $C_{A_0}$ from total $C_N$ and $C_A$ by subtracting the linearized contributions of the angle of attack and control derivatives. The pitch rate contribution is not subtracted because the total $C_N$ and $C_A$ predictions are assumed to be given for zero pitch rate, even though the maneuver may actually have a significant average pitch rate. Note that the $C_{N_0}$ and $C_{A_0}$ computed are functions of the angle of attack. They are consistent with the usual definition of $C_{N_0}$ and $C_{A_0}$ only if $\alpha$ is 0. We often refer to the program's quantities as $C_{N_0}$ and $C_{A_0}$ extended, to distinguish them from the usual definition.

Center of gravity transformation is done by cards 48 to 55 for lateral-directional data, and cards 58 to 65 for longitudinal data. The longitudinal transformations assume that the derivatives are in body axes; therefore, this code must follow the stability-to-body axis transformation.

## A.2.4 EISPACK Routines

The EISPACK routines used by MMLE3 are BALANC, BALBAK, ELMHES, ELTRAN, HQR, and HQR2. These subroutines are exactly as obtained from Argonne Laboratories and described in reference 6, with one exception. Cards 103 and 104 of HQR and cards 127 and 128 of HQR2 have been modified to increase the maximum number of *QR* iterations from 30 to 50. We have found cases where the routines require more than 30 *QR* iterations to converge to the specified accuracy on CDC computers. The individual subroutines are not described here, as adequate documentation is provided by reference 6. The EISPACK routines are called only from subroutine EIGENG.

The variables RADIX and MACHEP in subroutines BALANC, HQR, and HQR2 are machine dependent. These variables are discussed in section 2.4.

# INDEX OF COMMON DECKS AND SUBROUTINES

# APPENDIX B

## PROGRAM COMSUB

Program COMSUB performs common deck substitution. Users with access to CDC UPDATE (ref. 2) or other similar utility packages will not need to use COMSUB. It reads the UPDATE source cards described in section 1.1, substitutes the common decks in appropriate places, and punches the resulting FORTRAN decks.

No input cards are necessary except for the UPDATE source cards.

The program card and cards 43 to 46 of COMSUB can be altered as desired to define the file numbers for the card reader (UCARD), card punch (UPUNCH), line printer (UPRINT), and UPDATE source cards (UDATA). COMSUB does not use the file UCARD. The file UDATA can be assigned to the card reader, a disk, or tape file, depending on how the source data are available.

The maximum number of common decks allowed is 100, and no more than 200 total cards are allowed for all of the common decks. These dimension limits are easily charged on the dimension statement at card 38 and in the definitions at cards 47 and 48.

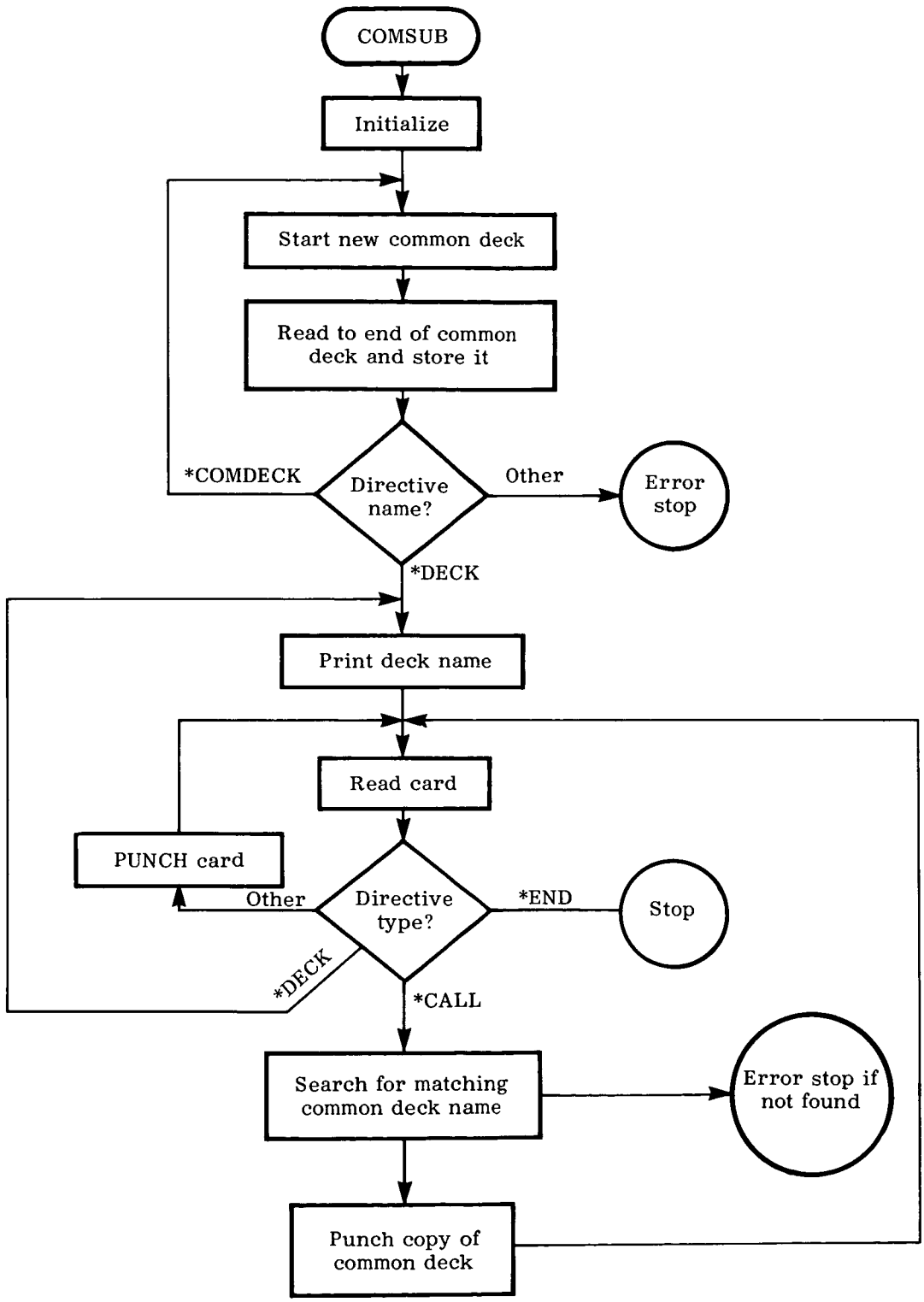
Program COMSUB requires an end-of-file check. The comment cards describe how to configure the program for IBM or CDC end-of-file conventions. Only three cards are affected. Alternately, the program can omit the end-of-file checks and watch for a card with "*END" in the first four columns to flag the end of the source deck. This alternative is not machine dependent, but does require that the *END card be at the end of the source data.

Cards 54 to 60 initialize the common deck counts to 0 and read the first card. This first card must be a *COMDECK card, or the program will do an error stop.

Cards 62 to 69 start a new common deck. The number of common decks is incremented by 1 and checked against the maximum. The common deck name is stored in CNAMES. A pointer in ICARD1 is defined to point to the first column in the matrix CDECKS included in this common deck.

Cards 71 to 79 read a common deck, storing each card in a column of the matrix CDECKS. A dimension limit check is made on the number of columns allowed. Any card with a star in column 1 will be assumed to be an UPDATE directive, and thus will define the end of the common deck.

Cards 81 to 87 finish the storing of a common deck and decide whether more common decks follow. A pointer in ICARD2 is defined to point to the last column in CDECKS included in the common deck. The UPDATE directive that follows the common deck is then examined. If it is a *COMDECK directive, another common deck follows, so the program loops back to read the new common deck. If it is a *DECK directive, the common decks have all been read, and the program continues to the next section. Any directive other than *DECK or *COMDECK will be flagged as an error and the program will stop.


COMSUB
Initialize
Start new common deck
Read to end of common deck and store it
Directive name?
*COMDECK
Other
Error stop
*DECK
Print deck name
Read card
PUNCH card
Other
Directive type?
*END
Stop
*DECK
*CALL
Search for matching common deck name
Error stop if not found
Punch copy of common deck

Cards 89 to 117 read and copy the remaining source cards, substituting common decks as called. Card 91 is used in the IBM version; card 92 is used in the CDC and machine general versions to read a source card. Card 93 is the CDC end-of-file check. When IBM end-of-file is detected, the program jumps to statement 800, and a card still remains to be processed; therefore, the variable LAST is defined at card 119, and the program jumps back to finish processing the last card.

If the card that is read does not have a star in column 1, cards 95 to 97 copy it to the punch file and loop back to read the next card. Cards 94 and 98 to 101 decide on the processing of source cards beginning with a star. If an *END card is found, processing is terminated. If a *DECK card is found, the deck name is printed out, and the card is otherwise ignored. If a *CALL card is found, common deck substitution is done. Any other card beginning with a star is copied to the punch file without special treatment.

Cards 103 to 117 perform common deck substitution when a *CALL card is detected. Cards 104 to 107 search the list of common deck names for the requested common deck. If a matching name is not found, the program stops with an error message. Cards 111 to 117 punch a copy of the common deck and jump back for processing of the next source card. Note that zero length common decks are allowed, in which case no cards are punched for them.

```
      PROGRAM COMSUB(INPUT=258,PUNCH=512,OUTPUT=258,DATA=512,                  COMSUB   2
     -     TAPE1=INPUT,TAPE2=PUNCH,TAPE3=OUTPUT,TAPE4=DATA)                    COMSUB   3
C                                                                              COMSUB   4
C         RICHARD E. MAINE                9 JAN 79                           COMSUB   5
C     PROGRAM TO CREATE FORTRAN SOURCE DECKS FROM UPDATE SOURCE DECKS.         COMSUB   6
C     READS IN COMMON DECKS FOLLOWED BY REGULAR DECKS.                         COMSUB   7
C     SUBSTITUTES COMMON DECKS INTO REGULAR DECKS AS CALLED FOR.               COMSUB   8
C     PUNCHES OUT RESULTING SUBSTITUTED DECKS.                                 COMSUB   9
C                                                                              COMSUB  10
C     END-OF-FILE IS CHECKED BY ONE OF THREE METHODS.                          COMSUB  11
C        1) CDC TYPE EOF FUNCTION (INDICATES NO DATA WAS RETURNED ON           COMSUB  12
C           PREVIOUS READ)                                                     COMSUB  13
C           FOR THIS TYPE CHECK, THE CARD FLAGGED IBM IN COL. 73 SHOULD        COMSUB  14
C           HAVE A C IN COL. 1 SINCE CDC COMPILERS DO NOT RECOGNIZE            COMSUB  15
C           THE SYNTAX OF THIS CARD. THE CARDS FLAGGED WITH CDC IN             COMSUB  16
C           COL. 73 SHOULD LEAVE COL. 1 BLANK.                                 COMSUB  17
C        2) IBM TYPE END= PARAMETER (BRANCHES IF DATA JUST RETURNED WAS         COMSUB  18
C           LAST RECORD IN FILE)                                               COMSUB  19
C           FOR THIS TYPE CHECK, THE CARDS FLAGGED CDC IN COL. 73 SHOULD       COMSUB  20
C           HAVE A C IN COL. 1 AND THE CARD FLAGGED WITH IBM IN COL. 73        COMSUB  21
C           SHOULD BE BLANK IN COL. 1.                                         COMSUB  22
C        3) FOR ANY MACHINE - PROGRAM CHECKS FOR A DATA CARD CONTAINING        COMSUB  23
C           *END STARTING IN COL. 1.                                           COMSUB  24
C           THIS CHECK IS ALWAYS VALID, BUT REQUIRES THE APPROPRIATE           COMSUB  25
C           CARD TO BE AT THE END OF THE DATA DECK. THE PROGRAM CHECKS         COMSUB  26
C           FOR THIS CONDITION IN EITHER THE CDC OR IBM CONFIGURATIONS.        COMSUB  27
C           FOR A MACHINE GENERAL PROGRAM THAT USES ONLY THIS CHECK,           COMSUB  28
C           PUT A C IN COL. 1 OF CARDS FLAGGED WITH EITHER CDC OR IBM          COMSUB  29
C           IN COL. 73 XCEPT FOR THE ONE CARD ALSO FLAGGED WITH ANY.           COMSUB  30
C                                                                              COMSUB  31
C     LIMITATIONS, DIMENSION LIMITS                                            COMSUB  32
C          MAX NUMBER OF COMMON DECKS IS 100 (NCOMMX)                          COMSUB  33
C          MAX TOTAL NUMBER OF CARDS IN COMMON DECKS IS 200 (NCRDMX)           COMSUB  34
C                                                                              COMSUB  35
      INTEGER UCARD,UPUNCH,UPRINT,UDATA                                        COMSUB  36
      LOGICAL LAST                                                             COMSUB  37
      DIMENSION CDECKS(21,200),CNAMES(2,100),ICARD1(100),ICARD2(100),          COMSUB  38
     -     COMCRD(21),CARD(22)                                                 COMSUB  39
      DATA STAR/1H*/,COMD,ECK/4HCOMD,3HECK/,DECK/4HDECK/,CALL/4HCALL/,         COMSUB  40
     -     END/3HEND/                                                          COMSUB  41
C                                                                              COMSUB  42
      UCARD = 1                                                                COMSUB  43
      UPUNCH = 2                                                               COMSUB  44
      UPRINT = 3                                                               COMSUB  45
      UDATA = 4                                                                COMSUB  46
      NCRDMX = 200                                                             COMSUB  47
      NCOMMX = 100                                                             COMSUB  48
      NCPUN = 0                                                                COMSUB  49
      LAST = .FALSE.                                                           COMSUB  50
      WRITE(UPRINT,3000)                                                       COMSUB  51
C     ******************** READ COMMON DECKS.                                  COMSUB  52
C     ********** INITIALIZE AND READ FIRST CARD.                              COMSUB  53
      NCOMS = 0                                                                COMSUB  54
      NCARD = 0                                                                COMSUB  55
      READ(UDATA,1001) COMCRD                                                  COMSUB  56
      IF(COMCRD(1).EQ.STAR .AND.                                               COMSUB  57
     -     COMCRD(2).EQ.COMD .AND. COMCRD(3).EQ.ECK)  GO TO 100                COMSUB  58
```

```
      WRITE(UPRINT,3006) COMCRD                                                  COMSUB  59
      STOP                                                                       COMSUB  60
C     ********** START NEW COMMON DECK.                                          COMSUB  61
  100 NCOMS = NCOMS+1                                                            COMSUB  62
      ICARD1(NCOMS) = NCARD+1                                                    COMSUB  63
      WRITE(UPRINT,3001) NCOMS,COMCRD(4),COMCRD(5),ICARD1(NCOMS)                 COMSUB  64
      IF(NCOMS.LE.NCOMMX) GO TO 110                                              COMSUB  65
      WRITE(UPRINT,3002) NCOMMX                                                  COMSUB  66
      STOP                                                                       COMSUB  67
  110 CNAMES(1,NCOMS) = COMCRD(4)                                                COMSUB  68
      CNAMES(2,NCOMS) = COMCRD(5)                                                COMSUB  69
C     ********** READ TO END OF COMMON DECK.                                     COMSUB  70
  200 READ(UDATA,1001) COMCRD                                                    COMSUB  71
      IF(COMCRD(1).EQ.STAR) GO TO 300                                            COMSUB  72
      NCARD = NCARD+1                                                            COMSUB  73
      IF(NCARD.LE.NCRDMX) GO TO 210                                              COMSUB  74
      WRITE(UPRINT,3003) NCRDMX                                                  COMSUB  75
      STOP                                                                       COMSUB  76
  210 DO 220 I=1,21                                                              COMSUB  77
  220 CDECKS(I,NCARD) = COMCRD(I)                                                COMSUB  78
      GO TO 200                                                                  COMSUB  79
C     ********** END OF COMMON DECK.                                             COMSUB  80
  300 ICARD2(NCOMS) = NCARD                                                      COMSUB  81
      IF(COMCRD(2).EQ.COMD .AND. COMCRD(3).EQ.ECK) GO TO 100                     COMSUB  82
      IF(COMCRD(2).EQ.DECK) GO TO 350                                            COMSUB  83
      WRITE(UPRINT,3004) COMCRD                                                  COMSUB  84
      STOP                                                                       COMSUB  85
  350 CARD(4) = COMCRD(3)                                                        COMSUB  86
      CARD(5) = COMCRD(4)                                                        COMSUB  87
C     ********** READ AND COPY DECKS.                                            COMSUB  88
  400 WRITE(UPRINT,3005) CARD(4),CARD(5)                                         COMSUB  89
  420 IF(LAST) GO TO 900                                                         COMSUB  90
C     READ(UDATA,1002,END=800) CARD                                  IBM       COMSUB  91
      READ(UDATA,1002) CARD                                          CDC ANY   COMSUB  92
      IF(EOF(UDATA).NE.0.) GO TO 900                                 CDC       COMSUB  93
  430 IF(CARD(1).EQ.STAR) GO TO 450                                              COMSUB  94
  440 NCPUN = NCPUN+1                                                            COMSUB  95
      WRITE(UPUNCH,1002) CARD                                                    COMSUB  96
      GO TO 420                                                                  COMSUB  97
  450 IF(CARD(2).EQ.END) GO TO 900                                               COMSUB  98
      IF(CARD(2).EQ.DECK) GO TO 400                                              COMSUB  99
      IF(CARD(2).EQ.CALL) GO TO 500                                              COMSUB 100
      GO TO 440                                                                  COMSUB 101
C     ********** SEARCH FOR CALLED COMMON DECK.                                  COMSUB 102
  500 WRITE(UPRINT,3006) CARD(4),CARD(5)                                         COMSUB 103
      DO 550 ICOM=1,NCOMS                                                        COMSUB 104
      IF(CARD(4).EQ.CNAMES(1,ICOM) .AND. CARD(5).EQ.CNAMES(2,ICOM))              COMSUB 105
     -     GO TO 600                                                           COMSUB 106
  550 CONTINUE                                                                   COMSUB 107
      WRITE(UPRINT,3007)                                                         COMSUB 108
      STOP                                                                       COMSUB 109
C     ********** PUNCH CALLED COMMON DECK.                                       COMSUB 110
  600 NCCRDS = ICARD2(ICOM)-ICARD1(ICOM)+1                                       COMSUB 111
      IF(NCCRDS.EQ.0) GO TO 420                                                  COMSUB 112
      NCPUN = NCPUN+NCCRDS                                                       COMSUB 113
      ICRD1 = ICARD1(ICOM)                                                       COMSUB 114
      ICRD2 = ICARD2(ICOM)                                                       COMSUB 115
```

```
      WRITE(UPUNCH,4001) ((CDECKS(J,I),J=1,21),I=ICRD1,ICRD2)          COMSUB 116
      GO TO 420                                                        COMSUB 117
C     ********************* END-OF-FILE PROCESSING.                    COMSUB 118
  800 LAST = .TRUE.                                                    COMSUB 119
      GO TO 430                                                        COMSUB 120
  900 WRITE(UPRINT,3010) NCPUN                                         COMSUB 121
C                                                                      COMSUB 122
 1001 FORMAT(A1,19A4,A3)                                               COMSUB 123
 1002 FORMAT(A1,A4,A1,18A4,A2)                                         COMSUB 124
 3000 FORMAT("1COMSUB PROGRAM FOR INSERTING COMMON DECKS"/)            COMSUB 125
 3001 FORMAT(" COMMON DECK",I4,1X,2A4," STARTS AT CARD",I5,".")        COMSUB 126
 3002 FORMAT("0*** ERROR. LIMIT OF",I5," COMMON DECKS REACHED.")       COMSUB 127
 3003 FORMAT("0**** ERROR. LIMIT OF",I6," TOTAL CARDS IN COMMON DECKS ",COMSUB 128
     -     "REACHED.")                                                 COMSUB 129
 3004 FORMAT("0**** ERROR. DIRECTIVE NOT RECOGNIZED - ",A1,19A4,A3)    COMSUB 130
 3005 FORMAT("0DECK ",2A4," BEING PROCESSED.")                         COMSUB 131
 3006 FORMAT(" CALLING FOR COMMON DECK ",2A4)                          COMSUB 132
 3007 FORMAT("0*** ERROR. ABOVE COMMON DECK NOT RECOGNIZED.")          COMSUB 133
 3008 FORMAT("0*** ERROR. FIRST CARD IS NOT A VALID COMDECK CARD."/    COMSUB 134
     -     1X,A1,19A4,A2)                                              COMSUB 135
 3010 FORMAT("0TOTAL OF",I6," CARDS PUNCHED.")                         COMSUB 136
      STOP                                                             COMSUB 137
      END                                                              COMSUB 138
```

# APPENDIX C

## PROGRAM COMPUN

Program COMPUN punches common decks for the MMLE3 program, substituting in the desired physical dimensions. Used in conjunction with program COMSUB (appendix B) or CDC UPDATE (ref. 2), COMPUN allows the physical dimensions of the MMLE3 program to be easily changed as discussed in section 3.2.

The file numbers used by COMPUN can be altered by changing the program card and cards 61 to 64. The input to COMPUN is on two separate files. The card reader file (UCARD) contains the NAMELIST IN, which defines the physical dimensions to be used. The variables in NAMELIST IN are described in the comment cards.

The file UDATA contains a template for the common decks. UDATA can be assigned to the same file as UCARD, in which case the template would follow the NAMELIST in the input cards. The template consists of the common decks (complete with *COMDECK cards), except that symbols are used for the dimensions. COMPUN will copy the template to the punch file, with the appropriate values substituted for the symbols.

Each symbol used in the template consists of a star followed by a two-character name. The two-character names used are described in the comment cards. COMPUN will substitute a two-digit value for the two-character name and a blank for the star. If more than two digits may be required for the value, a star should be placed after the two-character name in the template as well as before it. A four-digit value will then be substituted in the locations of the two-character name and the two stars. The template must have no stars after column 8 except for those used as described above. The end of the template is indicated by a card with "*END" starting in column 1.

Cards 66 to 79 define the default dimensions and then read and write the NAMELIST. The default values for the dimensions are the values used with the MMLE3 program as supplied. Cards 81 to 87 check several limitations on the allowable dimensions. These limitations are described in the comment cards. Cards 89 to 94 define dimensions computed from the basic dimensions read in NAMELIST IN. The loop at cards 97 to 107 converts each of the dimension values to four Hollerith digits.

Cards 110 and 111 read a card of the template and check to see if it is the *END card. Cards 120 to 134 insert two- or four-digit values in the places specified by the template.

The program listing for COMPUN is shown below, followed by a listing of the template.

```
      PROGRAM COMPUN(COMPUN,RESIZE,INPUT,OUTPUT,                        COMPUN   2
     -     TAPE4=COMPUN,TAPE5=RESIZE,                                   COMPUN   3
     -     TAPE1=INPUT,TAPE3=OUTPUT)                                    COMPUN   4
C                                                                       COMPUN   5
C     PUNCHES NMLE3  COMMON DECKS OR UPDATES                            COMPUN   6
C     PROGRAM LAST MODIFIED 15 AUG 79   RICHARD MAINE                   COMPUN   7
C                                                                       COMPUN   8
C     INPUT VARIABLES IN NAMELIST /IN/                                  COMPUN   9
C      (CORRESPONDING 2 CHARACTER NAMES IN PARENS).                     COMPUN  10
C          MAXX - MAX NO OF STATES                          (X )        COMPUN  11
C          MAXZ - MAX NO OF OBSERVATIONS                    (Z )        COMPUN  12
C          MAXU - MAX NO OF CONTROL INPUTS                  (U )        COMPUN  13
C          MAXB - MAX NO OF BIAS INPUTS                     (B )        COMPUN  14
C          LEX - MAX NO OF EXTRA SIGNALS                    (EX)        COMPUN  15
C          NI - MAX NO OF INDEPENDENT UNKNOWNS + 2          (NI)        COMPUN  16
C          MAXTV - MAX NO OF UNKNOWNS AND CONSTRAINTS       (TV)        COMPUN  17
C          MAXKV - MAX NO OF UNKNOWNS AFFECTING K           (KV)        COMPUN  18
C          MAXHRD - MAX NO OF HARD CONSTRAINTS + 1          (HD)        COMPUN  19
C          MAXSFT - MAX NO OF SOFT CONSTRAINTS + 1          (SF)        COMPUN  20
C          NTPLT - MAX TIME POINTS FOR PLOTTING + 2         (TP)        COMPUN  21
C          NSP - NO OF SIGNALS IN CORE FOR PLOTTING         (S )        COMPUN  22
C                                                                       COMPUN  23
C     COMPUTED VARIABLES AND 2 CHARACTER NAMES:                         COMPUN  24
C          (X1) = MAXX+1                                                COMPUN  25
C          (Z1) = MAXZ+1                                                COMPUN  26
C          (UX) = MAXU+LEX                                              COMPUN  27
C          (PL) = MAXX+MAXU+LEX                                         COMPUN  28
C          (S1) = NSP+1                                                 COMPUN  29
C          (S2) = 2*NSP                                                 COMPUN  30
C                                                                       COMPUN  31
C     LIMITATIONS                                                       COMPUN  32
C     BECAUSE OF PROGRAM ASSUMPTIONS IN USING SCRATCH MATRICES.         COMPUN  33
C          MAXZ .GE. MAXX                                               COMPUN  34
C     BECAUSE OF MINIMUM SPACE REQUIRED FOR MATRIX STORAGE CONVENTIONS. COMPUN  35
C          MAXX .GE. 4                                                  COMPUN  36
C          MAXZ .GE. 4                                                  COMPUN  37
C          MAXU .GE. 4                                                  COMPUN  38
C          MAXB .GE. 4                                                  COMPUN  39
C          NI .GE. 4                                                    COMPUN  40
C     FOR STATE NOISE CASE ONLY, IN SUBROUTINE KALMAN;                  COMPUN  41
C     COMPUN PRINTS A WARNING FOR VIOLATING THIS, BUT DOES NOT STOP.    COMPUN  42
C          NI .GE. 4*MAXX+4                                             COMPUN  43
C                                                                       COMPUN  44
      INTEGER UCARD,UPRINT,UDATA,UPUNCH                                 COMPUN  45
      INTEGER A(66),B(2,20),V(4,20),VAL(20),NUM(10)                     COMPUN  46
      INTEGER BLANK,END,STAR                                            COMPUN  47
      EQUIVALENCE (MAXX,VAL(1)),(MAXZ,VAL(2)),(MAXU,VAL(3)),            COMPUN  48
     -     (MAXB,VAL(4)),(LEX,VAL(5)),(NI,VAL(6)),(MAXTV,VAL(7)),       COMPUN  49
     -     (MAXKV,VAL(8)),(MAXHRD,VAL(9)),(MAXSFT,VAL(10)),             COMPUN  50
     -     (NTPLT,VAL(11)),(NSP,VAL(12))                                COMPUN  51
      DATA STAR/1H*/,BLANK/1H /,END/4H*END/,                            COMPUN  52
     -     NUM/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/,                COMPUN  53
     -     B/1HX,1H ,1HZ,1H ,1HU,1H ,1HB,1H ,1HE,1HX,1HN,1HI,1HT,1HV,   COMPUN  54
     -     1HK,1HV,1HH,1HD,1HS,1HF,1HT,1HP,1HS,1H ,1HX,1H1,1HZ,1H1,     COMPUN  55
     -     1HU,1HX,1HP,1HL,1HS,1H1,1HS,1H2,4*1H:/                       COMPUN  56
      NAMELIST /IN/ MAXX,MAXZ,MAXU,MAXB,LEX,NI,MAXTV,MAXKV,MAXHRD,      COMPUN  57
     -     MAXSFT,NTPLT,NSP                                             COMPUN  58
```

```
C                                                                       COMPUN  59
C     ****************************** FILE NUMBERS.                    COMPUN  60
      UCARD = 1                                                         COMPUN  61
      UPRINT = 3                                                        COMPUN  62
      UDATA = 4                                                         COMPUN  63
      UPUNCH = 5                                                        COMPUN  64
C     ****************************** DEFAULTS AND INPUT.              COMPUN  65
      MAXX = 7                                                          COMPUN  66
      MAXZ = 8                                                          COMPUN  67
      MAXU = 4                                                          COMPUN  68
      MAXB = 4                                                          COMPUN  69
      LEX = 20                                                          COMPUN  70
      NI = 35                                                           COMPUN  71
      MAXTV = 50                                                        COMPUN  72
      MAXKV = 15                                                        COMPUN  73
      MAXHRD = 36                                                       COMPUN  74
      MAXSFT = 11                                                       COMPUN  75
      NTPLT = 1202                                                      COMPUN  76
      NSP = 3                                                           COMPUN  77
      READ(UCARD,IN)                                                    COMPUN  78
      WRITE(UPRINT,IN)                                                  COMPUN  79
C     ********************* CHECK LIMITATIONS.                        COMPUN  80
      IF(MAXX.GT.MAXZ) GO TO 260                                        COMPUN  81
      IF(MAXX.LT.4) GO TO 200                                           COMPUN  82
      IF(MAXZ.LT.4) GO TO 200                                           COMPUN  83
      IF(MAXU.LT.4) GO TO 200                                           COMPUN  84
      IF(MAXB.LT.4) GO TO 200                                           COMPUN  85
      IF(NI  .LT.4) GO TO 200                                           COMPUN  86
      IF(NI.LT.4*MAXX+4) WRITE(UPRINT,3004)                             COMPUN  87
C     ****************************** DEFINE COMPUTED QUANTITIES.      COMPUN  88
      VAL(13) = MAXX+1                                                  COMPUN  89
      VAL(14) = MAXZ+1                                                  COMPUN  90
      VAL(15) = MAXU+LEX                                                COMPUN  91
      VAL(16) = MAXX+MAXU+LEX                                           COMPUN  92
      VAL(17) = NSP+1                                                   COMPUN  93
      VAL(18) = 2*NSP                                                   COMPUN  94
      NB = 18                                                           COMPUN  95
C     ****************************** CONVERT TO CHARACTER.            COMPUN  96
      DO 10 I=1,NB                                                      COMPUN  97
      I1=VAL(I)/1000                                                    COMPUN  98
      J=VAL(I)-I1*1000                                                  COMPUN  99
      I2=J/100                                                          COMPUN 100
      J=J-I2*100                                                        COMPUN 101
      I3=J/10                                                           COMPUN 102
      I4=J-I3*10                                                        COMPUN 103
      V(1,I)=NUM(I1+1)                                                  COMPUN 104
      V(2,I)=NUM(I2+1)                                                  COMPUN 105
      V(3,I)=NUM(I3+1)                                                  COMPUN 106
   10 V(4,I)=NUM(I4+1)                                                  COMPUN 107
C                                                                       COMPUN 108
C     ****************************** READ TEMPLATE.                   COMPUN 109
  100 READ(UDATA,1001) A                                                COMPUN 110
      IF(A(1).EQ.END)  GO TO 500                                        COMPUN 111
      I = 2                                                             COMPUN 112
  110 I=I+1                                                             COMPUN 113
      IF(I.LT.64) GO TO 120                                             COMPUN 114
      WRITE(UPUNCH,1001) A                                              COMPUN 115
```

```
      WRITE(UPRINT,2002) A                                              COMPUN 116
      GO TO 100                                                         COMPUN 117
  120 IF(A(I).NE.STAR) GO TO 110                                        COMPUN 118
C     ********************  INSERT DIMENSIONS WHERE INDICATED.          COMPUN 119
      DO 130 J=1,NB                                                     COMPUN 120
      IF(A(I+1).NE.B(1,J) .OR. A(I+2).NE.B(2,J)) GO TO 130              COMPUN 121
      IF(A(I+3).EQ.STAR) GO TO 125                                      COMPUN 122
      A(I)=BLANK                                                        COMPUN 123
      A(I+1)=V(3,J)                                                     COMPUN 124
      A(I+2)=V(4,J)                                                     COMPUN 125
      I=I+3                                                             COMPUN 126
      GO TO 110                                                         COMPUN 127
  125 A(I)=V(1,J)                                                       COMPUN 128
      A(I+1)=V(2,J)                                                     COMPUN 129
      A(I+2)=V(3,J)                                                     COMPUN 130
      A(I+3)=V(4,J)                                                     COMPUN 131
      I=I+4                                                             COMPUN 132
      GO TO 110                                                         COMPUN 133
  130 CONTINUE                                                          COMPUN 134
C     ******************************** ERROR EXIT.                      COMPUN 135
      WRITE(UPRINT,2001) A(I+1),A(I+2),I,A                              COMPUN 136
      GO TO 400                                                         COMPUN 137
  200 WRITE(UPRINT,3003)                                                COMPUN 138
  400 DO 450 I=1,10000                                                  COMPUN 139
  450 READ(UCARD,1001) A                                                COMPUN 140
C     ******************************** NORMAL EXIT.                     COMPUN 141
  500 REWIND UPUNCH                                                     COMPUN 142
C                                                                       COMPUN 143
 1001 FORMAT(2A4,64A1)                                                  COMPUN 144
 2001 FORMAT("0ERROR UNKNOWN VARIABLE ",2A1," IN COLUMN",I3,"+7"/       COMPUN 145
     -      1X,2A4,64A1)                                               COMPUN 146
 2002 FORMAT(1X,2A4,64A1)                                               COMPUN 147
 3003 FORMAT("0**** ERROR. INPUT VARIABLES NOT WITHIN ALLOWED LIMITS.", COMPUN 148
     -      " SEE PROGRAM LISTING FOR LIMITS.")                        COMPUN 149
 3004 FORMAT("0***** WARNING. NI IS TOO SMALL FOR STATE NOISE CASE.")   COMPUN 150
      STOP                                                              COMPUN 151
      END                                                               COMPUN 152
```

```
*COMDECK HISTORY
C     MODIFICATION HISTORY:
C     END OF MODIFICATIONS.
*COMDECK AMCOM
      COMMON /AMCOM/ BCONST(*TV)
*COMDECK AVGCOM
      COMMON /AVGCOM/ ZAVG(*Z ),UAVG(*U ),EXAVG(*EX),ZSIG(*Z ),
     -     USIG(*U ),EXSIG(*EX),ZMINM(*Z ),UMINM(*U ),EXMINM(*EX),
     -     ZMAXM(*Z ),UMAXM(*U ),EXMAXM(*EX)
*COMDECK BIASES
      COMMON /BIASES/ UOFF(*U ),YOFF(*Z )
*COMDECK BILIN
      COMMON /BILIN/ USEAVG,TIMVAR,Z(*Z ),U(*U ),EXTRA(*EX),ONES(*B )
      LOGICAL USEAVG,TIMVAR
*COMDECK COM
      COMMON /COM/ NCASE,NPTT,NPTS(15),ITMSTS(15)
*COMDECK CRMAT
      COMMON /CRMAT/ AC(*X1,*X ),BC(*X1,*U ),SC(*X1,*B ),RC(*X1,*X ),
     -     CC(*Z1,*X ),DC(*Z1,*U ),HC(*Z1,*B ),EC(*Z1,*X ),FC(*X1,*X )
*COMDECK DETERM
      COMMON /DETERM/ NVAR,IMAT(*TV),IROW(*TV),ICOL(*TV),ILOC(*TV),
     -     ACONST(*TV)
*COMDECK DIMMAT
      COMMON /DIMMAT/ A (*X1,*X ),B (*X1,*U ),S (*X1,*B ),R (*X1,*X ),
     -     C (*Z1,*X ),D (*Z1,*U ),H (*Z1,*B ),E (*Z1,*X )
*COMDECK DUMCOM
      COMMON /DUMCOM/ DUM(*X1,*X ),DUM2(*X1,*X ),DUM3(*X1,*X )
*COMDECK DUMVEC
      COMMON /DUMVEC/ DUMX(*X ),DUM2X(*X )
*COMDECK ECOM
      COMMON /ECOM/ RIA(*X1,*X ),RIB(*X1,*U ),RIS(*X1,*B ),RI(*X1,*X )
*COMDECK ERLIST
      COMMON /ERLIST/ BLOWUP,NITER,ERRVEC(50)
      LOGICAL BLOWUP
*COMDECK FCOM
      COMMON /FCOM/ F(*X1,*X )
*COMDECK FILES
      COMMON /FILES/ UCARD,UPUNCH,UPRINT,UDATA,UT1,UT2,UTHOUT,UWT,UPLOT
      INTEGER UCARD,UPUNCH,UPRINT,UDATA,UT1,UT2,UTHOUT,UWT,UPLOT
*COMDECK GICOM
      COMMON /GICOM/ ITG,DIAGG,FREQCR,RLXG,FC1,FC2,ERRFLT,SGNLS,
     -     GGI(*Z1,*Z ),RSQ(*Z1,*Z ),FRSQ(*Z1,*Z ),WRSQ(*Z ),WFRSQ(*Z )
      LOGICAL DIAGG
*COMDECK GRADS
      COMMON /GRADS/ GRADX(*X1,*NI),GRADY(*Z1,*NI),GRAD1(*Z1,*NI)
*COMDECK GRADS
      DIMENSION XDT2(*X ),XDT12(*X ),XT12(*X )
*COMDECK GRDCOM
      COMMON /GRDCOM/ DK(*X1,*Z ,*KV)
*COMDECK GRSIZE
      COMMON /GRSIZE/ JKMM1,NK
*COMDECK HEADNG
      COMMON /HEADNG/ TITLE(20),ADATE,ATIME,
     -     SIGLAB(2,*Z ),XLAB(2,*X ),CONLAB(2,*U ),EXLAB(2,*EX)
*COMDECK ICOND
      COMMON /ICOND/ USERIC,VARICS,VARIC(*X ),DX1C(*X )
      LOGICAL USERIC,VARICS,VARIC
*COMDECK INMAT
      COMMON /INMAT/ ALAB,II,JJ,IM
*COMDECK INOPT
      COMMON /INOPT/ CARD,TAPE
      LOGICAL CARD,TAPE
*COMDECK INORD
      COMMON /INORD/ NREC,ZCHAN(*Z ),UCHAN(*U ),EXCHAN(*EX)
      INTEGER ZCHAN,UCHAN,EXCHAN
*COMDECK INTEGR
      COMMON /INTEGR/ DT,NEAT
*COMDECK KCOM
      COMMON /KCOM/ P(*X1,*X ),KGAIN(*X1,*Z )
      REAL KGAIN
*COMDECK MAPCOM
      COMMON /MAPCOM/ MAPUK(*TV),MAPKG(*KV)
*COMDECK MATIN
      COMMON /MATIN/  AV(*X1,*X ),BV(*X1,*U ),SV(*X1,*B ),RV(*X1,*X ),
     -     CV(*Z1,*X ),DV(*Z1,*U ),HV(*Z1,*B ),EV(*Z1,*X ),FV(*X1,*X ),
     -     APRA(*X1,*X ),APRB(*X1,*U ),APRS(*X1,*B ),APKR(*X1,*X ),
     -     APRC(*Z1,*X ),APKD(*Z1,*U ),APRH(*Z1,*B ),APRE(*Z1,*X ),
     -     APRF(*X1,*X ),HAKD(*HD,7)
```

```
*COMDECK MATRAT
      COMMON /MATRAT/ AM(*X1,*X ),BM(*X1,*U ),SM(*X1,*B ),RM(*X1,*X ),
     -     CM(*Z1,*X ),DM(*Z1,*U ),HM(*Z1,*B ),EM(*Z1,*X )
*COMDECK MATLAB
      COMMON /MATLAB/ NMATS,LAB(31),INFLAG(31)
      REAL LAB
*COMDECK MATRIX
      COMMON /MATRIX/ AN(*X1,*X ),BN(*X1,*U ),SN(*X1,*B ),RN(*X1,*X ),
     -     CN(*Z1,*X ),DN(*Z1,*U ),HN(*Z1,*B ),EN(*Z1,*X )
*COMDECK MAXCON
      COMMON /MAXCON/ MAXHRD,MAXSFT
*COMDECK MAXIM
      COMMON /MAXIM/ MAXX1,MAXZ1,NI,MAXIV,MAXKV
*COMDECK MAXIMS
      COMMON /MAXIMS/ MAXX,MAXZ,MAXU,MAXB,LEX,LOKD
*COMDECK MODCOM
      COMMON /MODCOM/ UMOD
      LOGICAL UMOD
*COMDECK OBSRV
      COMMON /OBSRV/ ERIAC(*Z1,*X ),ERIBD(*Z1,*U ),ERISH(*Z1,*B ),
     -     ERI(*Z1,*X )
*COMDECK OUTOPT
      COMMON /OUTOPT/ PRINTX,PRINTY,PRINTD,PLOTEM,PUNCH,TEST,PLTMAX,
     -     ERRTH
      LOGICAL PRINTX,PRINTY,PRINTD,PLOTEM,PUNCH,TEST,ERRTH
*COMDECK PBCOM
      COMMON /PBCOM/ PR(*NI)
*COMDECK PHICOM
      COMMON /PHICOM/ PHI(*X1,*X ),PSI(*X1,*X ),PSIB(*X1,*U ),
     -     PSIS(*X1,*B )
*COMDECK RECRD
      COMMON /RECRD/ EOFTH,T(4),RECORD(100)
      LOGICAL EOFTH
      INTEGER T
*COMDECK RICCOM
      COMMON /RICCOM/ DUMXZ(*X1,*Z ),DUMZX(*Z1,*X ),
     -     RIAP(*X1,*X ),CTG(*X1,*Z ),RIF(*X1,*X ),RIFRIF(*X1,*X )
*COMDECK SIZE
      COMMON /SIZE/ MX,MZ,MU,MB
*COMDECK SOFCOM
      COMMON /SOFCOM/ SOFT(*SF,7)
*COMDECK SUMCOM
      COMMON /SUMCOM/ JKM,SUM(*NI,*NI)
*COMDECK SUMSAV
      COMMON /SUMSAV/ DIAGON(*NI),APRDIF(*NI),NAPR,ITAPR
*COMDECK TAPPOS
      COMMON /TAPPOS/ ITM,REW
      LOGICAL REW
*COMDECK THPLOTS
      DIMENSION Z(*Z ),ZZ(*Z ),DC(*PL),IPLT(*PL),VMINS(*S2),VMAXS(*S2),
     -     TIME(*TP*),XXX(*TP*,*S2),X(*TP*,*S ),XX(*TP*,*S )
      EQUIVALENCE (X(1,1),XXX(1,1)),(XX(1,1),XXX(1,*S1))
C
      NTPLT=*TP*
      NCH=*S
*COMDECK TODATA
      COMMON /TODATA/ STC(15),ETC(15),THIN,PRINTI,MAXREC,
     -     ZBIAS(*Z ),UBIAS(*U ),EXBIAS(*EX),ZSCALE(*Z ),USCALE(*U ),
     -     EXSCAL(*EX)
      INTEGER THIN,STC,ETC
      LOGICAL PRINTI
*COMDECK TOGIRL
      COMMON /TOGIRL/ BOUND,ERRMAX,FULL1,NOITER,DFAC,ITDFAC,SNOISE
      LOGICAL FULL1,SNOISE
*COMDECK TOGRAD
      COMMON /TOGRAD/ XT1(*X ),XT2(*X ),XH2(*X ),
     -     V1(*X ),V2(*X ),V12(*X ),U1(*U ),U2(*U ),U12(*U ),
     -     Y(*Z ),ZMY2(*Z ),ZMYFLT(*Z ),W(*Z )
*COMDECK TOPLOT
      COMMON /TOPLOT/ZMAX(*Z ),ZMIN(*Z ),XMAX(*X ),UMAX(*U ),EXMAX(*EX),
     -     XMIN(*X ),UMIN(*U ),EXMIN(*EX),XPLOT(*X ),NUPLT,NEXPLT,
     -     TIMESC,RATIO,PAPER
      LOGICAL XPLOT
```

C

```
*COMDECK VARDEFS
      MAXX = *X
      MAXZ = *Z
      MAXU = *U
      MAXB = *B
      LEX = *EX
      NI = *NI
      MAXTV = *TV
      MAXKV = *KV
      MAXHRD = *HD
      MAXSFT = *SF
*COMDECK XSUMS
      COMMON /XSUMS/ XSUM(*X ),X2SUM(*X )
*COMDECK FLCOND
      COMMON /FLCOND/ CBAR,V,THETA,PHI,ALPHA,MACH,PARAM,CG,G
      REAL MACH
*COMDECK GFDEFS
      COMMON /GFDEFS/ GGILAT(*Z1,*Z ),GGILLN(*Z1,*Z ),FLAT(*X1,*X ),
     -     FLON(*X1,*X )
*COMDECK GRAV
      COMMON /GRAV/ DGDT,DGDP
*COMDECK INERTS
      COMMON /INERTS/ IX,IY,IZ,IXZ,IXE,MASS,AREA,CHORD,SPAN,WTCG,SHIFT
      REAL IX,IY,IZ,IXZ,IXE,MASS
      LOGICAL SHIFT
*COMDECK INSTR
      COMMON /INSTR/ KALF,KB,XALF,XR,XAN,XAX,XAY,YALF,YB,YAN,YAX,YAY,
     -     ZALF,ZB,ZAN,ZAX,ZAY,DCGFT
      REAL KALF,KB
*COMDECK LONLAT
      COMMON /LONLAT/ LONG,LATR
      LOGICAL LONG,LATR
*COMDECK UVCOM
      COMMON /UVCOM/ UVAR(*U )
      INTEGER UVAR
*END
```

# APPENDIX D

# TEST CASES

Four test cases are provided for the MMLE3 program. These test cases were chosen to illustrate and check out several features of the program. This appendix contains descriptions of the test cases. The input cards and output listings are shown in supplement 2 on microfiche. Slight changes may be necessary in the NAMELIST format for different systems. The format shown is for CDC systems. For IBM systems, the dollar signs in the NAMELISTS should be changed to ampersands. The options and input variables illustrated in these test cases are all described in reference 1, sections 3.3 and 4.3.

## D.1 One-Dimensional Test Case

The first test case uses the basic program. The system analyzed for this case is one-dimensional

$$\dot{x}(t) = Ax(t) + Bu(t) + Fn(t) \qquad x(0) = 0$$

$$z(t_i) = x(t_i) + G\eta(t_i)$$

The true values of A, B, F, and G are -1, 10, 2, and 1, respectively. The input is a square wave with a period of 2 seconds; its value is 0 for the first second and 1 for the second second, repeating thereafter. Ten seconds of data at 100 samples per second are used. A pseudorandom number generator is used to create the state and measurement noise signals. Subroutine READTH is modified to compute the data for this test case, instead of reading a data file. This test case can be used to experiment with ways to easily implement modifications on particular computer systems. The case can also be run without modifying READTH by computing the data with a separate program. The modification instruction for READTH are shown in supplement 2. These modifications also add subroutine GAUSSN to generate the pseudorandom noise. The resulting modified READTH and GAUSSN are shown in supplement 2 after the modification instructions. Note that no measurement noise is included in the first time point. Next in supplement 2 are the input cards and output listings for the case.

The following options and features are used in this test case. The data channel numbers are specified in the NAMELIST, as the defaults are not correct for this case. Note that CARD, TAPE, and NREC (ref. 1, sec. 3.3.8(2) and (6)) can be ignored since they are not used by the modified READTH (and are never used outside of READTH). If (instead of being computed in READTH) the data for the test case are computed by a separate program and stored on a file, CARD, TAPE, and NREC will be relevant. ITG is set in order to turn on $G$ determination, and the maximum total number of iterations is set to 10. Note that the program converges with $G$ fixed before the seventh iteration, triggering the start of the $G$ determination. Final convergence is then achieved and the program stops well before reaching the 10th iteration.

The TEST option is turned on for this case in order to print out dimensional matrices and gradients. The PRINTI option is turned on in the example so that the generated time history can be checked. PRINTO is turned on to provide a check on the implementation of the time history estimation in GIRL. PRINTI and PRINTO result in large amounts of output (particularly PRINTI); thus, this case should probably be run first without them. The case can be rerun with them turned on if necessary for debugging. The RELAB option is used to read in more meaningful labels than the defaults. The time history plot option is on by default. A plot is requested of the corrected state estimate. NEXPLT is used to request plots of the true state, state noise, and measurement noise, which are carried as the first three extra signals for this case. Setting NUPLT to 1 does not affect the resulting plot for this case, but saves the computer time otherwise required to notice that the second to fourth controls are identically zero and thus need not be plotted.

Starting values are read in for the AN, CN, F, and GGI matrices. Both full and diagonal input formats are used (there is little difference for 1 X 1 matrices). A starting BN is not read in, so it is 0, along with all of the other unmentioned matrices except for RN. The AV, BV, and FV matrices are read in to specify the unknowns. GGI is treated separately and is specified to be unknown by ITG in the NAMELIST. The values of MX and MZ are set to 1 from the sizes read in for the AN and GGI matrices. The BN matrix was not read in, but MU is set to 1 because of the unknown derivative of the first control specified by BV. The value of MB is set to NCASE which is 1; even though $S$ and $H$ are both 0 for this case, MB must be defined, and 0 is not an allowable value.

The final estimates, Cramér-Rao bounds, and true values are found in the following table.

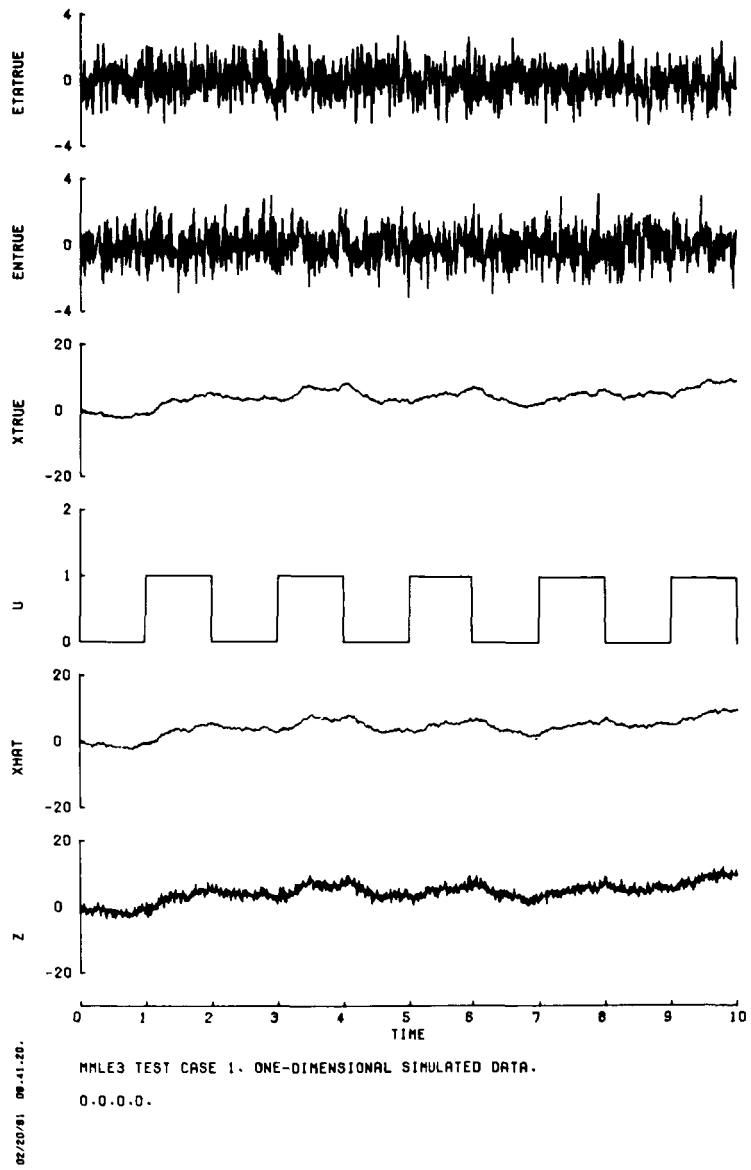
| | Estimate | Cramér-Rao bound | True value |
|---|---|---|---|
| AN | -0.8897 | 0.20 | -1.0 |
| BN | 9.619 | 1.5 | 10.0 |
| FN | 2.071 | 0.21 | 2.0 |

The measurement noise covariance matrix, $\mathcal{G}$, is not directly estimated, but can be computed from the residual power estimate, $GGI^{-1}$, and the estimate of the prediction error power, $P$ (the Riccati covariance matrix).

$$\begin{aligned}\mathcal{G} &= \sqrt{G^2 - P} \\ &= \sqrt{GGI^{-1} - P} \\ &= \sqrt{.8176^{-1} - .2184} \\ &= 1.0023\end{aligned}$$

This compares to the true value of 1. A Cramér-Rao bound is not computed for $\mathcal{G}$.

The time history plots from this case are shown below.



## D.2 Longitudinal Test Cases

The second and third test cases are longitudinal cases using the standard aircraft routines. Both use actual flight data from a T-37 aircraft (ref. 10). These two cases are set up as a single job, but can be run separately.

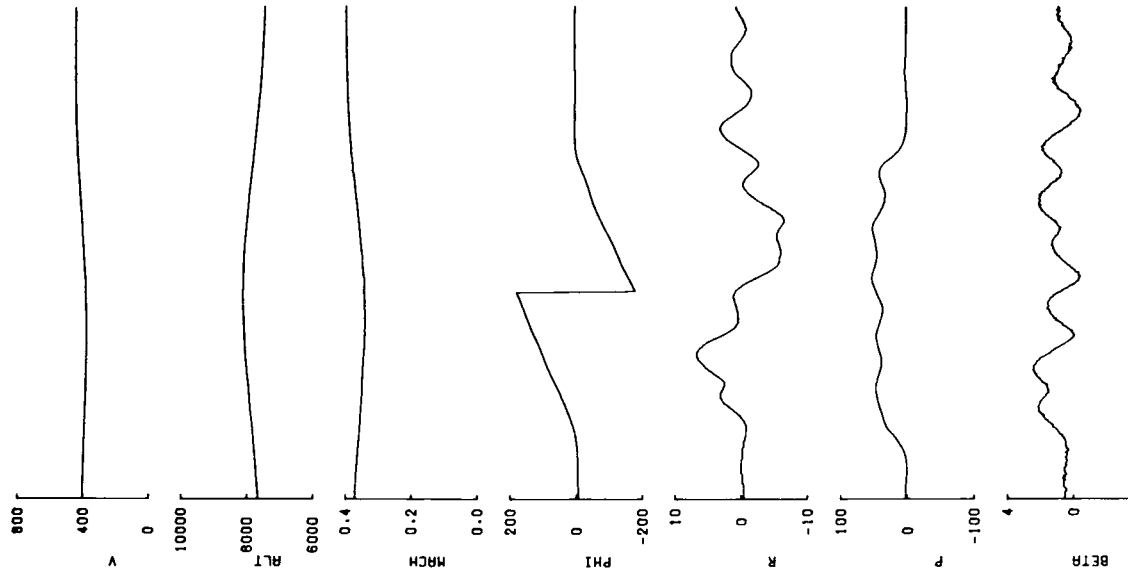
A common type of update to the standard aircraft routines is illustrated in these check cases: a modification to automatically compute weights and inertias. Sometimes fuel weights or other quantities recorded as extra signals are used for such computations. In the test case here, a table of fuel weights and times obtained from pilot lap notes is read in subroutine ONCE. Also read in are tables to obtain total weight and inertias as a function of fuel weight. Subroutine AVERAG then uses linear interpolation on these tables to automatically obtain the total weight and inertias for each maneuver. The modification instructions for this update, followed by the resulting modified program listings, are shown in supplement 2. The test cases can be run without modifying the program by entering the weights and inertias from the output listing into NAMELIST USER.

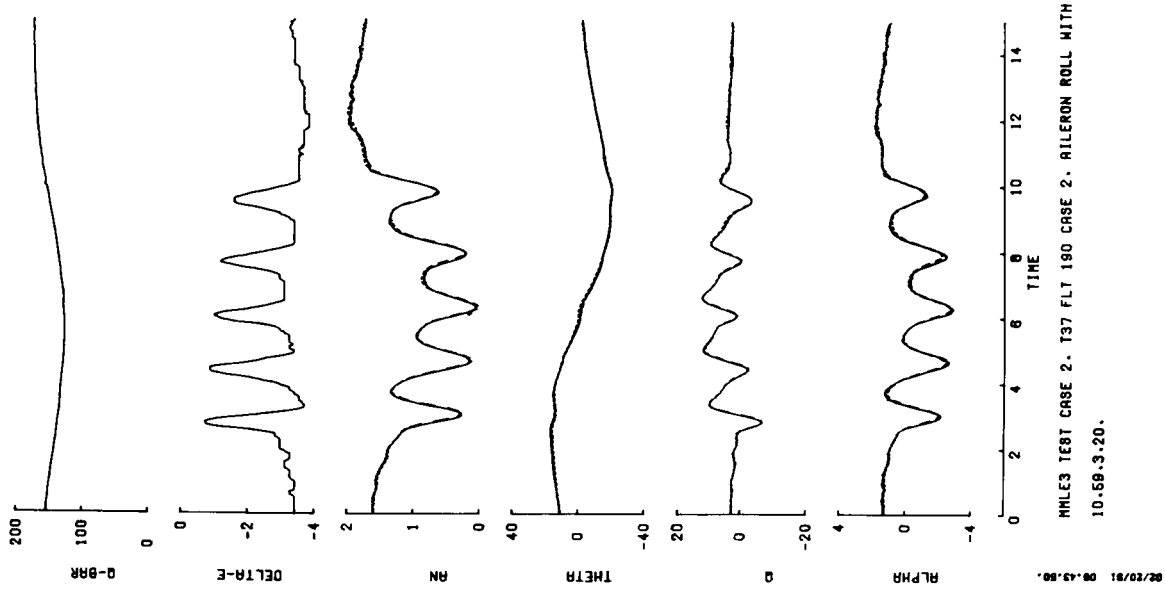
Next on supplement 2 are the input cards and output listings for the test cases. A simple predicted-derivative data set is used which has a constant value for each derivative. One lateral-directional derivative, $C_{\ell_\beta}$, is included to illustrate that it is ignored for these longitudinal cases.

Test case 2 is an unusual maneuver, designed for estimating $C_{m_{\dot{\alpha}}}$ (ref. 11). This maneuver requires several of the program's more sophisticated features. Because of the significant variations of $\bar{q}$, $V$, $\theta$, and $\varphi$ during the maneuver, the time-varying option must be used. Since the aircraft does a complete 360° roll during this longitudinal maneuver, the lateral-directional cross-coupling terms are quite important, particularly in the $\dot{\alpha}$ and $\dot{\theta}$ equations. MZ is set to 4 to eliminate the $a_x$ observation equation, using the simplified low $\alpha$ longitudinal equations (compare the lists of unknowns in the output of the second and third test cases). In order to reasonably match $a_x$ for this maneuver, thrust and drag must be treated separately because of the $\bar{q}$ variation. (Such treatment can be made, but is not included in the test case.) The default automatic scaling for altitude (extra signal 7) is overridden to obtain a more sensitive scale, since the default would include 0 in the scale. FREQCR is set in order to obtain filtered residual powers and use them to adjust the Cramér-Rao bounds. An RV matrix is read in to define $C_{m_{\dot{\alpha}}}$ ($RN_{2,1}$) to be unknown. The time history follows on cards. The record length of the input time history is set to 23 in order to use one less card per time point than with the default, because the last two channels are not used for this case.

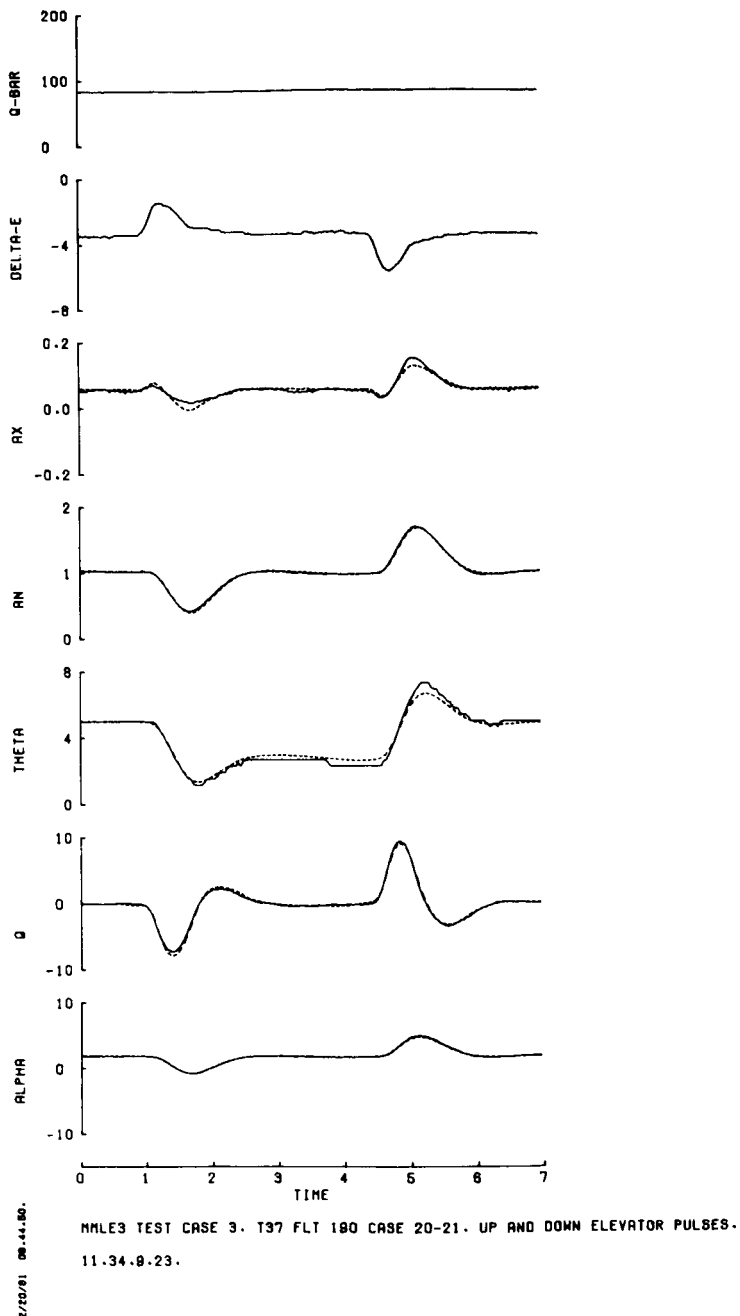
Convergence is rapid and monotone. The first iteration changes the linear unknowns only and lowers the cost functional by $2\frac{1}{2}$ orders of magnitude. The second iteration changes all of the unknowns and reduces the cost functional by a factor of 4. The solution has then been essentially reached; the remaining iterations just add more significant digits. Note that the filtered error sum and log determinant are slightly better (in the fourth and fifth places) after iteration 3 than at the final value. This

is not unusual, because these are not the quantities minimized (the unfiltered weighted error sum is minimized). Needless to say, these quantities should be expected to be at least near their minima as they are on this case. The fit is shown on the following pages.


V
800
400
0
ALT
10000
8000
6000
MACH
0.4
0.2
0.0
PHI
200
0
-200
R
10
0
-10
P
100
0
-100
BETA
4
0
-4


Q-BAR
200
100
0
DELTA-E
0
-2
-4
AN
2
1
0
THETA
40
0
-40
Q
20
0
-20
ALPHA
4
0
-4
0
2
4
6
8
10
12
14
TIME
MMLE3 TEST CASE 2. T37 FLT 190 CASE 2. AILERON ROLL WITH ELEVATOR PULSES.
10.50.3.20.
02/03/91 08.43.00.

Test case 3 is a set of elevator pulses typical of the data normally obtained for longitudinal stability and control derivative estimation. The time-varying option is not needed for this case. The more complicated longitudinal equations including $a_x$ are used in the sample run (although the case will run quite well and use less computer time if MZ is set to 4 to eliminate $a_x$). The convergence is excellent and similar to the previous case. Several observations can be made about the fit shown below.



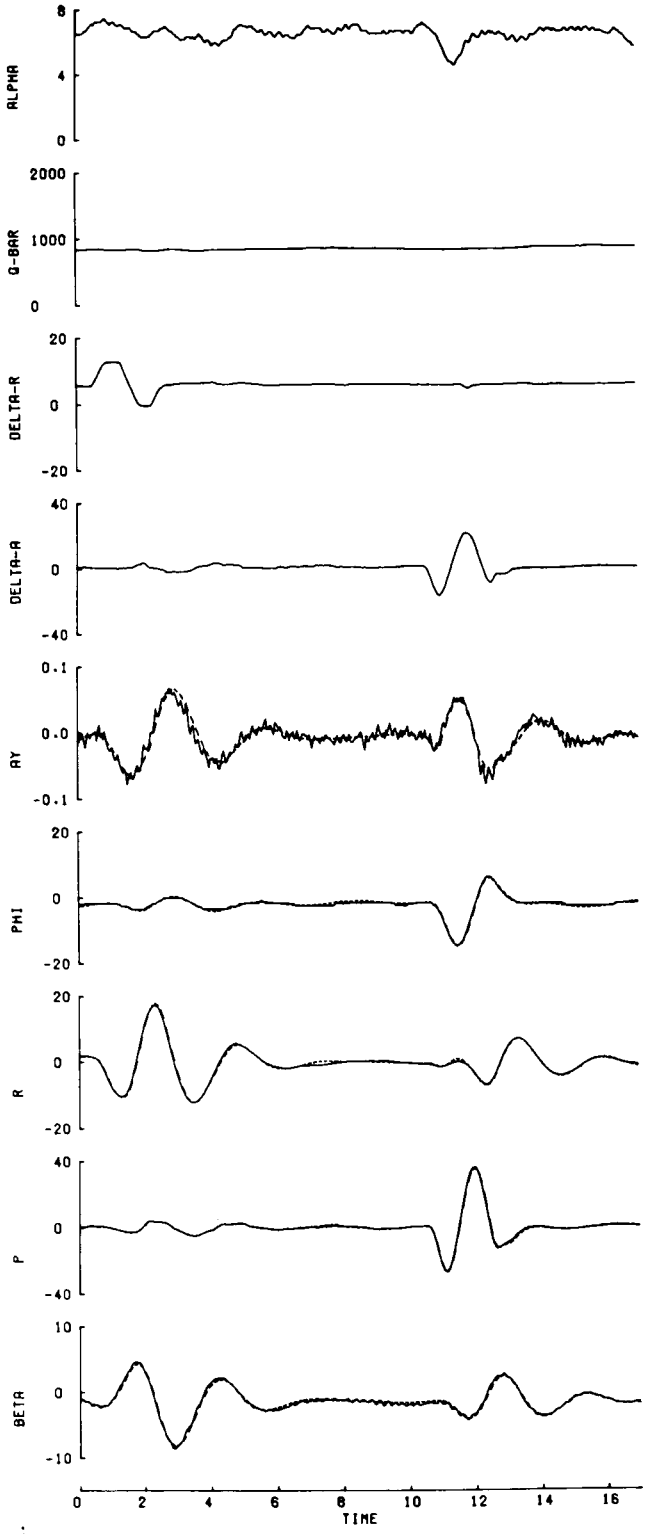
The resolution on $\theta$ is relatively poor for this maneuver, but does not appear to have caused any problems (the rounded corners on the bit jumps are due to digital filtering done after the flight). The $a_x$ match shows some significant discrepancies. The discrepancies are strongly correlated with $\alpha$, and not with $q$ or $\delta_e$. This suggests a significant nonlinearity in the $C_X$ versus $\alpha$ curve. Since $\alpha$ ranges from -1° to 5°, it would not be surprising to find the linear derivative $C_{X_\alpha}$ inadequate. A $C_{X_{\alpha^2}}$ term, either replacing or in addition to the $C_{X_\alpha}$ term, might give a better model. Nonlinear terms such as $C_{X_{\alpha^2}}$ can be implemented in MMLE3 by forming $\alpha^2$ as an extra control. This task is appropriate for subroutine THMOD or can be done by a separate program to add the signal to the data file (of course, the $\alpha$ used for this purpose should be corrected to the center of gravity and for any upwash). Running this case with a $C_{X_{\alpha^2}}$ term is left as a relatively simple exercise in using MMLE3.

## D.3 Lateral-Directional Test Case

The fourth test case is a lateral-directional case using the standard aircraft routines. This case consists of actual flight data from an oblique wing aircraft (ref. 12); the wing is not skewed during this maneuver. The data are typical of those obtained for lateral-directional stability and control derivative estimation.

There are no program modifications for this test case. The input cards and output listing are in supplement 2 following the previous test cases. This case is run without a predicted-derivative data set to illustrate that option. Vehicle geometry and instrument positions are read in NAMELIST USER. The weight and inertias are also read in the NAMELIST for this case, in contrast to the previous two test cases, which contain an update to compute them. This case is run using metric units. The engine revolutions per minute (extra signal 11) is specified to be found on channel 10 of the input data instead of the default channel 26. Since no predicted-derivative data set was used, a starting AN matrix must be read in to provide reasonable starting estimates. Note that the AN read in is not square. The program accepts this input, but later changes the dimensions used to be consistent. No BN matrix is read in, so the starting estimates of the control derivatives are all 0. BV is read in order to override the default that includes $C_{Y_{\delta_a}}$ as an unknown. The BV matrix read in also includes values in the fifth row to illustrate that they are ignored. The number of rows of BN (and thus BV) is forced internally to equal MX (4 for this case) and any entries outside of this range are ignored.

Convergence is rapid and uneventful. The resulting fit is shown below.




02/20/91 08.41.11.

PMLE3 TEST CASE 4. SKEW WING FLIGHT 1 CASE 15-16. RUDDER-AILERON DOUBLETS.

6.32.31.6.

## REFERENCES


1. Maine, Richard E.; and Iliff, Kenneth W.: User's Manual for MMLE3, A General FORTRAN Program for Maximum Likelihood Parameter Estimation. NASA TP-1563, 1980.

2. UPDATE Version 1 Reference Manual. Pubn. No. 60449900, Control Data Corp., 1980.

3. FORTRAN Extended Version 4 Reference Manual. Pubn. No. 60497800, Control Data Corp., 1978.

4. Programming CalComp Pen Plotters. California Computer Products, Inc., Sept. 1969.

5. American Standard FORTRAN. ASA X3.9-1966, American Stand. Assn., Inc., 1966.

6. Smith, B. T.; Boyle, J. M.; Dongarra, J. J.; Garbow, B. S.; Ikebe, Y.; Klema, V. C.; and Moler, C. B.: Matrix Eigensystem Routines—EISPACK Guide. Lecture Notes in Computer Science, 6. Second ed. Springer-Verlag (Berlin), 1976.

7. Moler, Cleve; and Van Loan, Charles: Nineteen Dubious Ways to Compute the Exponential of a Matrix. SIAM Review, vol. 20, no. 4, Oct. 1978, pp. 801-836.

8. Wilkinson, J. H.: The Algebraic Eigenvalue Problem. Clarendon Press (London), 1978.

9. Vaughan, David R.: A Nonrecursive Algebraic Solution for the Discrete Riccati Equation. IEEE Trans. Automat. Contr., vol. AC-15, no. 5, 1970, pp. 597-599.

10. Shafer, Mary F.: Stability and Control Derivatives of the T-37B Airplane. NASA TM X-56036, 1975.

11. Maine, Richard E.; and Iliff, Kenneth W.: Maximum Likelihood Estimation of Translational Acceleration Derivatives From Flight Data. AIAA Paper 78-1342, Aug. 1978.

12. Maine, Richard E.: Aerodynamic Derivatives for an Oblique Wing Aircraft Estimated From Flight Data by Using a Maximum Likelihood Technique. NASA TP-1336, 1978.

| 1. Report No. NASA TP-1690 | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| 4. Title and Subtitle<br>PROGRAMMER'S MANUAL FOR MMLE3, A GENERAL FORTRAN PROGRAM FOR MAXIMUM LIKELIHOOD PARAMETER ESTIMATION | | 5. Report Date<br>June 1981<br>6. Performing Organization Code<br>505-36-24 |
| 7. Author(s)<br>Richard E. Maine | | 8. Performing Organization Report No.<br>H-1105 |
| 9. Performing Organization Name and Address<br>NASA Dryden Flight Research Center<br>P.O. Box 273<br>Edwards, California 93523 | | 10. Work Unit No.<br>11. Contract or Grant No. |
| 12. Sponsoring Agency Name and Address<br>National Aeronautics and Space Administration<br>Washington, D.C. 20546 | | 13. Type of Report and Period Covered<br>Technical Paper<br>14. Sponsoring Agency Code |

15. Supplementary Notes

Microfiche supplements are provided in back cover.
Information for requesting the MMLE3 program is included.

16. Abstract

This report is a programmer's manual for the FORTRAN IV computer program MMLE3. MMLE3 is a maximum likelihood parameter estimation program capable of handling general bilinear dynamic equations of arbitrary order with measurement noise and/or state noise (process noise). The basic MMLE3 program is quite general and, therefore, applicable to a wide variety of problems. The basic program can interact with a set of user-written problem-specific routines to simplify the use of the program on specific systems. A set of user routines for the aircraft stability and control derivative estimation problem is provided with the program. The implementation of the program on specific computer systems is discussed. The structure of the program is diagrammed, and the function and operation of individual routines is described. Complete listings and reference maps of the routines are included on microfiche as a supplement. Four test cases are discussed; listings of the input cards and program output for the test cases are included on microfiche as a supplement. The theory and use of the program are described in the User's Manual for MMLE3, A General FORTRAN Program for Maximum Likelihood Parameter Estimation by Richard E. Maine and Kenneth W. Iliff (NASA TP-1563).

| 17. Key Words (Suggested by Author(s)) | 18. Distribution Statement |
|---|---|
| Maximum likelihood<br>Parameter estimation<br>Aircraft stability and control<br>Computer programs<br>System identification<br>Aircraft flight testing | Unclassified-Unlimited<br>STAR category 59 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price* |
|---|---|---|---|
| Unclassified | Unclassified | 118 | A06 |

*For sale by the National Technical Information Service, Springfield, Virginia 22161*

NASA-Langley, 1981