Special Report 863
July 1990

Uptake, Transport, and Accumulation of Organic Chemicals by Plants (UTAB 4.6)

Agricultural Experiment Station
Oregon State University

UPTAKE, TRANSPORT, AND ACCUMULATION OF CHEMICALS BY PLANTS (UTAB 4.6)

PROGRAM LISTING

D. E. CAWLFIELD, F. T. LINDSTROM, L. BOERSMA

AUTHORS: D. E. Cawlfield is senior systems analyst, F. T. Lindstrom is associate professor and L. Boersma is professor, Department of Soil Science, Oregon State University

FOREWORD

The uptake, transport, and accumulation of organic and inorganic chemicals by plants is influenced by characteristics of the plant, properties of the chemical, concentration of solute to which the roots are exposed, and by prevailing environmental conditions. A mathematical model was developed to simulate this uptake. The model takes into account the complex interrelationships that exist between the physical, chemical, and physiological processes occurring in specific plant tissues and the response of these processes to environmental conditions.

The model is based on definition of the plant as a set of compartments separated from each other by membranes and/or diffusion-transport paths. Movement of water and solutes between compartments occurs by mass flow and diffusion. The compartments represent major pools for accumulation of water and solutes. Anatomical features of the compartments and the manner in which they are connected are described by a series of equations based on conservation of mass. The model has been successfully used to simulate uptake and distribution patterns for several chemicals in soybean plants. The model offers promise for future use, but additional testing and validation are needed.

This report provides the source program listing for the UTAB computer program, version 4.6. UTAB is an implementation of a mathematical model that describes the uptake, transport, and accumulation of organic and inorganic chemicals by plants.

The model is written in ANSI standard Fortran 77 and runs without difficulty on most microcomputers of the IBM PC/XT/AT class, using the DOS operating system. The use of an 8087 math co-processor is desirable, but not required.

ACKNOWLEDGMENT

This publication reports results of studies supported by Cooperative Agreement GR811940-01-0 "A Mathematical Model of the Bioaccumulation of Xenobiotic Organic Chemicals in Plants" between the Corvallis Environmental Research Laboratory of the U.S. Environmental Protection Agency and the Department of Soil Science at Oregon State University. Additional support was from the Oregon Agricultural Experiment Station.

PROGRAM LISTING

```
*PLANT46      FILE: PLANT46.FOR        Last revision:  November 14, 1989
C
C
C*-------------------------------------------------------------------*
C*                                                                   *
C*    P L A N T    T R A N S P O R T    T H R E E    L E A F         *
C*                          M O D E L                                *
C*                       (VERSION 4.6)                               *
C*                                                                   *
C*    PROGRAM MANAGER : DR. CRAIG MCFARLANE                          *
C*                                                                   *
C*    PROGRAMMER : DAVID E. CAWLFIELD            1/01/1988           *
C*                 GILBERT A. BACHELOR          11/03/1988           *
C*                 OREGON STATE UNIVERSITY                           *
C*                 SOIL SCIENCE DEPARTMENT                           *
C*                 CORVALLIS, OREGON 97331                           *
C*                                                                   *
C*    PROGRAM DEVELOPED FOR EPA GRANT NO. CR811940-01-03             *
C*                                                                   *
C*                    ALL RIGHTS RESERVED                            *
C*                                                                   *
C*                                                                   *
C*    THIS IS THE MAIN PROGRAM FOR UTAB VER 4.6, AND SERVES AS THE   *
C*    "DRIVER" ROUTINE FOR THE SYSTEM.  ITS PURPOSE IS TO INITIALIZE *
C*    THE VARIOUS INPUT AND OUTPUT FILES, AND THEN PASS CONTROL TO   *
C*    ROUTINES WHICH READ THE INPUT DATA, PREPARE THE SYSTEM MASS    *
C*    BALANCE EQUATIONS, SIMULATE THE SOLUTE TRANSPORT (THAT IS,     *
C*    INTEGRATE THE SYSTEM OVER THE REQUESTED TIME PERIOD), AND      *
C*    FINALLY  GENERATE SELECTED OUTPUT REPORTS.                     *
C*                                                                   *
C*    META COMMANDS ARE PROVIDED TO INCLUDE FILES NAMED "COMSIZE.I46" *
C*    AND "COMBLOC.I46".  THESE INCLUDED FILES CONTAIN THE DEFINITIONS *
C*    OF ALL OF THE GLOBAL ARRAYS AND VARIABLES WHICH ARE IN "COMMON". *
C*    IF THE PROGRAM TERMINATES ABNORMALLY, AS IT MIGHT WHEN GIVEN BAD *
C*    INPUT DATA, "STOP" STATEMENTS, WHICH RETURN NUMERIC EXIT CODES,  *
C*    ARE EXECUTED.  THESE NUMERIC EXIT CODES ARE USEFUL WHEN THE     *
C*    PROGRAM IS INITIATED IN A BATCH ENVIRONMENT, SINCE THEY MAY BE  *
C*    USED FOR PROCESSING CONTROL.                                   *
C*                                                                   *
C*    DEFINITIONS OF VARIABLES                                       *
```

```
C*    ------------------------                                          *
C*       REPLACE       :  A CHARACTER VARIABLE WHICH IS USED TO TEST    *
C*                        "YES" OR "NO" RESPONSES FROM THE USER.        *
C*       EXIST         :  A LOGICAL VARIABLE USED TO TEST WHETHER OR    *
C*                        NOT AN OUTPUT FILENAME PROVIDED BY THE USER   *
C*                        ALREADY EXISTS.                               *
C*       MORE          :  A LOGICAL VARIABLE WHICH IS SET TO "TRUE"     *
C*                        IF THERE IS AN ADDITIONAL "TIME EVENT" TO BE  *
C*                        RUN.  IT IS OTHERWISE SET TO "FALSE" AND THE  *
C*                        PROGRAM TERMINATES.                           *
C*       FIRST         :  A LOGICAL VARIABLE WHICH IS SET TO "TRUE"     *
C*                        THE FIRST TIME THAT THE DATA INPUT ROUTINE    *
C*                        IS INVOKED; AND "FALSE" OTHERWISE.            *
C*    MAJOR FUNCTION CALLS                                              *
C*    --------------------                                              *
C*       CALL CLS      : CLEAR CURRENT SCREEN.  THE CODE IS             *
C*                        IN FILE "UTIL1.FOR"                           *
C*       CALL SCRNHD   : PRINT SCREEN HEADER.   THE CODE IS             *
C*                        IN FILE "BLOCK46.FOR"                         *
C*       CALL MSSGE1   : PRINT MESSAGE TO SCREEN.  THE CODE IS          *
C*                        IN FILE "PLANT46.FOR"                         *
C*       CALL SETPR()  : SET PRINTER TO COMPRESSED MODE WHEN ('ON ');   *
C*                        SET PRINTER TO NORMAL MODE WHEN ('OFF').      *
C*                        THE CODE IS IN FILE "UTIL1.FOR"               *
C*       CALL DTADRVR  : TRANSFER CONTROL TO DATA DRIVER SUBROUTINE.    *
C*                        THE CODE IS IN "DTADR46.FOR"                  *
C*       CALL  MTXA    : COMPUTE MATRIX A(N,N). THE CODE IS             *
C*                        IN FILE "MTRXA46.FOR"                         *
C*       CALL INTGRL   : TRANSFER CONTROL TO SIMULATOR SUBROUTINE.      *
C*                        THE CODE IS IN "INRES46.FOR"                  *
C*       CALL BYE      : PRINT MESSAGE TO SCREEN. THE CODE IS           *
C*                        IN FILE "BLOCK46.FOR"                         *
C*-------------------------------------------------------------------*
C
      PROGRAM PLANT46
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER   LEN, LENSTR
      CHARACTER DAYTEM*8
      CHARACTER DRIVE*2, CACHF*20
      CHARACTER*20   FI, FO, FM, FL, FG, FD, FC, FROOT
      LOGICAL   EXIST, FIRST, MORE, REPLACE
      EXTERNAL  LENSTR
*************************************************************************
*  The code for the main program begins here.                           *
*************************************************************************
C
C-----INPUT FORMAT
```

```
C
 1000 FORMAT(A)
C
C-----OUTPUT FORMATS
C
 2000 FORMAT(//,T10,'Enter date [CR for ',A,'] >- ',$)
 2005 FORMAT(T10,'*Error*  -- cannot find file ', A)
 2010 FORMAT(T10,'Enter PLANT input filename >- ',$)
 2020 FORMAT(T10,'Enter filename for output -or-',/,
     &       T10,'Use LPT1 or LPT2 for printer >- ',$)
 2023 FORMAT(T10,'Enter data-cache drive [CR for C:] >- ',$)
 2030 FORMAT(3(/),T12,' ** WARNING **',/,
     &         T12,' The filename you choose is being used.')
 2033 FORMAT(  T12,' [Cache Filename = ', A,' ]')
 2035 FORMAT(  T12,' Do you want to replace (write over)',
     &             ' this file (T/F) [', L1, '] >- ',$)
 2037 FORMAT(T10,'If you want wide reports (compressed mode),',/,
     & T10, 'Enter T; enter F for narrow form. (T/F) [',
     &   L1, '] >- ',$)
 2038 FORMAT(T10,'Are you using a Laser-Jet Printer (T/F) [',
     &  L1, '] >- ', $)
 2040 FORMAT(T10,'Enter T to go another time-slice; F to QUIT [',
     &  L1, '] >- ', $)
C
C*--------------------------------------------------------------------*
C*            B E G I N   M A I N   P R O G R A M                     *
C*--------------------------------------------------------------------*
C
C-----CLEAR SCREEN, PRINT SCREEN HEADER, GET DATE...
C
      CALL CLS
      CALL SCRNHD
      CALL DAYSTR(DATE)
      CALL TIMSTR(TIME)
      WRITE(*,2000) DATE
      READ (*,1000) DAYTEM
      IF (DAYTEM .NE. '          ')  DATE = DAYTEM
C
C  Get the Input Filename.  Quit if missing (best for batch).
C
      WRITE(*,2010)
      READ (*,1000)  INFILE
      CALL STRUP(INFILE)
      INQUIRE(FILE=INFILE ,EXIST=EXIST)
      IF (.NOT. EXIST) THEN
           WRITE(*,2005) INFILE
           STOP 2005
      ENDIF
```

```
C
C  Get the Output Filename.  *May* be printer.
C
   10 WRITE(*,2020)
      READ (*,1000) OUTFILE
      CALL STRUP(OUTFILE)
      IF (OUTFILE(1:4) .EQ. 'LPT1' .OR.
     &    OUTFILE(1:4) .EQ. 'LPT2') THEN
C
C  Output Direct to Printer Device
C
           CALL CLS
           CALL SCRNHD
           CALL MSSGE1
           CALL CLS
           CALL SCRNHD
C..If output to printer, get file root from input file...
           FROOT = INFILE
           LEN = LENSTR(FROOT)
           LEN = MAX(LEN,5)
      ELSE
C
C-------Check for existing output filename.  If filename already
C       exists then allow user to change to another filename or
C       write over the old filename.
C
           INQUIRE(FILE=OUTFILE, EXIST=EXIST)
           REPLACE = .TRUE.
           IF (EXIST) THEN
                CALL CLS
                CALL SCRNHD
                WRITE(*,2030)
   15           WRITE(*,2035) REPLACE
                CALL GET1LO(REPLACE, REPLACE, *15)
                IF (.NOT. REPLACE) GOTO 10
                CALL CLS
                CALL SCRNHD
           ENDIF
           FROOT = OUTFILE
           LEN = LENSTR(FROOT)
           LEN = MAX(LEN,5)
      ENDIF
C
C  Get Drive letter for data cache...
C
   20 WRITE(*,2023)
      READ (*,1000) DRIVE
      IF (DRIVE .EQ. '  ') DRIVE = 'C:'
```

```
      IF ( INDEX(DRIVE, ':') .EQ. 0 ) THEN
           DRIVE(2:2) = ':'
      ENDIF
      CACHF = DRIVE // FROOT(1:LEN-4) // '.SAV'
      INQUIRE(FILE=CACHF, EXIST=EXIST)
      REPLACE = .TRUE.
      IF (EXIST) THEN
           CALL CLS
           CALL SCRNHD
           WRITE(*,2030)
           WRITE(*,2033) CACHF
   25      WRITE(*,2035) REPLACE
           CALL GET1LO(REPLACE, REPLACE, *25)
           IF (.NOT. REPLACE) GO TO 20
           CALL CLS
           CALL SCRNHD
      ENDIF
C
C-----OPEN FILES
C
      FI = INFILE
      FO = OUTFILE
      LEN = MAX(LEN,5)
      FM = FROOT(1:LEN-4) // '.FMA'
      FL = FROOT(1:LEN-4) // '.LMP'
      FG = FROOT(1:LEN-4) // '.GRP'
      FD = FROOT(1:LEN-4) // '.DAO'
      FC = FROOT(1:LEN-4) // '.CHG'
      OPEN(LUDAI, FILE=FI, STATUS='OLD', ERR=990)
      OPEN(LUPRN, FILE=FO, STATUS='UNKNOWN', ERR=990)
      OPEN(LUDAO, FILE=FD, STATUS='UNKNOWN', ERR=990)
      OPEN(LUFMO, FILE=FM, STATUS='UNKNOWN', ERR=990)
      OPEN(LULMO, FILE=FL, STATUS='UNKNOWN', ERR=990)
      OPEN(LUGMO, FILE=FG, STATUS='UNKNOWN', ERR=990)
      OPEN(LUXPI, FILE=FC, STATUS='UNKNOWN', ERR=990)
C  Data Cache ...
      OPEN(LUDAC, FILE=CACHF, ACCESS='SEQUENTIAL', FORM='UNFORMATTED',
     &  STATUS='UNKNOWN', BLOCKSIZE=6000)
C
C-----ALLOW USER TO SELECT WIDE OR NARROW REPORT
C
      WIDSET = .FALSE.
      WIDE = .FALSE.
   52 WRITE(*,2037) WIDE
      CALL GET1LO(WIDE, WIDE, *52)
      LASJET = .FALSE.
   54 WRITE(*,2038) LASJET
      CALL GET1LO(LASJET, LASJET, *54)
```

```
C
C-----Perform Simulation (Integrate System).
C     WHILE (MORE) DO ...
C
      FIRST = .TRUE.
      MORE  = .FALSE.
      MDATA = .FALSE.
  100 CONTINUE
          CALL DTADRVR( FIRST )
          CALL MIXA
          CALL CLS
          CALL SCRNHD
          CALL MSSGE3
C-----SIMULATE SYSTEM OPERATION OVER TIME.
          CALL INTGRL( FIRST )
          FIRST = .FALSE.
          CALL CLS
          MORE = .FALSE.
   40     WRITE(*,2040) MORE
          CALL GET1LO(MORE, MORE, *40)
      IF (MORE) GO TO 100
C
C  Empty (flush) the data cache buffer into the intermediate file
C  before the report generator is invoked.
C
      CALL BUFOUT(IBUFF)
C-----GENERATE REPORT(S)
      CALL REPORT
C
C-----SET PRINTER BACK TO NORMAL MODE
C
      IF (WIDSET) THEN
          IF (LASJET) THEN
              CALL SETLJ('OFF', LUPRN)
          ELSE
              CALL SETPR('OFF', LUPRN)
          ENDIF
      ENDIF
      CALL CLS
      CALL SCRNHD
      CALL BYE
C
C-----CLOSE FILES
C
      CLOSE(LUDAI)
      CLOSE(LUDAO)
      CLOSE(LUPRN)
      CLOSE(LUFMO)
```

```
      CLOSE(LUIMO)
      CLOSE(LUGMO)
      CLOSE(LUXPI)
      CLOSE(LUDAC, STATUS='DELETE')
C
C*--------------------------------------------------------------------*
C*              E N D    M A I N    P R O G R A M                     *
C*--------------------------------------------------------------------*
C
      STOP 'Normal UTAB Termination'
  990 PRINT *,'An error occurred in attempting to open a data file.'
      STOP 990
      END
*MSSGE1
C
C*--------------------------------------------------------------------*
C*            S U B R O U T I N E     M S S G E 1                     *
C*    THIS SUBROUTINE PRINTS AN ADVICE MESSAGE TO CONSOLE TO REMIND   *
C*    USER TO TURN PRINTER POWER ON AND SET PRINTER SPOOLER LARGER.   *
C*    THIS SUBROUTINE IS CALLED FROM THE MAIN PROGRAM, WHICH IS       *
C*    CONTAINED IN THIS FILE.                                         *
C*--------------------------------------------------------------------*
C
      SUBROUTINE MSSGE1
      WRITE(*,2000)
 2000 FORMAT(3(/),
     & T10, 'Set printer power ON to receive output.',/,
     & T10, 'Recommendation:',/,T10,'**************',//,
     & T10, '1.  Set printer buffer ≥ 64K to speed up runtime, or,',/,
     & T10, '2.  Don''t use printer directly (print to file instead).',
     & 3(/))
      PAUSE
      RETURN
      END
*MSSGE3
C
C*--------------------------------------------------------------------*
C*            S U B R O U T I N E       M S S G E 3                   *
C*    THIS SUBROUTINE DISPLAYS A MESSAGE ON SCREEN WHEN THE MASS      *
C*    BALANCE EQUATIONS ARE CONSTRUCTED INTO A MATRIX FORM.  FROM     *
C*    THIS POINT USER CAN TERMINATE THE PROGRAM AS THEY WISH, OR      *
C*    PROCEED TO SIMULATE.   IT THEN DISPLAYS A MESSAGE SHOWING       *
C*    THE SIMULATED TIME.                                             *
C*--------------------------------------------------------------------*
      SUBROUTINE MSSGE3
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      LOGICAL   MORE
```

```
      WRITE(*,2000)
      MORE = .TRUE.
   11 WRITE(*,2010) MORE
      CALL GETLLO(MORE, MORE, *11)
      IF (.NOT. MORE) THEN
           IF (WIDSET) CALL SETPR('OFF', LUPRN)
           STOP '                Normal Termination.'
      ENDIF
C
      CALL CLS
      CALL SCRNHD
      WRITE(*,2020) INT(TSTOP)
C
      RETURN
 1000 FORMAT(L1)
 2000 FORMAT(5(/),
     &  T10,'The mass balance equations have been constructed.')
 2010 FORMAT(//,
     &  T10,'Do you want simulation to proceed (T/F) [', L1, '] >- ',$)
 2020 FORMAT(3(/),
     &        20X,'   ------ Simulation in Progress. ------',//,
     *      /,20X,'Targeted simulation time is ',I7,' hours.',
     *     //,20X,'If the program issues a warning message, ',/,
     *        20X,'do not panic, but let the execution proceed. ',/,
     *        20X,'Refer to the user guide for explanation of the ',/,
     *        20X,'warning message(s).',3(/),
     *        20X,'The simulated time is          hour(s).')
      END
```

```
*DTADRVR     FILE: DTADR46.FOR          Last revision: November 14, 1989
C
C
C*-------------------------------------------------------------------*
C*                                                                   *
C*   P L A N T    T R A N S P O R T    T H R E E    L E A F          *
C*                          M O D E L                                *
C*                       (VERSION 4.6)                               *
C*                                                                   *
C*    PROGRAM MANAGER : DR. CRAIG MCFARLANE                          *
C*                                                                   *
C*    PROGRAMMERS: DAVID E. CAWLFIELD          1/01/1988             *
C*                 GILBERT A. BACHELOR        11/03/1988             *
C*                 OREGON STATE UNIVERSITY                           *
C*                 SOIL SCIENCE DEPARTMENT                           *
C*                 CORVALLIS, OREGON 97331                           *
C*                                                                   *
C*    PROGRAM DEVELOPED FOR EPA GRANT NO. CR811940-01-03             *
C*                                                                   *
C*                    ALL RIGHTS RESERVED                            *
C*-------------------------------------------------------------------*
C
C
C*-------------------------------------------------------------------*
C*          S U B R O U T I N E      D T A D R V R                   *
C*    THE PURPOSE OF THIS SUBROUTINE IS TO PERFORM DATA INPUT AND    *
C*    DATA ORGANIZATION FOR THE PROGRAM SIMULATOR.                   *
C*    THIS SUBROUTINE IS CALLED FROM THE MAIN PROGRAM.               *
C*                                                                   *
C*    DEFINITION OF VARIABLES                                        *
C*    -----------------------                                        *
C*      TINTIAL   :  INITIAL TIME TO START SIMULATION                *
C*      TSTOP     :  TERMINATION TIME WHEN SIMULATION COMPLETED       *
C*      DT        :  TIME STEP INCREMENT                             *
C*      PRINTVL   :  PRINT RESULTS AT INTERVALS SPECIFIED BY PRINTVL *
C*      NCPMT     :  NUMBER OF COMPARIMENTS.  IT IS A CONSTANT       *
C*                   DECLARED IN FILE "COMSIZE.I46"                  *
C*                                                                   *
C*    FUNCTION CALLS                                                 *
C*    --------------                                                 *
C*      CALL CLS     : CLEAR CURRENT SCREEN.  THE CODE IS            *
C*                     IN FILE "UTIL1.FOR"                           *
C*      CALL SCRNHD  : PRINT SCREEN HEADER.  THE CODE IS             *
C*                     IN FILE "BLOCK46.FOR"                         *
C*      CALL MSSGE2  : PRINT MESSAGE TO SCREEN WHILE THE PROGRAM IS  *
C*                     PROCESSING.                                   *
C*      CALL PRHEAD  : PRINT HEADING FOR OUTPUT.  THE CODE IS IN FILE*
C*                     "PRINT46.FOR"                                 *
```

```
C*      CALL LOADDTA : LOAD INPUT DATA.  THE CODE IS                    *
C*                     IN FILE "DTADR46.FOR"                            *
C*      CALL INTVLE  : LOAD INITIAL MASS VALUES.  THE CODE IS           *
C*                     IN FILE "DTADR46.FOR"                            *
C*-------------------------------------------------------------------------*
C
      SUBROUTINE DTADRVR( FIRST )
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER   I, IM, IW, IBLOCK
      LOGICAL   FIRST, OK
      DOUBLE PRECISION    TEMP
C
C-----OUTPUT FORMATS
C
 1025 FORMAT(F12.0)
 2000 FORMAT(T10, A, ' (', F7.2, ') >- ',$)
 2015 FORMAT(T10,'Oops!  Start time < Stop time?  Try again.')
cx 2020 FORMAT(T10,'Enter time interval (step size) to integrate >- ',$)
 2025 FORMAT(T10,'Enter tolerance (default is ',1P,E10.4,') >-',$)
 2030 FORMAT(T10,'Enter time interval for output listing:',/,
     &       T10,'   {E.G.  1.0 for listing every 1  hour.',/,
     &       T10,'          1.5 for listing every 1½ hours.',/,
     &       T10,'          3.0 for listing every 3  hours.} ',
     &                                          '(', F6.2, ') >- ',$)
 2033 FORMAT(T10,'{Q''s computed internally.}')
 2040 FORMAT(I2)
 2050 FORMAT(1P, 30(1X,E16.8))
 2060 FORMAT(//, (T5,A) )
C
C
C-----ENTER INITIAL SIMULATION TIME
C     REPEAT...UNTIL (Tinitial < Tstop)
C
C  Note:  TSTOP is re-set to TNOW at the conclusion of a complete
C  time-slice integration, and hence is in step with the current time.
C
      TINITIAL = 0.D0
   10 CONTINUE
         IF ( FIRST ) THEN
            WRITE(*,'(//)')
   20       WRITE(*,2000)
     &      'Enter time (in hours) to start simulation', TINITIAL
            CALL GET1DP(TEMP, TINITIAL, *20)
            TSTOP = 0.D0
         ELSE
            TEMP = TSTOP
         ENDIF
```

```
            TINTIAL = TEMP
C
C-----ENTER TERMINATION TIME
C
            TEMP = TSTOP + 24.D0
   30       WRITE(*,2000)
     &      'Enter time (in hours) to stop  simulation', TEMP
            CALL GET1DP(TEMP, TEMP, *30)
       IF (TINTIAL .GE. TEMP) THEN
            WRITE(*,2015)
            GO TO 10
       ENDIF
       TSTOP = TEMP
C
C-----ENTER TIME INTERVAL FOR OUTPUT
C
   50 PRINTVL = 1.0D0
      WRITE(*,2030)          PRINTVL
      CALL GET1DP(PRINTVL, PRINTVL, *50)
      IF (PRINTVL .LE. 0.D0) GO TO 50
      DT = PRINTVL
C
C-----ALLOW USER TO SET THE TOLERANCE FOR THE INTEGRATION
C     (Not used in this version)
C
Cx    TOL = 1.D-4
C  60 WRITE(*,2025) TOL
C     READ (*,1025,ERR=60,END=70) TOL
C  70 IF ( TOL .LE. 0.D0 ) TOL = 1.D-4
      IF (WIDE) THEN
           IM =  5
           IW = 10
           IF (.NOT. WIDSET) THEN
                IF (LASJET) THEN
                     CALL SETLJ('ON ', LUPRN)
                ELSE
                     CALL SETPR('ON ', LUPRN)
                ENDIF
                WIDSET = .TRUE.
           ENDIF
      ELSE
           IM = 2
           IW = 6
           WIDSET = .FALSE.
      ENDIF
C
C-----LIST INPUT INFORMATION TO FILE 'xxxxx.DAO'
C
```

```
      WRITE(LUDAO,2040) NCPMT
      WRITE(LUDAO,2050) TINTIAL, TSTOP, DT, PRINTVL, TOL
C
C-----CLEAR SCREEN, PRINT SCREEN HEADER AND REPORT
C     PROCESSING MESSAGE TO USER.
C
      CALL CLS
      CALL SCRNHD
      CALL MSSGE2
C
C-----LOAD INPUT DATA, COMPUTE MATRIX A, AND LOAD INITIAL
C     VALUE.
C
      PRINTVL = PRINTVL - 0.01D0*DT
      IF      (FIRST) THEN
           CALL PRHEAD
           CALL LOADDTA(IM,IW)
           CALL INTVLE(IM,IW)
           PLEFT = 999
           IBLOCK = 0
      ELSE
           WRITE(LUDAO,2050) (XNOW(I), I = NCL, NCU)
      ENDIF
      CALL MODDAT(OK, FIRST)
      IF (.NOT. OK) THEN
           PRINT *,'Data Modification Rejected'
           STOP 4
      ENDIF
      IF (MDATA) THEN
           IBLOCK = IBLOCK + 1
           WRITE(LUPRN,'(//,T65,A,I3,/)') 'B L O C K  ', IBLOCK
           CALL LOADDTA(IM,IW)
      ENDIF
C
C-----THE FLOW RATE 'Q' WILL BE COMPUTED BY SUBROUTINE FINDQ...
C
      IF (FIRST .OR. MDATA) THEN
           WRITE(*,2033)
           CALL FINDQ
           CALL LSTVEC(LUPRN, Q, 1, NWU, IM, IW, 'Q-FLOW')
      ENDIF
      IF (FIRST) WRITE(LUPRN,2060) WORDS
      RETURN
      END
*LOADDTA
C*--------------------------------------------------------------------*
C*       S U B R O U T I N E      L O A D D T A                      *
C*      THIS SUBROUTINE IS TO LOAD THE APPROPRIATE INPUT DATA.       *
```

```
C*-------------------------------------------------------------------*
C
      SUBROUTINE LOADDTA(IM, IW)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      CHARACTER STRING*80
      INTEGER   LU, IM, IW
      LOGICAL   EOF
C
C-----LOAD DATA AND ECHO TO USER GIVEN OUTPUT FILENAME.
C-----INPUT IS FREE FORMAT
C
      LU = LUPRN
C
C  The data file now has a two-line title.  The third line of header
C  is ignored.  The title will be printed out *after* the rest of the
C  data (after initial values), where it is closer to the report.
C
      READ (LUDAI,1100,ERR=998,END=999) WORDS(1)
      READ (LUDAI,1100,ERR=998,END=999) WORDS(2)
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) AREA
      CALL LSTVEC(LU, AREA, NWL, NWU, IM, IW, 'AREA')
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) DELTAX
      CALL LSTVEC(LU, DELTAX, NWL, NWU, IM, IW, 'DELTA X')
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) SIGMA
      CALL LSTVEC(LU, SIGMA, NWL, NWU, IM, IW, 'SIGMA')
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) DIFU
      CALL LSTVEC(LU, DIFU, NWL, NWU, IM, IW, 'DIFFU')
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) CAIR
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) DCH
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) HC
      IF (WIDE) THEN
           WRITE(LUPRN,2030) 'CAIR', 'DCH', 'HC'
           WRITE(LUPRN,2040)  CAIR,   DCH,   HC
      ELSE
           WRITE(LUPRN,2033) 'CAIR', 'DCH', 'HC'
           WRITE(LUPRN,2043)  CAIR,   DCH,   HC
      ENDIF
2030 FORMAT(//,4X, 10(1X,A10,1X) )
2040 FORMAT(4X  1P, 10E12.3)
2033 FORMAT(//,4X, 10(1X,A10,1X) )
```

```
 2043 FORMAT(4X, 1P, 10E12.3)
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) VOL
      CALL LSTVEC(LU, VOL, NCL, NCU, IM, IW, 'VOLUME')
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) SB
      CALL LSTVEC(LU, SB, NCL, NCU, IM, IW, 'B')
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) LAMBDA
      CALL LSTVEC(LU, LAMBDA, NCL, NCU, IM, IW, 'LAMBDA')
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) QSTF
      CALL LSTVEC(LU, QSTF, NWL, NQSTO, IM, IW, 'QST-F')
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) QSTB
      CALL LSTVEC(LU, QSTB, NWL, NQSTO, IM, IW, 'QST-B')
C
C-----FRACTIONS OF THE FLOW RATES ARE READ FOR LEAF1 THROUGH
C     LEAF3 IN THAT ORDER.
C
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) FRAC
      CALL LSTVEC(LU, FRAC, 1, 3, IM, IW, 'FRAC')
C
C-----THE TRANSPIRATION RATES ARE READ FOR LEAF1 THROUGH LEAF3
C     IN THAT ORDER.
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999) TR
      CALL LSTVEC(LU, TR, 1, 3, IM, IW, 'TRANSP')
      RETURN
C
C  The following code is used to trap errors due to bad data,
C  early end-of-file, and so forth.
C
  998 IF ( EOF(LUDAI) ) THEN
            WRITE(*,2120) 'Early END-OF-FILE.'
      ENDIF
      BACKSPACE LUDAI
      READ (LUDAI,1100,END=999) STRING
      IF (INDEX(STRING,CHAR(26)) .NE. 0 ) THEN
            WRITE(*,2120) 'Early END-OF-FILE.'
            WRITE(*,2110)  STRING
            STOP 997
      ELSE
            WRITE(*,2120) 'Bad data?'
            WRITE(*,2110)  STRING
            STOP 998
      ENDIF
```

```
  999 STOP 999
 1100 FORMAT(A)
 2000 FORMAT(A,/,T8,A,/,T8,'This file is named --> ',A,
     *       T80,  'Date : ',A,2X,A,/)
 2003 FORMAT(A,/,T8,A,/,T8,'This file is named --> ',A,
     *       T52,  'Date : ',A,2X,A,/)
 2110 FORMAT(T10,'The last record read was:  ',/,1X,A)
 2120 FORMAT(/,T10,'*** W H O O P S ! ***     ',A)
      END
*INTVLE
C*--------------------------------------------------------------------*
C*            S U B R O U T I N E       I N T V L E                   *
C*       THIS SUBROUTINE IS TO INPUT INITIAL MASS VALUES             *
C*--------------------------------------------------------------------*
C
      SUBROUTINE INTVLE(IM, IW)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      CHARACTER STRING*80
      INTEGER   IM, IW
      LOGICAL   EOF
      READ (LUDAI,*,ERR=998,END=999)
      READ (LUDAI,*,ERR=998,END=999)    XNOW
      WRITE(LUDAO,2000) XNOW
      CALL LSTVEC(LUPRN, XNOW, NCL, NCU, IM, IW, 'INITIAL')
      RETURN
C
C  The following code is used to trap errors due to bad data,
C  early end-of-file, and so forth.
C
  998 IF ( EOF(LUDAI) ) THEN
          WRITE(*,2120) 'Early END-OF-FILE.'
      ENDIF
      BACKSPACE LUDAI
      READ (LUDAI,1100,END=999) STRING
      IF (INDEX(STRING,CHAR(26)) .NE. 0 ) THEN
          WRITE(*,2120) 'Early END-OF-FILE.'
          WRITE(*,2110)  STRING
          STOP 997
      ELSE
          WRITE(*,2120) 'Bad data?'
          WRITE(*,2110)  STRING
          STOP 998
      ENDIF
  999 STOP 999
 1100 FORMAT(A)
 2000 FORMAT(1P, 30(1X, E16.8))
 2110 FORMAT(T10,'The last record read was:  ',/,1X,A)
```

```
 2120 FORMAT(/,T10,'*** W H O O P S ! ***     ',A)
      END
*MSSGE2
C*-------------------------------------------------------------------*
C*           S U B R O U T I N E     M S S G E 2                     *
C*      THIS SUBROUTINE IS TO PROMPT SCREEN MESSAGE TO USER          *
C*-------------------------------------------------------------------*
C
      SUBROUTINE MSSGE2
      WRITE(*,2000)
 2000 FORMAT(5(/),
     *   T10,'The input data is now being printed to the user',/,
     *   T10,'designated output device.',//,
     *   T10,'PLEASE check the input data.',//,
     *   T10,'The program is now constructing the mass balance',/,
     *   T10,'equations.',///,
     *   T10,'Please wait a moment !')
      RETURN
      END
*MODDAT
C*-------------------------------------------------------------------*
C*           S U B R O U T I N E     M O D D A T                     *
C* A routine which is used to accept data modifications for the next *
C* time event ("time-slice").  This routine will allow the user to   *
C* either read another block of data from the input file, or to input *
C* selected data from the terminal.  The rationale for having two    *
C* alternatives is that entering data from the terminal is convenient*
C* for a small number of items, while the use of a data file is more *
C* appropriate for changing a large number of variables.  This routine*
C* calls upon UPDAT1, UPDAT2 and UPDAT3 which are listed after this  *
C* code.                                                             *
C*-------------------------------------------------------------------*
C
      SUBROUTINE MODDAT(OK, FIRST)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER   LEFT
      LOGICAL   OK, FIRST, CHANGE
      OK = .TRUE.
      IF (FIRST) LEFT = 1
   10 WRITE(*,2010) LEFT
      CALL GET1IN(LEFT, LEFT, *10)
      IF (LEFT .LE. 0 .OR. LEFT .GT. PLEFT) THEN
   20      WRITE(*,2020) LEFT, OK
           CALL GET1LO(OK, OK, *20)
           IF (OK) GO TO 10
      ENDIF
      PLEFT = LEFT
```

```
      IF (FIRST) PSTART = PLEFT
      MDATA = .FALSE.
      IF (FIRST) GO TO 999
 2025 FORMAT(T10,'Read another block from the data file?',
     &  ' (T/F) [', L1, '] >- ',$)
   25 WRITE(*,2025) MDATA
      CALL GET1LO(MDATA, MDATA, *25)
 2027 FORMAT(T10,'Do you want to make *any* further data changes? '
     &  '(T/F) [', L1, '] >- ',$)
   27 WRITE(*,2027) CHANGE
      CALL GET1LO(CHANGE, CHANGE, *27)
      IF (CHANGE) THEN
           CALL UPDAT1('SIGMAs', SIGMA, NWL, NWU)
           CALL UPDAT1('DIFFUs', DIFU, NWL, NWU)
C  UPDAT2 reads from a file ... but now see above
Cold       CALL UPDAT2('AREAs', AREA, NWL, NWU)
Cold       CALL UPDAT2('VOLs',  VOL,  NCL, NCU)
C
           CALL UPDAT1('LAMBDAs', LAMBDA, NCL, NCU)
           CALL UPDAT1('Q-Fs', QSTF, NWL, NQSTO)
           CALL UPDAT1('Q-Bs', QSTB, NWL, NQSTO)
           CALL UPDAT3('FRACs', FRAC, 1, 3)
           CALL UPDAT3('TRs', TR, 1, 3)
      ENDIF
C
C  End of Changes . . .
C
  999 RETURN
 2010 FORMAT(T10,'Enter number of plants left [',I2.2,'] >- ',$)
 2020 FORMAT(T10,'Opps.  Can''t have ', I2, ' plants left.',/,
     &       T10,'Continue? (T/F) [', L1, '] >- ',$)
      END
*UPDAT1
      SUBROUTINE UPDAT1(WHO, DATA, ILO, IHI)
      CHARACTER*(*) WHO
      INTEGER ILO, IHI, ITRY, INDEX, LIMIT
      DOUBLE PRECISION DATA(ILO:IHI), DTEMP
      LOGICAL CHANGE
*****************************************************************************
*  Routine to change a selected data item (in array DATA).                  *
*  The name of the data item is passed in character string WHO.  ILO        *
*  and IHI are the upper and lower limits of the data array.  If this       *
*  routine gets trapped in an "endless loop" (perhaps by reading a bad      *
*  data file) it will abort the program at this point.                      *
*****************************************************************************
      DATA LIMIT/25/
C
C  Change WHO? . . .
```

```
C
      ITRY = 0
      CHANGE = .FALSE.
   30 WRITE(*,2030)  WHO, CHANGE
      CALL GET1LO(CHANGE, CHANGE, *30)
      IF (CHANGE) THEN
   35      WRITE(*,2035) ILO, IHI
   40      CONTINUE
                ITRY = ITRY + 1
                WRITE(*,2040)
                READ (*,*,END=35,ERR=35) INDEX
                IF (INDEX .LT. ILO .OR. INDEX .GT. IHI) GO TO 50
   48           DTEMP = DATA(INDEX)
                WRITE(*,2045) DTEMP
                CALL GET1DP(DTEMP, DTEMP, *48)
                IF (ITRY .GT. LIMIT) GO TO 999
           GO TO 40
      ENDIF
   50 RETURN
C
C  This section of code is used to trap an "endless loop".  The
C  code will be executed if ITRY is greater than LIMIT.  This error
C  recovery was found to be necessary if the program is invoked from
C  a batch environment.
C
  999 WRITE(*,2099)
      STOP 2099
 2030 FORMAT(T10,'Change any ', A, ' (T/F) [', L1, '] >- ',$)
 2035 FORMAT(T10,'Enter INDEX, in range ', I2, ' to ', I2, '.',/,
     &       T10,'Use out-of-range (like  -99) to stop: ')
 2040 FORMAT(T10,'> ',$)
 2045 FORMAT(T10,'Current value is:  ',1P,G14.6,' --> ',$)
 2099 FORMAT(T10,'ITRY exceeds LIMIT...program ABORTing.')
      END
*UPDAT2
      SUBROUTINE UPDAT2(WHO, DATA, ILO, IHI)
$include:'comsize.i46'
      CHARACTER*(*) WHO, STRING*4
      INTEGER ILO, IHI, INDEX
      DOUBLE PRECISION DATA(ILO:IHI)
      LOGICAL CHANGE, DOVOL
      INTRINSIC INDEX
**********************************************************************
*  Routine similar to UPDAT1, except it is designed for large arrays   *
*  like AREA and VOL, which will be read from a file.  If the AREA has *
*  been changed at this time sweep, then the VOL must change too.      *
*     *NOTE*  In this routine, INDEX is the Fortran intrinsic function *
*  for sub-string location, which is a different usage than in UPDAT1. *
```

```
**************************************************************************
      DATA DOVOL/.FALSE./
C
C  Change WHO (read new data from a file).  Keep track of AREA change,
C  since the volume must change also.
C
      IF (DOVOL) THEN
           WRITE(*,2000)
           READ (LUXPI,*,ERR=998,END=999)
           READ (LUXPI,*,ERR=998,END=999) DATA
           DOVOL = .FALSE.
      ELSE
  130      WRITE(*,2130)  WHO, CHANGE
           CALL GET1LO(CHANGE, CHANGE, *130)
           IF (CHANGE) THEN
                IF (INDEX(WHO,'AREA') .NE. 0) THEN
                     DOVOL = .TRUE.
                ELSE
                     DOVOL = .FALSE.
                ENDIF
                READ (LUXPI,*,ERR=998,END=999)
                READ (LUXPI,*,ERR=998,END=999) DATA
           ENDIF
      ENDIF
      RETURN
 2130 FORMAT(T10,'Change any ', A, ' (T/F) [', L1, '] >- ',$)
C
C  The following code is used to trap errors due to bad data,
C  early end-of-file, and so forth.
C
  998 IF ( EOF(LUXPI) ) THEN
           WRITE(*,2120) 'Early END-OF-FILE.'
      ENDIF
      BACKSPACE LUXPI
      READ (LUXPI,1100,END=999) STRING
      IF (INDEX(STRING,CHAR(26)) .NE. 0 ) THEN
           WRITE(*,2120) 'Early END-OF-FILE.'
           WRITE(*,2110)  STRING
           STOP 997
      ELSE
           WRITE(*,2120) 'Bad data?'
           WRITE(*,2110)  STRING
           STOP 998
      ENDIF
  999 STOP 999
 1100 FORMAT(A)
 2000 FORMAT(T10,'Since AREAs have changed, then the VOLs must also.')
 2110 FORMAT(T10,'The last record read was:  ',/,1X,A)
```

```
 2120 FORMAT(/,T10,'*** W H O O P S ! ***     ',A)
      END
*UPDAT3
      SUBROUTINE UPDAT3(WHO, DATA, ILO, IHI)
      CHARACTER*(*) WHO
      INTEGER ILO, IHI
      DOUBLE PRECISION DATA(ILO:IHI)
      LOGICAL CHANGE
*****************************************************************************
*  Routine similar to UPDAT1, designed for small arrays like FRAC and  *
*  TR, where all (3) elements change at once.                                *
*****************************************************************************
C
C  Change WHO? (3) . . .
C
 2130 FORMAT(T10,'Change any ', A, ' (T/F) [', L1, '] >- ',$)
 2135 FORMAT(10X,'Current values are:  ',1P,3G14.6)
 2140 FORMAT(10X,'Enter ', I2, ' desired ', A, /, 8X, '> ', $)
      CHANGE = .FALSE.
  130 WRITE(*,2130)  WHO, CHANGE
      CALL GET1LO(CHANGE, CHANGE, *130)
      IF (CHANGE) THEN
  135      WRITE(*,2135) DATA
           WRITE(*,2140) IHI-ILO+1, WHO
           READ(*,*,END=135,ERR=135) DATA
      ENDIF
      RETURN
      END
```

```
*FINDQ      FILE: FINDQ46.FOR        Last Revision: November 14, 1989
C
C*--------------------------------------------------------------------*
C*            S U B R O U T I N E      F I N D Q                      *
C*                                                                    *
C*    THIS SUBROUTINE IS DESIGNED TO COMPUTE THE Q COEFFICIENTS.      *
C*    THIS SUBROUTINE IS CALLED FROM THE DTADRVR SUBROUTINE.          *
C*--------------------------------------------------------------------*
C
      SUBROUTINE FINDQ
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      DOUBLE PRECISION    F1, F2, F3, F1P1, F2P1, F3P1
C
C  Set up local, non-indexed, copies of variables.
C
      F1    = FRAC(1)
      F2    = FRAC(2)
      F3    = FRAC(3)
      F1P1  = 1.D0 + F1
      F2P1  = 1.D0 + F2
      F3P1  = 1.D0 + F3
      Q(11) = TR(1)
      Q(15) = TR(2)
      Q(19) = TR(3)
C
C  Compute the Q values.
C
      Q( 1) = (     Q(11) +      Q(15) +      Q(19))
      Q( 2) = (F1P1*Q(11) + F2P1*Q(15) + F3P1*Q(19))
      Q( 3) = (  F1*Q(11) +   F2*Q(15) +   F3*Q(19))
      Q( 4) =  Q(3)
      Q( 5) = (F2P1*Q(15) + F3P1*Q(19))
      Q( 6) = (  F2*Q(15) +   F3*Q(19))
      Q( 7) =  F3P1*Q(19)
      Q( 8) =  F3 * Q(19)
      Q( 9) =  F1P1*Q(11)
      Q(10) =  F1 * Q(11)
C**   Q(11) =   specified above
      Q(12) =  F1 * Q(11)
      Q(13) =  F2P1*Q(15)
      Q(14) =  F2 * Q(15)
C**   Q(15) =   specified above
      Q(16) =  F2 * Q(15)
      Q(17) =  F3P1*Q(19)
      Q(18) =  F3 * Q(19)
C**   Q(19) =   specified above
      Q(20) =  F3 * Q(19)
```

```
      RETURN
      END
```

```
*MTXA     FILE: MTRXA46.FOR          Last revision:  November 14, 1989
C
C
C*-------------------------------------------------------------------*
C*                                                                   *
C*   P L A N T    T R A N S P O R T   T H R E E   L E A F           *
C*                          M O D E L                                *
C*                        (VERSION 4.6)                              *
C*                                                                   *
C*    PROGRAM MANAGER : DR. CRAIG MCFARLANE                          *
C*                                                                   *
C*    PROGRAMMERS: DAVID E. CAWLFIELD            01/01/88            *
C*                 GILBERT A. BACHELOR           11/03/88            *
C*                 OREGON STATE UNIVERSITY                           *
C*                 SOIL SCIENCE DEPARTMENT                           *
C*                 CORVALLIS, OREGON 97331                           *
C*                                                                   *
C*    PROGRAM DEVELOPED FOR EPA GRANT NO. CR811940-01-03             *
C*                                                                   *
C*                    ALL RIGHTS RESERVED                            *
C*-------------------------------------------------------------------*
C*           S U B R O U T I N E       M T X A                       *
C*    THIS SUBROUTINE IS TO COMPUTE THE COEFFICIENTS FOR ALL OF THE  *
C*    COMPARTMENT INFLOWS AND OUTFLOWS.  THE VALUE COMPUTED IS       *
C*    STORED IN MATRIX A AND MATRIX A IS A GLOBAL TWO DIMENSIONAL    *
C*    ARRAY.  THE SYSTEM EQUATIONS USED FOR THE THREE LEAF MODEL HAVE*
C*    SIMILAR TERMS BUT DIFFERENT INDEXES FOR DIFFERENT COMPART-     *
C*    MENTS, THEREFORE FUNCTIONS (FD, FQ, FDQ, ETC.) ARE             *
C*    USED TO SIMPLIFY THE CODING.  PLEASE REFER TO THE MATHEMATICAL *
C*    MODEL PAPER FOR THE EXPLANATION OF THE SYSTEM OF EQUATIONS USED.*
C*    THIS SUBROUTINE IS CALLED FROM THE MAIN PROGRAM IN FILE        *
C*    "PLANT46.FOR".                                                 *
C*                                                                   *
C*    SUBROUTINE CALLS                                               *
C*    ----------------                                               *
C*     CALL LSTMTX() = LIST MATRIX A TO OUTPUT FILE 'xxxxx.DAO'.     *
C*                     THE CODE IS IN FILE 'DTADR46.FOR'.            *
C*                                                                   *
C*    FUNCTION CALLS, CONTAINED IN THIS FILE                         *
C*    --------------                                                 *
C*      FD = [ DIFU()*AREA() ] / DELTAX()                            *
C*      FQ = Q()*[1. - SIGMA()]                                      *
C*      FV = VOL() * [1. + BETA()]                                   *
C*     FDQ = FD() + FQ()                                             *
C*     FDV = FD(i) /  FV(j)                                          *
C*    FDQV = FDQ(i) /  FV(j)                                         *
C*     FQB = QSTB(i) / [VOL(j) * (1. + BETA(j))]                     *
C*     FQF = QSTF(i) / [VOL(j) * (1. + BETA(j))]                     *
```

```
C*     FHC = [DCH * AREA() * HC] / DELTAX()                              *
C*    FAIR = [DCH * AREA() * CAIR] / DELTAX()                            *
C*-------------------------------------------------------------------------*
      SUBROUTINE MTXA
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER             I, J
      DOUBLE PRECISION    FD, FQ, FV, FDQ, FDV, FDQV, FQB, FQF,
     &                    FHC, FAIR
      EXTERNAL            FD, FQ, FV, FDQ, FDV, FDQV, FQB, FQF,
     &                    FHC, FAIR
C
C  Initialize matrix A to zero; matrix A(NCPMT,NCPMT) is global
C
      DO 10, I = NCL, NCU
         DO 10, J = NCL, NCU
           A(I,J) = 0.D0
  10  CONTINUE
C
C  Assign non-zero values to specific elements
C
C--SOIL
      A(-1,-1) = -(  PLEFT*(FD(0) + FQ(1))/FV(-1) + LAMBDA(-1) )
      A(-1, 0) =     PLEFT*FDV(0,0)
C--ROOT FREE SPACE REGION
      A( 0,-1) =    (FD(0) + FQ(1)) / FV(-1)
      A( 0, 0) = -( (FD(0) + FDQ(1) + QSTF(0))/FV(0) + LAMBDA(0) )
      A( 0, 1) =     FQB(0,1)
      A( 0, 2) =     FDV(1,2)
C  ROOT EXTERIOR CELLS STORAGE
      A( 1, 0) =     FQF(0,0)
      A( 1, 1) = -(  FQB(0,1) + LAMBDA(1) )
C--ROOT
      A( 2, 0) =     FDQV(1,0)
      A( 2, 2) = -( (FD(1) + FDQ(2) + FD(4) + QSTF(1))
     &               /FV(2) + LAMBDA(2) )
      A( 2, 3) =     FQB(1,3)
      A( 2, 4) =     FDQV(4,4)
      A( 2, 5) =     FDV(2,5)
C
      A( 3, 2) =     FQF(1,2)
      A( 3, 3) = -( (QSTB(1) + QSTB(2))/FV(3) + LAMBDA(3) )
      A( 3, 4) =     FQF(2,4)
C
      A( 4, 2) =     FDV(4,2)
      A( 4, 3) =     FQB(2,3)
      A( 4, 4) = -( (FDQ(4) + FD(3) + QSTF(2))
     &               /FV(4) + LAMBDA(4) )
```

```
      A( 4, 7) =     FDQV(3,7)
C--BOTTOM STEM
      A( 5, 2) =     FDQV(2,2)
      A( 5, 5) = -( (FD(2) + FDQ(5) + FDQ(9) + QSTF(3))
     &               /FV(5) + LAMBDA(5) )
      A( 5, 6) =     FQB(3,6)
      A( 5, 8) =     FDV(5, 8)
      A( 5,14) =     FDV(9,14)
C
      A( 6, 5) =     FQF(3,5)
      A( 6, 6) = -( (QSTB(3) + QSTB(4))/FV(6) + LAMBDA(6) )
      A( 6, 7) =     FQF(4,7)
C
      A( 7, 4) =     FDV(3,4)
      A( 7, 6) =     FQB(4,6)
      A( 7, 7) = -( (FDQ(3) + FD(6) + FD(12) + QSTF(4))
     &               /FV(7) + LAMBDA(7) )
      A( 7,10) =     FDQV(6,10)
      A( 7,16) =     FDQV(12,16)
C--MID STEM
      A( 8, 5) =     FDQV(5,5)
      A( 8, 8) = -( (FD(5) + FDQ(7) + FDQ(13) + QSTF(5))
     &               /FV(8) + LAMBDA(8) )
      A( 8, 9) =     FQB(5,9)
      A( 8,11) =     FDV(7,11)
      A( 8,17) =     FDV(13,17)
C
      A( 9, 8) =     FQF(5,8)
      A( 9, 9) = -( (QSTB(5) + QSTB(6))/FV(9) + LAMBDA(9) )
      A( 9,10) =     FQF(6,10)
C
      A(10, 7) =     FDV(6,7)
      A(10, 9) =     FQB(6,9)
      A(10,10) = -( (FDQ(6) + FD(8) + FD(16) + QSTF(6))
     &               /FV(10) + LAMBDA(10) )
      A(10,13) =     FDQV(8,13)
      A(10,19) =     FDQV(16,19)
C--TOP STEM
      A(11, 8) =     FDQV(7,8)
      A(11,11) = -( (FD(7) + FDQ(17) + QSTF(7))
     &               /FV(11) + LAMBDA(11) )
      A(11,12) =     FQB(7,12)
      A(11,20) =     FDV(17,20)
C
      A(12,11) =     FQF(7,11)
      A(12,12) = -( (QSTB(7) + QSTB(8))/FV(12) + LAMBDA(12) )
      A(12,13) =     FQF(8,13)
C
```

```
      A(13,10) =     FDV(8,10)
      A(13,12) =     FQB(8,12)
      A(13,13) = -( (FDQ(8) + FD(20) + QSTF(8))
     &               /FV(13) + LAMBDA(13) )
      A(13,22) =     FDQV(20,22)
C--LEAF 1
      A(14, 5) =     FDQV(9,5)
      A(14,14) = -( (FDQ(10) + FD(9) + FHC(11) + QSTF(9))
     &               /FV(14) + LAMBDA(14) )
      A(14,15) =     FQB(9,15)
      A(14,16) =     FDV(10,16)
C
      A(15,14) =     FQF(9,14)
      A(15,15) = -( (QSTB(9) + QSTB(10))/FV(15) + LAMBDA(15) )
      A(15,16) =     FQF(10,16)
C
      A(16, 7) =     FDV(12,7)
      A(16,14) =     FDQV(10,14)
      A(16,15) =     FQB(10,15)
      A(16,16) = -( (FDQ(12) + FD(10) + QSTF(10))
     &               /FV(16) + LAMBDA(16) )
C--LEAF 2
      A(17, 8) =     FDQV(13,8)
      A(17,17) = -( (FD(13) + FDQ(14) + FHC(15) + QSTF(11))
     &               /FV(17) + LAMBDA(17) )
      A(17,18) =     FQB(11,18)
      A(17,19) =     FDV(14,19)
C
      A(18,17) =     FQF(11,17)
      A(18,18) = -( (QSTB(11) + QSTB(12))/FV(18) + LAMBDA(18) )
      A(18,19) =     FQF(12,19)
C
      A(19,10) =     FDV(16,10)
      A(19,17) =     FDQV(14,17)
      A(19,18) =     FQB(12,18)
      A(19,19) = -( (FDQ(16) + FD(14) + QSTF(12))
     &               /FV(19) + LAMBDA(19) )
C--LEAF 3
      A(20,11) =     FDQV(17,11)
      A(20,20) = -( (FD(17) + FDQ(18) + FHC(19) + QSTF(13))
     &               /FV(20) + LAMBDA(20) )
      A(20,21) =     FQB(13,21)
      A(20,22) =     FDV(18,22)
C
      A(21,20) =     FQF(13,20)
      A(21,21) = -( (QSTB(13) + QSTB(14))/FV(21) + LAMBDA(21) )
      A(21,22) =     FQF(14,22)
C
```

```
      A(22,13) =     FDV(20,13)
      A(22,20) =     FDQV(18,20)
      A(22,21) =     FQB(14,21)
      A(22,22) = -( (FDQ(20) + FD(18) + QSTF(14))
     &              /FV(22) + LAMBDA(22) )
C
C  Initialize source array.
C
      DO 30, I = NCL, NCU
           SOURC(I) = 0.D0
  30  CONTINUE
C
      SOURC(14) = FAIR(11)
      SOURC(17) = FAIR(15)
      SOURC(20) = FAIR(19)
C
C-----OUTPUT COMPUTED MATRIX
C
      CALL LSTMTX(A)
      RETURN
      END
*FD
      DOUBLE PRECISION FUNCTION FD(I)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I
************************************************************************
*  Used by mtrxa to compute matrix elements.                           *
*  Computes:  [ DIFU()*AREA() ] / DELTAX()                             *
************************************************************************
      FD = ( DIFU(I)*AREA(I) ) / DELTAX(I)
      RETURN
      END
*FQ
      DOUBLE PRECISION FUNCTION FQ(I)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I
************************************************************************
*  Used by mtrxa to compute matrix elements.                           *
*  Computes:  Q()*[1. - SIGMA()]                                       *
************************************************************************
      FQ = Q(I) * (1.D0 - SIGMA(I))
      RETURN
      END
*FV
      DOUBLE PRECISION FUNCTION FV(I)
$INCLUDE:'COMSIZE.I46'
```

```
$INCLUDE:'COMBLOC.I46'
      INTEGER I
*****************************************************************************
*  Used by mtrxa to compute matrix elements.                                *
*  Computes:  VOL() * [1. + BETA()]                                         *
*****************************************************************************
      FV = VOL(I) * (1.D0 + SB(I))
      RETURN
      END
*FDQ
      DOUBLE PRECISION FUNCTION FDQ(I)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I
      DOUBLE PRECISION    FD, FQ
      EXTERNAL            FD, FQ
*****************************************************************************
*  Combines FD and FQ.                                                      *
*  computes:  FD() + FQ()                                                   *
*****************************************************************************
      FDQ = FD(I) + FQ(I)
      RETURN
      END
*FDV
      DOUBLE PRECISION FUNCTION FDV(I, J)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I, J
      DOUBLE PRECISION    FD, FV
      EXTERNAL            FD, FV
*****************************************************************************
*  Combines FD with FV using two arguments.                                 *
*  Computes:  FD(i) /  FV(j)                                                *
*****************************************************************************
      FDV = FD(I) / FV(J)
      RETURN
      END
*FDQV
      DOUBLE PRECISION FUNCTION FDQV(I, J)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I, J
      DOUBLE PRECISION    FDQ, FV
      EXTERNAL            FDQ, FV
*****************************************************************************
*  Combines FDQ with FV using two arguments.                                *
*  Computes:  FDQ(i) /  FV(j)                                               *
*****************************************************************************
```

```
      FDQV = FDQ(I) / FV(J)
      RETURN
      END
*FQB
      DOUBLE PRECISION FUNCTION FQB(I, J)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I, J
      DOUBLE PRECISION    FV
      EXTERNAL            FV
*****************************************************************************
*  Used by mtrxa to compute matrix elements.                                *
*  Computes:  QSTB(i) / [VOL(j) * (1. + BETA(j))]                           *
*****************************************************************************
      FQB = QSTB(I) / FV(J)
      RETURN
      END
*FQF
      DOUBLE PRECISION FUNCTION FQF(I, J)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I, J
      DOUBLE PRECISION    FV
      EXTERNAL            FV
*****************************************************************************
*  Used by mtrxa to compute matrix elements.                                *
*  Computes:  QSTF(i) / [VOL(j) * (1. + BETA(j))]                           *
*****************************************************************************
      FQF = QSTF(I) / FV(J)
      RETURN
      END
*FHC
      DOUBLE PRECISION FUNCTION FHC(I)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I
*****************************************************************************
*  Used by mtrxa to compute matrix elements.                                *
*  Computes:  [DCH * AREA() * HC] / DELTAX()                                *
*****************************************************************************
      FHC = (DCH * AREA(I) * HC) / DELTAX(I)
      RETURN
      END
*FAIR
      DOUBLE PRECISION FUNCTION FAIR(I)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I
```

```
**************************************************************************
*  Used by mtrxa to compute source array elements.                       *
*  Computes:  [DCH * AREA() * CAIR] / DELTAX()                           *
**************************************************************************
      FAIR = (DCH * AREA(I) * CAIR) / DELTAX(I)
      RETURN
      END
*LSTMTX
C*--------------------------------------------------------------------*
C*            S U B R O U T I N E     L S T M T X                     *
C*     THIS SUBROUTINE PRINTS MATRIX A TO FILE "xxxxx.DAO".           *
C*     THIS FILE CAN BE USED AS PROGRAM VERIFICATION AND THE MATRIX   *
C*     CAN BE USED TO COMPUTE STEADY STATE.                           *
C*--------------------------------------------------------------------*
C
      SUBROUTINE LSTMTX(OMTXA)
$INCLUDE:'COMSIZE.I46'
      INTEGER             I, J
      DOUBLE PRECISION    OMTXA(NCL:NCU, NCL:NCU)
C
C-----PARAMETER DEFINITION
C        1. OMTXA = TWO DIMENSIONAL MATRIX.
C
C-----OUTPUT FORMAT
 2000 FORMAT(1P, 30(1X, E16.8))
C
C-----LIST THE COMPUTED M BY M MATRIX A
      DO 10, I = NCL, NCU
        WRITE(LUDAO,2000) (OMTXA(I,J), J = NCL, NCU)
  10  CONTINUE
      RETURN
      END
```

```
*INTGRL      FILE: INRES46.FOR        Last revision:  November 14, 1989
C
C  Revised by Gilbert A. Bachelor, November 1988.
C  Further modified by D. E. Cawlfield, '88 to present.
C
      SUBROUTINE INTGRL(FIRST)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      LOGICAL   FIRST
      INTEGER   I, J, IERR, ITER, ITRIP
      REAL      SECOND, START, START0, ELAPS1, ELAPS2
      EXTERNAL  SECOND
      DOUBLE PRECISION    ETOA(NCL:NCU,NCL:NCU), AINV(NCL:NCU,NCL:NCU)
      DOUBLE PRECISION    TNOW, HCTOL, FRACT
      DOUBLE PRECISION    ATEMP(NCL:NCU,NCL:NCU), STEMP(NCL:NCU)
*************************************************************************
*      Infinite time resolution - Matrix exponential method.  This      *
*      method is independent of the time increment.                      *
*      Uses MATEXP subroutine in file "MATEXP.FOR".                      *
*      Uses LUFINV subroutine in library MATS.                           *
*************************************************************************
      HCTOL = 1.D-5
      START0 = SECOND()
C
C  This routine is entered after Q <-- FINDQ and (little) A <-- MTRXA.
C
C  Set ATEMP = -dt * A; also set A = -A.
C
      DO  10, I = NCL, NCU
           DO  10, J = NCL, NCU
                ATEMP(I,J) = - DT * A(I,J)
                A(I,J) = - A(I,J)
   10 CONTINUE
C
C  Call MATEXP to compute EXP(-ATEMP) = EXP(DT * A); ETOA is result.
C
      CALL MATEXP(NCPMT,ATEMP,ETOA,IERR)
      IF (IERR .LT. 0) GOTO 900
C
C  If CAIR = 0 and HC < HCTOL, set SOURC to zero and skip the
C  code to compute the modified form of SOURC --
C
      IF (CAIR.EQ.0.D0 .AND. HC.LT.HCTOL) THEN
           DO  15, I = NCL, NCU
                SOURC(I) = 0.D0
   15      CONTINUE
      ELSE
C
```

```
C  -- otherwise, compute modified SOURC.
C
C  Set ATEMP = I - ETOA.
C
           DO  20, I = NCL, NCU
                DO  20, J = NCL, NCU
                     ATEMP(I,J) = - ETOA(I,J)
                     IF (I .EQ. J) ATEMP(I,J) = 1.D0 + ATEMP(I,J)
   20      CONTINUE
C
C  Set STEMP = ATEMP * SOURC.
C
           DO  30, I = NCL, NCU
                STEMP(I) = 0.D0
                DO  30, J = NCL, NCU
                     STEMP(I) = STEMP(I) + ATEMP(I,J)*SOURC(J)
   30      CONTINUE
C
C  Call LUFINV to invert A (now -A); inverse to AINV;
C  ATEMP is a work array.
C
           CALL LUFINV(NCPMT,A,ATEMP,AINV,IERR)
           IF (IERR .LT. 0) GOTO 950
C
C  Set SOURC = AINV * STEMP
C            = - INV(A) * (I - ETOA) * SOURC  (Original A and SOURC)
C
           DO  40, I = NCL, NCU
                SOURC(I) = 0.D0
                DO  40, J = NCL, NCU
                     SOURC(I) = SOURC(I) + AINV(I,J)*STEMP(J)
   40      CONTINUE
      ENDIF
C  End of code to compute modified SOURC.
C
C  Set XOLD = XNOW to prepare for loop.  XNOW is initial mass
C  distribution here.
C
      DO  50, I = NCL, NCU
           XOLD(I) = XNOW(I)
   50 CONTINUE
C
C  Setup variables TNOW, buffer counters, etc.,
C  only when called the first time.
C
      IF (FIRST)  THEN
           IBUFF = 0
           ITIME = 0
```

```
            TNOW = TINITIAL
            REWIND (LUDAC)
            FRACT = 0.50D0
         ENDIF
         ELAPS1  = SECOND() - START0
         START   = SECOND()
C
C  For fixed DT's, we may calculate the "trip count" much the
C  same as Fortran '77 DO-loops --
C     itrip = max(int((tstop - (tnow + 0.1*dt) + dt) / dt )), 0)
C
         ITRIP = INT( (TSTOP - TNOW + FRACT*DT) / DT )
         TPRT  = 0.D0
         ITER  = 0
C
C                    <<<   M A I N   L O O P   >>>
C
  200 IF (ITER .LT. ITRIP) THEN
C
C  Set XNOW = ETOA * XOLD + SOURC
C
            DO  60, I = NCL, NCU
                XNOW(I) = SOURC(I)
                DO  60, J = NCL, NCU
                    XNOW(I) = XNOW(I) + ETOA(I,J)*XOLD(J)
   60       CONTINUE
            TNOW = TNOW + DT
            ITER = ITER + 1
Cdbg        PRINT 2010, ITER, TNOW, XNOW
C
C  EOUT stores current mass distribution in MASS array, if current
C  time is one of the times when the distribution is to be printed.
C
            CALL EOUT(TNOW)
C
C  Set XOLD = XNOW and loop back.
C
            DO  70, I = NCL, NCU
                XOLD(I) = XNOW(I)
   70       CONTINUE
            GO TO 200
         ENDIF
  210 CONTINUE
         TSTOP = TNOW
         ELAPS2 = SECOND() - START
         WRITE(LUPRN,2040) ELAPS1, ELAPS2
         RETURN
C
```

```
C  Error handling --
C
  900 PRINT 1000, IERR
      GOTO 999
  950 PRINT 1050, IERR
  999 STOP 950
C
C  Formats
C
 1000 FORMAT(1X,'INTGRL: Error from MATEXP =', I3)
 1050 FORMAT(1X,'INTGRL: Error from LUFINV =', I3)
 2010 FORMAT(1X,'I:',I5,2X,'T= ',G12.6, 4(1X,G12.6), /,
     &  (T13, 5(1X, G12.6, :)) )
C  Note use of re-scan in above format
 2020 FORMAT(//,T5,'The additional TEST stops here, with TNOW = ',
     &  1P, G10.4, ', TSTOP= ',G10.4,/,
     & T40, 0P,     '  ITER = ', I3, ', and ITRIP = ', I3)
 2040 FORMAT(/,T10,'Integration took ',F12.3,' + ',F12.3,' seconds.')
      END
```

```
*MATEXP                                  Last Revision:  November 14, 1989
C                                        File:  MATEXP.FOR   v 2.0
C  Originally coded by Tom Lindstrom and Dave Cawlfield.
C  Modified by Gil Bachelor.  (October 1988)
C  Further modified by D. E. Cawlfield, '88 to present.
C
      SUBROUTINE MATEXP(NDIM, W, ARRAY, IERR)
C  ARGUMENTS
      INTEGER NDIM, IERR
      DOUBLE PRECISION  W(NDIM,NDIM), ARRAY(NDIM,NDIM)
C  CONSTANTS
      INTEGER MAXN, NLOOPS
      DOUBLE PRECISION TOLRAN
      PARAMETER(MAXN=35, NLOOPS = 100, TOLRAN = 1.E-13)
C  LOCAL VARIABLES AND ARRAYS
      INTEGER I, J, K, L, N, NSCALE
      DOUBLE PRECISION ASUM, SCALE, TEMP
      DOUBLE PRECISION TESTG, TESTN
      DOUBLE PRECISION SUM(MAXN,MAXN), OLDSUM
      DOUBLE PRECISION TRM(MAXN,MAXN), PROD(MAXN,MAXN)
*************************************************************************
*  Compute the matrix exponential function.                             *
*  Sets ARRAY to EXP(-W); both W and ARRAY are matrices of dimension    *
*  NDIM by NDIM.  W is altered.  IERR is set to a negative value if     *
*  any error occurs.                                                    *
*************************************************************************
      IERR=0
      IF (NDIM.GT.MAXN) THEN
           PRINT *, 'N > MAXN (=', MAXN, ')'
           STOP 763
      ENDIF
C
C  FIND LARGEST DIAGONAL ELEMENT
C
      TESTG=DABS(W(1,1))
      DO 60, I=2,NDIM
           TESTN=DABS(W(I,I))
           IF(TESTN.GE.TESTG) TESTG=TESTN
   60 CONTINUE
C
C  COMPUTE THE APPROPRIATE POWER OF 2 SCALING FACTOR
C
      DO 62, K=1,100
           TESTN=TESTG/2.0D0
           IF(TESTN.LE.1.0D0) GO TO 65
           TESTG=TESTN
   62 CONTINUE
      IERR=-5
```

```
      WRITE(*,*)' MATEXP WARNING!! Maximum diagonal entry >2^100'
   65 NSCALE=K
      SCALE=2.0D0**NSCALE
C
C  SCALE ALL THE MATRIX ELEMENTS BY "SCALE".
C
      DO 67, I=1,NDIM
           DO 68, J=1,NDIM
                W(I,J)=W(I,J)/SCALE
   68      CONTINUE
   67 CONTINUE
C
C  INITIALIZE THE ARRAYS FOR COMPUTING THE MATRIX EXPONENTIAL FUNCTION
C  BY SETTING THE SUM AND TRM ARRAYS TO THE IDENTITY MATRIX.
C
      DO 100, I=1,NDIM
           DO 101, J=1,NDIM
                SUM(I,J)=0.0D0
                TRM(I,J)=0.0D0
  101      CONTINUE
  100 CONTINUE
C
      DO 105, I=1,NDIM
           SUM(I,I)=1.0D0
           TRM(I,I)=1.0D0
  105 CONTINUE
C
C  ADD TERMS TO THE SERIES FOR THE EXPONENTIAL FUNCTION UNTIL
C  CONVERGENCE OCCURS.  MULTIPLY TRM BY W AND DIVIDE BY K;
C  ADD TRM TO SUM. TRM IS OF THE FORM (W**K)/(K!).
C  COMPUTE ASUM = ABSOLUTE SUM OF DIFFERENCE BETWEEN NEW SUM
C  AND OLD SUM.
C
      DO 110, K=1,NLOOPS
           N=K
           ASUM=0.0D0
           DO 130, I=1,NDIM
                DO 135, J=1,NDIM
                     TEMP=0.0D0
                     DO 120, L=1,NDIM
                          TEMP=TEMP+W(I,L)*TRM(L,J)/DBLE(K)
  120                CONTINUE
                     PROD(I,J)=TEMP
                     OLDSUM=SUM(I,J)
                     SUM(I,J)=SUM(I,J)+TEMP
                     ASUM=ASUM+DABS(SUM(I,J)-OLDSUM)
  135           CONTINUE
  130      CONTINUE
```

```
C
C  COPY NEW TERM (PROD) BACK INTO TRM ARRAY.
C
           DO 140, I=1,NDIM
                DO 141, J=1,NDIM
                     TRM(I,J)=PROD(I,J)
  141           CONTINUE
  140      CONTINUE
C
C  IF ASUM <= TOLERANCE, TERMINATE LOOP.
C
           IF(ASUM.LE.TOLRAN) GO TO 1000
           IF(N.GE.NLOOPS) GO TO 3000
  110 CONTINUE
 1000 CONTINUE
C
C  INVERT THE MATRIX EXPONENTIAL FUNCTION.  FIRST, COPY SUM ARRAY
C  BACK INTO W ARRAY.  ARGUMENTS "W" AND "ARRAY" FOR LUFINV MUST
C  BE OF SIZE NDIM BY NDIM.  "TRM" CAN BE LARGER (IT IS A WORKING
C  STORAGE ARRAY).
C
      DO 160, I=1,NDIM
           DO 161, J=1,NDIM
                W(I,J)=SUM(I,J)
  161      CONTINUE
  160 CONTINUE
C
      CALL LUFINV(NDIM, W, TRM, ARRAY, IERR)
      IF (IERR.LT.0) RETURN
C
C  RAISE ARRAY MATRIX TO THE 2**NSCALE POWER BY SQUARING IT NSCALE TIMES
C
      DO 400, L=1,NSCALE
           DO 405, I=1,NDIM
                DO 406, J=1,NDIM
                     PROD(I,J)=0.0D0
                     DO 407, K=1,NDIM
                          PROD(I,J)=PROD(I,J)+ARRAY(I,K)*ARRAY(K,J)
  407                CONTINUE
  406           CONTINUE
  405      CONTINUE
           DO 408, I=1,NDIM
                DO 409, J=1,NDIM
                     ARRAY(I,J)=PROD(I,J)
  409           CONTINUE
  408      CONTINUE
  400 CONTINUE
C
```

```
      GO TO 4000
C
 3000 WRITE(*,*)' MATEXP WARNING!! Lack of convergence in NLOOPS.'
      IERR=-6
 4000 RETURN
      END
```

```
*EOUT      FILE: EOUTP46.FOR          Last Revision:  November 14, 1989
C
      SUBROUTINE EOUT(TNOW)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
C  Parameters...
      DOUBLE PRECISION    TNOW
C  Local control
      INTEGER             IT, IIT, J
**********************************************************************
*  Store the values obtained by INTGRL into the appropriate arrays.  *
*  When the arrays are full, as indicated by the IBUFF pointer, then *
*  call BUFOUT to write the arrays to a file and re-set IBUFF (flush *
*  the buffer).                                                      *
*                    David E. Cawlfield,  Fall 1987                  *
*        Revised by: Gilbert A. Bachelor, October 1988               *
**********************************************************************
      DATA IIT/0/
C
C-----Indicate every simulated hour to screen
      IT = INT(TNOW)
      IF (IT .GT. IIT) THEN
           CALL CURSOR(20,42)
           WRITE(*,50) IT
           CALL CURSOR(20,50)
           IIT = IT
   50      FORMAT(I8,$)
      ENDIF
      TPRT = TPRT + DT
C
C-----Store mass at user specified time increments
C
      IF (TPRT .GT. PRINTVL) THEN
C
C-----Increment next time storage location
C
           ITIME = ITIME + 1
           IBUFF = IBUFF + 1
           DO 80 J = NCL, NCU
                MASS(IBUFF,J) = XNOW(J)
   80      CONTINUE
C
C-----Store current simulated time to storage area.
           TIMES(IBUFF) = TNOW
C
C-----Store number of plants left in last array element.  Used to
C     put a break in the report(s).
           NPLFT(IBUFF) = PLEFT
```

```
C
C-----Reset printing interval flag to zero for next storage
C     increments.  Dump Cache buffer when full!
           IF ( IBUFF .GE. NHOLD ) THEN
                CALL BUFOUT( IBUFF )
                IBUFF = 0
           ENDIF
           TPRT = 0.0
      ENDIF
      RETURN
      END
```

```
*REPORT     FILE: PRINT46.FOR          Last revision:  November 14, 1989
C
C
C*-------------------------------------------------------------------*
C*                                                                   *
C*    P L A N T    T R A N S P O R T   T H R E E   L E A F          *
C*                         M O D E L                                 *
C*                       (VERSION 4.6)                               *
C*                                                                   *
C*    PROGRAM MANAGER : DR. CRAIG MCFARLANE                          *
C*                                                                   *
C*    PROGRAMMERS: DAVID E. CAWLFIELD        01/01/88                *
C*                 GILBERT A. BACHELOR       11/14/88                *
C*                 OREGON STATE UNIVERSITY                           *
C*                 SOIL SCIENCE DEPARIMENT                           *
C*                 CORVALLIS, OREGON 97331                           *
C*                                                                   *
C*    PROGRAM DEVELOPED FOR EPA GRANT NO. CR811940-01-03             *
C*                                                                   *
C*                    ALL RIGHTS RESERVED                            *
C*                                                                   *
C*    CALL MSSGE3    : PRINT PROCESSING MESSAGE TO SCREEN AND        *
C*                     PERMIT USER TO EXIT PROGRAM BEFORE            *
C*                     STARTING SIMULATION.                          *
C*    CALL MENU      : OUTPUT OPTIONS PRESENTED IN SCREEN MENU       *
C*                     FORMAT.                                       *
C*    CALL PROUT     : OUTPUT INDIVIDUAL DATA IN COMPARIMENTS OVER   *
C*                     TIME.                                         *
C*    CALL PRHEAD    : PRINT PAGE HEADING.                           *
C*    CALL OUTGROUP  : OUTPUT GROUPED MASS REPORT.                   *
C*    CALL OUTLUMPS  : OUTPUT LUMPED  MASS REPORT.                   *
C*-------------------------------------------------------------------*
      SUBROUTINE REPORT
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER   IPICK
      EXTERNAL  CLS, SCRNHD, MSSGE3, MENU,
     &          PROUT, OUTGROUP
C
10    CALL CLS
      CALL SCRNHD
      CALL MENU(IPICK)
C
C-----EXIT COMMAND
      IF (IPICK .EQ. 6) GOTO 999
C
C-----OUTPUT INDIVIDUAL MASS IN COMPARIMENTS
      IF (IPICK .EQ. 1 .OR. IPICK .EQ. 5)
```

```
     &   CALL PROUT(1, LUFMO, NCL, NCU, MASS)
C
C-----OPTION 2 NOT USED IN THIS VERSION.
C
      IF (IPICK .EQ. 2 .OR. IPICK .EQ. 5) CONTINUE
C
C-----OUTPUT GROUPED MASS WITH ROOT, STEM, AND LEAVES.
C
      IF (IPICK .EQ. 3 .OR. IPICK .EQ. 5)  CALL OUTGROUP
C
C-----OUTPUT LUMPED MASS WITH ROOT, STEM, AND LEAVES.
C
      IF (IPICK .EQ. 4 .OR. IPICK .EQ. 5) CALL OUTLUMPS
C
C-----END OF RUN.
C
C-----RETURN TO SCREEN MENU UNTIL USER PICKED OPTION #6 TO EXIT.
      GOTO 10
 999  RETURN
      END
*MENU
C
C*--------------------------------------------------------------------*
C*              S U B R O U T I N E       M E N U                     *
C*  THIS SUBROUTINE DISPLAYS THE OUTPUT OPTIONS AS A SCREEN MENU.     *
C*  USER IS ALLOWED TO PICK ANY COMBINATION OF OUTPUT REPORTS.        *
C*  THE REPORT NUMBER SELECTED IS PASSED BACK TO THE CALLING ROUTINE. *
C*--------------------------------------------------------------------*
      SUBROUTINE MENU(OPICK)
      INTEGER OPICK, MAXPIK, IGOOF
      PARAMETER ( MAXPIK = 6 )
      LOGICAL   DONE
C
C  REPEAT...UNTIL <done>...
C
      DONE = .TRUE.
    5 WRITE(*,10)
   10 FORMAT(3(/),T10,'!!!!! Simulation has been completed !!!!!',
     * ///,T10,'Reort Selections --',/,
     *   /,T10,'[1] Individual Mass in Compartments. ',/,
     *     T10,'[2] [Inactive option]                ',/,
     *     T10,'[3] Grouped Mass; Root, Stems and Leaves.',/,
     *     T10,'[4] Lumped Mass; Soil, Root, Stems and Leaves.',/,
     *     T10,'[5] List all of the above.',/,
     *     T10,'[6] Done [with output.]',3(/),
     *     T10,'Please enter your selection >- ',$)
C
      READ(*,*,IOSTAT=IGOOF) OPICK
```

```
      IF (IGOOF .NE. 0 .OR. OPICK .LT. 1 .OR. OPICK .GT. MAXPIK) THEN
C-----Retry for user entering wrong choices.
           WRITE(*,2100) CHAR(7), MAXPIK
           DONE = .FALSE.
      ELSE
C-----Check for wrong selection.  But only when "done" (opt 6, now).
           IF (OPICK .EQ. 6) THEN
   15           WRITE(*,20)
                READ(*,30,ERR=15,END=15) DONE
           ENDIF
      ENDIF
      IF ( .NOT. DONE ) GO TO 5
      RETURN
   20 FORMAT(/,T10,'Are you sure (T/F) ? >- ',$)
   30 FORMAT(L1)
 2100 FORMAT(/,T10,A,'Should be a *number* between 1 and ',I2)
      END
*PROUT
C
C*--------------------------------------------------------------------*
C*              S U B R O U T I N E     P R O U T                     *
C*    THIS SUBROUTINE PRINTS OUT ALL THE CONTENTS OF THE PASSED       *
C*    "DATA" ARRAY.  THE OUTPUT INCLUDES A PAGE HEADER,               *
C*    PAGE NUMBER, PRINT OUT (LISTING) FILENAME, AND THE PLANT        *
C*    COMPARIMENT NUMBERS.  IT SKIPS TO A NEW PAGE EVERY 45           *
C*    LINES PRINTED.                                                  *
C*--------------------------------------------------------------------*
C
       SUBROUTINE PROUT(INDEX, LUS, IB, IE, DATA)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER I, J, JJ, JJJ, INDEX, LUS, IB, IE, JINC, LINE, NPLEFT
      DOUBLE PRECISION    DATA(NHOLD, NCL:NCU)
      LOGICAL FIRST
C
C-----Print out mass distributions to user specified output filename.
C     Printout includes page header, page number, output filename and
C     compartment numbers.
C-----The mass distribution file 'xxxxx.FMA'.
C     Does not include page headers. The output is as follows:
C        Column 1 : Time (in hours)
C        Column 2 - 31: Compartment 1 through Compartment ncpmt
C
      NPAGE = 1
      FIRST = .TRUE.
      IF (WIDE) THEN
           JINC  =  10-1
      ELSE
```

```
          JINC  =   5-1
        ENDIF
        JJ    = IB
        JJJ   = JJ + JINC
        CALL MASSHD(INDEX,1,JJ,JJJ)
        REWIND(LUDAC)
        IBUFF = NHOLD
        NPLEFT = PSTART
        LINE  = 0
   20 DO 50, I = 1, ITIME
C
C  Fill from Cache Buffer as necessary...
C
            IF ( IBUFF .GE. NHOLD ) THEN
                 CALL BUFIN(NHOLD)
                 IBUFF = 0
            ENDIF
C
C-----Skip to a new page for every 45 lines and print header
C
            LINE = LINE + 1
            IF ( LINE .NE. 1 .AND. MOD(LINE-1,45) .EQ. 0 ) THEN
                 CALL MASSHD(INDEX,2,JJ,JJJ)
            ENDIF
            IBUFF = IBUFF + 1
            IF (NPLEFT .NE. NPLFT(IBUFF)) THEN
                 NPLEFT = NPLFT(IBUFF)
                 WRITE(LUPRN,2015) NPLEFT
                 LINE = LINE + 3
            ENDIF
            WRITE(LUPRN,2010) TIMES(IBUFF),
     &                       (DATA(IBUFF,J), J= JJ, JJJ)
C
C  Write to "xxxxx.xxx" only on first pass.  Done this way to avoid
C  separate loop which must also read Cache Buffer...
C
            IF ( FIRST ) THEN
                 WRITE(LUS,2000)  TIMES(IBUFF),
     &                           (DATA(IBUFF,J), J = NCL, NCU)
            ENDIF
   50 CONTINUE
        FIRST = .FALSE.
C
C-----REPEAT PROCESS UNTIL FINISHED, THEN EXIT SUBROUTINE
C
      IF (JJJ .EQ. IE) GO TO 999
      NPAGE = NPAGE + 1
      JJ  = JJJ + 1
```

```
      JJJ = MIN(JJ + JINC, IE)
      CALL MASSHD(INDEX,1,JJ,JJJ)
      REWIND(LUDAC)
      IBUFF = NHOLD
      LINE = 0
      GOTO 20
C
  999 RETURN
C
C-----OUTPUT FORMATS (Assumes masses are non-negative)
C
 2000 FORMAT(1P, 34(E9.2))
 2010 FORMAT(T7, F7.2, 3X, 1P, 10(1X,E10.3))
 2015 FORMAT(/,T8,'Number of plants is now: ',I2,/)
      END
*OUTGROUP
C
C*--------------------------------------------------------------------*
C*          S U B R O U T I N E      O U T G R O U P                  *
C*    THIS SUBROUTINE CREATES THE GROUPED MASS REPORT, WHICH LISTS     *
C*    THE SUM OF THE SOLUTE MASSES IN THE ROOT COMPARTMENTS; THE SUM   *
C*    OF THE SOLUTE MASSES IN THE STEM BOTTOM, MIDDLE, AND TOP; THE    *
C*    SUM OF THE SOLUTE MASSES IN LEAF 1 THROUGH LEAF 3; AND THE SUM   *
C*    OF THE SOLUTE MASSES FOR THE TOTAL PLANT.                        *
C*    ARE PREDEFINED IN THE MODEL.                                     *
C*                                                                     *
C*    LOCAL VARIABLE DEFINITIONS                                       *
C*    --------------------------                                       *
C*     SUMR  = ROOT SUM                                                *
C*     SUMSB = STEM BOTTOM SUM                                         *
C*     SUMSM = STEM MIDDLE SUM                                         *
C*     SUMST = STEM TOP SUM                                            *
C*     SUML1 = LEAF 1 SUM                                              *
C*     SUML2 = LEAF 2 SUM                                              *
C*     SUML3 = LEAF 3 SUM                                              *
C*     TOTAL  = TOTAL PLANT MASS                                       *
C*--------------------------------------------------------------------*
      SUBROUTINE OUTGROUP
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      DOUBLE PRECISION SUMR, SUMSB, SUMSM, SUMST,
     &                 SUML1, SUML2, SUML3, TOTAL
      INTEGER   I, LINE, NPLEFT
C
      NPAGE = 1
      CALL GMASSHD
      REWIND(LUDAC)
      IBUFF = NHOLD
```

```
      NPLEFT = PSTART
      LINE  = 0
      DO 50, I = 1, ITIME
C
C  Fill the Cache Buffer as necessary...
C
           IF ( IBUFF .GE. NHOLD ) THEN
                CALL BUFIN(NHOLD)
                IBUFF = 0
           ENDIF
           IBUFF = IBUFF + 1
C
C  Do sums...
C
           SUMR  = MASS(IBUFF, 0) + MASS(IBUFF, 1)
     &           + MASS(IBUFF, 2) + MASS(IBUFF, 3)
     &           + MASS(IBUFF, 4)
C
           SUMSB = MASS(IBUFF, 5) + MASS(IBUFF, 6)
     &           + MASS(IBUFF, 7)
C
           SUMSM = MASS(IBUFF, 8) + MASS(IBUFF, 9)
     &           + MASS(IBUFF,10)
C
           SUMST = MASS(IBUFF,11) + MASS(IBUFF,12)
     &           + MASS(IBUFF,13)
C
           SUML1 = MASS(IBUFF,14) + MASS(IBUFF,15)
     &           + MASS(IBUFF,16)
C
           SUML2 = MASS(IBUFF,17) + MASS(IBUFF,18)
     &           + MASS(IBUFF,19)
C
           SUML3 = MASS(IBUFF,20) + MASS(IBUFF,21)
     &           + MASS(IBUFF,22)

           TOTAL = SUMR+SUMSB+SUMSM+SUMST+SUML1+SUML2+SUML3
C
C-------SKIP TO A NEW PAGE FOR EVERY 45 LINES PRINTED
C
           LINE = LINE + 1
           IF ( LINE .NE. 1 .AND. MOD(LINE-1,45) .EQ. 0 ) THEN
                NPAGE = NPAGE + 1
                CALL GMASSHD
           ENDIF
C
C---PRINT OUT GROUPED MASS REPORT TO USER SPECIFIED OUTPUT FILENAME.
C
```

```
            IF (NPLEFT .NE. NPLFT(IBUFF)) THEN
                 NPLEFT =   NPLFT(IBUFF)
                 WRITE(LUPRN,2015) NPLEFT
                 LINE = LINE + 3
            ENDIF
            WRITE(LUPRN,2000) TIMES(IBUFF), SUMR, SUMSB, SUMSM,
     &          SUMST, SUML1, SUML2, SUML3, TOTAL
C
C---PRINT OUT GROUPED MASS RESULTS TO FILE 'xxxxx.GRP'
C
            WRITE(LUGMO,2000) TIMES(IBUFF), SUMR, SUMSB, SUMSM,
     &          SUMST, SUML1, SUML2, SUML3, TOTAL
   50 CONTINUE
      IF (.NOT. WIDE .AND. WIDSET) THEN
            IF (LASJET) THEN
                 WRITE(LUPRN,'(1X,A)') CHAR(27) // '(s10H'
            ELSE
                 CALL SETPR('OFF', LUPRN)
            ENDIF
            WIDSET = .FALSE.
      ENDIF
      RETURN
 2000 FORMAT(T7, F7.2, 7X, 1P, 7(1X,E10.3),2X,E10.3)
 2015 FORMAT(/,T8,'Number of plants is now: ',I2,/)
      END
*OUTLUMPS
C
C*--------------------------------------------------------------------*
C*          S U B R O U T I N E      O U T L U M P S                  *
C*    THIS SUBROUTINE CREATES THE LUMPED MASS REPORT, WHICH LISTS     *
C*    THE SUM OF THE SOLUTE MASSES FOR THE ROOT COMPARTMENTS; THE SUM  *
C*    OF THE SOLUTE MASSES FOR ALL STEM COMPARTMENTS; THE SUM OF THE   *
C*    SOLUTE MASSES FOR ALL LEAF COMPARTMENTS; AND THE SUM OF THE      *
C*    SOLUTE MASSES FOR THE TOTAL PLANT.                               *
C*    ARE PREDEFINED IN THE MODEL.                                     *
C*                                                                     *
C*    LOCAL VARIABLE DEFINITIONS                                       *
C*    --------------------------                                       *
C*     SOIL  = SOIL COMPARTMENT                                        *
C*     SUMR  = ROOT SUM                                                *
C*     SUMS  = STEM SUM                                                *
C*     SUML  = LEAF SUM                                                *
C*     TOTAL = TOTAL PLANT MASS                                        *
C*--------------------------------------------------------------------*
      SUBROUTINE OUTLUMPS
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      DOUBLE PRECISION SOIL, SUMR, SUMS, SUML, TOTAL
```

```
      INTEGER   I, LINE, NPLEFT
C
      NPAGE = 1
      CALL LMASSHD
      REWIND(LUDAC)
      IBUFF = NHOLD
      NPLEFT = PSTART
      LINE  = 0
      DO 50, I = 1, ITIME
C
C  Fill the Cache Buffer as necessary...
C
           IF ( IBUFF .GE. NHOLD ) THEN
                CALL BUFIN(NHOLD)
                IBUFF = 0
           ENDIF
           IBUFF = IBUFF + 1
C
C  Do sums...
C
           SOIL  = MASS(IBUFF,-1)
C
           SUMR  = MASS(IBUFF, 0) + MASS(IBUFF, 1)
     &           + MASS(IBUFF, 2) + MASS(IBUFF, 3)
     &           + MASS(IBUFF, 4)
C
           SUMS  = MASS(IBUFF, 5) + MASS(IBUFF, 6)
     &           + MASS(IBUFF, 7)
     &           + MASS(IBUFF, 8) + MASS(IBUFF, 9)
     &           + MASS(IBUFF,10)
     &           + MASS(IBUFF,11) + MASS(IBUFF,12)
     &           + MASS(IBUFF,13)
C
           SUML  = MASS(IBUFF,14) + MASS(IBUFF,15)
     &           + MASS(IBUFF,16)
     &           + MASS(IBUFF,17) + MASS(IBUFF,18)
     &           + MASS(IBUFF,19)
     &           + MASS(IBUFF,20) + MASS(IBUFF,21)
     &           + MASS(IBUFF,22)

           TOTAL = SUMR + SUMS + SUML
C
C-------SKIP TO A NEW PAGE FOR EVERY 45 LINES PRINTED
C
           LINE = LINE + 1
           IF ( LINE .NE. 1 .AND. MOD(LINE-1,45) .EQ. 0 ) THEN
                NPAGE = NPAGE + 1
                CALL LMASSHD
```

```
            ENDIF
C
C---PRINT LUMPED MASS REPORT TO USER SPECIFIED OUTPUT FILENAME.
C
            IF (NPLEFT .NE. NPLFT(IBUFF)) THEN
                 NPLEFT = NPLFT(IBUFF)
                 WRITE(LUPRN,2015) NPLEFT
                 LINE = LINE + 3
            ENDIF
            WRITE(LUPRN,2000)
     &         TIMES(IBUFF), SOIL, SUMR, SUMS, SUML, TOTAL
C
C---PRINT OUT LUMPED MASS RESULTS TO FILE 'xxxxx.GRP'
C
            WRITE(LULMO,2000)
     &         TIMES(IBUFF), SOIL, SUMR, SUMS, SUML, TOTAL
   50 CONTINUE
      IF (WIDE .AND. .NOT. WIDSET) THEN
            IF (LASJET) THEN
                 WRITE(LUPRN,'(1X,A)') CHAR(27) // '(s16.66H'
            ELSE
                 CALL SETPR('ON ', LUPRN)
            ENDIF
            WIDSET = .TRUE.
      ENDIF
      RETURN
 2000 FORMAT(T7, F7.2, 4X, 1P, 7(1X,E10.3),2X,E10.3)
 2015 FORMAT(/,T8,'Number of plants is now: ',I2,/)
      END
*MASSHD
C*--------------------------------------------------------------------*
C*              S U B R O U T I N E     M A S S H D                   *
C*    THIS SUBROUTINE PRINTS OUT A HEADING AND IS CALLED FROM PROUT.  *
C*--------------------------------------------------------------------*
      SUBROUTINE MASSHD(INDEX,NN,JJ,JJJ)
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER   NN, JJ, JJJ, I, INDEX
      CHARACTER*7 LONG(2), TITLE(6)*40
      DATA LONG/'       ','(cont.)'/
      DATA TITLE /'  MASS IN COMPARIMENT',
     &            5*' '/
C
C-----OUTPUT FORMATS
C
 2000 FORMAT(T8,'Number of Plants at Start: ',I3,T60,'Page ',I4,1X,A7)
 2010 FORMAT(T8,'TIME',6X,10(1X,I5,5X))
C 2020 FORMAT(T8,'TIME',6X, 2(1X,I5,5X),2X,'STORED')
```

```
 2050 FORMAT(/,T40,A,/)
C
C-----PRINT MASS OUTPUT HEADING.
      CALL PRHEAD
      WRITE(LUPRN,2000) PSTART, NPAGE, LONG(NN)
      WRITE(LUPRN,2050) TITLE(INDEX)
      WRITE(LUPRN,2010) (I, I = JJ, JJJ)
      RETURN
      END
*GMASSHD
C
C*-------------------------------------------------------------------------*
C*          S U B R O U T I N E       G M A S S H D                        *
C*    THIS SUBROUTINE PRINTS A HEADING AND IS CALLED BY SUBROUTINE         *
C*    OUTGROUP.                                                            *
C*-------------------------------------------------------------------------*
C
      SUBROUTINE GMASSHD
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
C
C-----OUTPUT FORMAT
 2000 FORMAT(A,/,T8,A,/,T8,'This Filename is -->',A,/)
 2010 FORMAT(T8,'Date : ',A,2X,A,4X,'Number of Plants at Start: ',I3,:,
     &       T67,'Page ',I4,/)
 2020 FORMAT(T35,'GROUPED     MASS',/,
     &       T103,'TOTAL',/,T8,'TIME',
     &       13X,'Roots   +  Stem B  +  Stem M  +  Stem T  +  Leaf 1 '
     &                   ' +  Leaf 2  +  Leaf 3  =   PLANT')
 2030 FORMAT(T8,'Number of Plants at Start: ',I3,:,T67,'Page ',I4,/)
C
C-----PRINT GROUPED MASS OUTPUT HEADING TO FILE 'xxxxx.GRP'
C
Cold  WRITE(LUGMO,2000) ' '    , PGHD, OUTFILE
Cold  WRITE(LUGMO,2010) DATE, TIME, PSTART
Cold  WRITE(LUGMO,2020)
C
C-----PRINT GROUPED MASS OUTPUT HEADING ON "PRINTER" OUTPUT.
      IF (.NOT. WIDSET) THEN
           IF (LASJET) THEN
                WRITE(LUPRN,'(1X,A)') CHAR(27) // '(s16.66H'
           ELSE
                CALL SETPR('ON ', LUPRN)
           ENDIF
           WIDSET = .TRUE.
      ENDIF
      CALL PRHEAD
      WRITE(LUPRN,2030) PSTART, NPAGE
```

```
      WRITE(LUPRN,2020)
      RETURN
      END
*LMASSHD
C
C*-------------------------------------------------------------------*
C*          S U B R O U T I N E      L M A S S H D                   *
C*    THIS SUBROUTINE PRINTS OUT A HEADING AND IS CALLED BY SUBROUTINE *
C*    OUTLUMPS.                                                      *
C*-------------------------------------------------------------------*
C
      SUBROUTINE LMASSHD
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
C
C-----OUTPUT FORMATS
C 2000 FORMAT(A,/,T8,A,/,T8,'This Filename is -->',A,/)
C 2010 FORMAT(T8,'Date : ',A,2X,A,4X,'Number of Plants at Start: ',
C     &       I3,:,T67,'Page ',I4,/)
 2020 FORMAT(T35,'LUMPED      MASS',/,
     &       T66,'TOTAL',/,T8,'TIME',
     &       10X,'[Soil]     Roots   +  Stem    +  Leaf    '
     &                    ' =  PLANT')
 2030 FORMAT(T8,'Number of Plants at Start: ',I3,:,T67,'Page ',I4,/)
C
C-----PRINT LUMPED MASS OUTPUT HEADING TO FILE 'xxxxx.GRP'
C
Cold  WRITE(LULMO,2000) ' '     , PGHD, OUTFILE
Cold  WRITE(LULMO,2010) DATE, TIME, PSTART
Cold  WRITE(LULMO,2020)
C
C-----PRINT LUMPED MASS OUTPUT HEADING ON "PRINTER" OUTPUT.
      IF (WIDSET) THEN
           IF (LASJET) THEN
                WRITE(LUPRN,'(1X,A)') CHAR(27) // '(s10H'
           ELSE
                CALL SETPR('OFF', LUPRN)
           ENDIF
           WIDSET = .FALSE.
      ENDIF
      CALL PRHEAD
      WRITE(LUPRN,2030) PSTART, NPAGE
      WRITE(LUPRN,2020)
      RETURN
      END
*PRHEAD
C
C*-------------------------------------------------------------------*
```

```
C*          S U B R O U T I N E       P R H E A D                    *
C*   THIS SUBROUTINE PRINTS OUT THE PAGE HEADING, BOTH FOR DTADRVR,    *
C*   AND FOR THE OUTPUT SUBROUTINES IN "PRINT46.FOR".                  *
C*   LOCAL VAR 'NEWPAGE' IS INITIALLY BLANK, THEN CHANGED TO FORM-FEED *
C*   AFTER FIRST CALL; THUS PRHEAD DOES NOT EJECT PAGE ON FIRST CALL.  *
C*--------------------------------------------------------------------*
C
      SUBROUTINE PRHEAD
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      CHARACTER*1 NEWPAGE
      SAVE NEWPAGE
      DATA NEWPAGE/' '/
C
C-----OUTPUT FORMATS
 3010 FORMAT(A,/,T8,A,/,T8,'Filename is:  ',A,T80,
     &       'Date:  ',A,2X,A,/)
 3020 FORMAT(A,/,T5,A,/,T5,'Filename is:  ',A,T52,
     &       'Date:  ',A,2X,A,/)
 3030 FORMAT(T8,'Start time: ', F7.2,'; Stop time: ', F7.2,
     &  '; Print interval: ', F6.3,/)
 3040 FORMAT(T5,'Start time: ', F7.2,'; Stop time: ', F7.2,
     &  '; Print interval: ', F6.3,/)
C
C-----PRINT PAGE HEADING.
C  [DT is printed instead of PRINTVL, which has beed adjusted down.]
C
      IF ( WIDSET ) THEN
        WRITE(LUPRN,3010) NEWPAGE, PGHD, OUTFILE, DATE, TIME
        WRITE(LUPRN,3030) TINITIAL, TSTOP, DT
      ELSE
        WRITE(LUPRN,3020) NEWPAGE, PGHD, OUTFILE, DATE, TIME
        WRITE(LUPRN,3040) TINITIAL, TSTOP, DT
      ENDIF
      NEWPAGE = CHAR(12)
      RETURN
      END
```

```
*FILE: CACHE46.FOR                          Last revision:  November 14, 1989
C
      SUBROUTINE BUFIN( LEN )
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER   LEN
*************************************************************************
*  Reads in a block from the Cache Buffer.  Intended that LEN should   *
*  be in full NHOLD chunks.  These routines were added so that the     *
*  internal data storage (set by NHOLD) could be kept reasonable.      *
*                          David E. Cawlfield,  Winter '88              *
*************************************************************************
      IF ( LEN .NE. NHOLD ) THEN
           STOP 1024
      ENDIF
      CALL CURSOR(21,20)
      WRITE(*,2000)
      READ (LUDAC) NPLFT
      READ (LUDAC) MASS
      READ (LUDAC) TIMES
      CALL CURSOR(21,20)
      WRITE(*,2010)
      RETURN
 2000 FORMAT('Reading Data Cache Buffer...',$)
 2010 FORMAT('                            ',$)
      END
*BUFOUT
      SUBROUTINE BUFOUT( LEN )
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      INTEGER   LEN, IZERO, IC
*************************************************************************
*  Writes in a block to the Cache Buffer.  Intended that LEN should    *
*  be in full NHOLD chunks.                                            *
*                          David E. Cawlfield,  Winter '88              *
*************************************************************************
C
C  Two special cases may occur on the last call to BUFOUT:
C  1)  If LEN (IBUFF) is zero, beat a hasty exit.
C  2)  If LEN (IBUFF) is < NHOLD, zero out remaining locations.
C
      IF ( LEN .EQ. 0 ) RETURN
      IF ( LEN .LT. NHOLD ) THEN
           CALL CURSOR(21,20)
           WRITE(*,2020)
           DO  10, IZERO = NHOLD, LEN+1, -1
                NPLFT(IZERO) = 0
                TIMES(IZERO) = 0.D0
```

```
                DO  20, IC = NCL, NCU
                     MASS(IZERO,IC) = 0.D0
   20           CONTINUE
   10      CONTINUE
      ENDIF
C
C  Now, write the Buffer...
C
      CALL CURSOR(21,20)
      WRITE(*,2000)
      WRITE (LUDAC) NPLFT
      WRITE (LUDAC) MASS
      WRITE (LUDAC) TIMES
      CALL CURSOR(21,20)
      WRITE(*,2010)
      RETURN
 2000 FORMAT('Writing Data Cache Buffer...',$)
 2010 FORMAT('                            ',$)
 2020 FORMAT('Zeroing Buffer tail...      ',$)
      END
```

```
*LSTID      FILE: BLOCK46.FOR          Last Revision:  June 14, 1989
C
C*--------------------------------------------------------------------*
C* THIS IS A BLOCK DATA WHICH IS USED TO STORE A HEADER FOR THE OUTPUT *
C* LISTINGS.
C*--------------------------------------------------------------------*
      BLOCK DATA LSTID
      CHARACTER*20  INFILE, OUTFILE, PGHD*70, DATE*8, TIME*8,
     &              WORDS(2)*72
      COMMON /BLK7/ INFILE, OUTFILE, PGHD,    DATE,   TIME, WORDS
      DATA PGHD/'Plant Transport Three Leaf Model (Ver 4.6).      All ri
     &ghts reserved.  '/
      END
*SCRNHD
C
C*--------------------------------------------------------------------*
C*            S U B R O U T I N E    S C R N H D                      *
C*    THIS SUBROUTINE PRINTS THE SCREEN HEADER AND IS CALLED          *
C*    BY MOST DISPLAY ROUTINES.                                       *
C*--------------------------------------------------------------------*
C
      SUBROUTINE SCRNHD
      WRITE(*,2000)
 2000 FORMAT(T10,'▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒',/
     &       T10,'▒▒▒  PLANT TRANSPORT THREE LEAF MODEL (VER 4.6)  ▒▒▒',/
     &       T10,'▒▒▒              ALL RIGHTS RESERVED.            ▒▒▒',/
     &       T10,'▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒')
      RETURN
      END
*BYE
C
C*--------------------------------------------------------------------*
C*              S U B R O U T I N E      B Y E                        *
C*    THIS SUBROUTINE PRINTS MESSAGE TO THE CONSOLE BEFORE LEAVING THE *
C*    UTAB SYSTEM;  THE MESSAGE RE-STATES THE NAME OF THE OUTPUT FILE. *
C*    THIS SUBROUTINE IS CALLED FROM THE MAIN PROGRAM WHICH IS IN      *
C*    FILE "PLANT46.FOR".                                             *
C*--------------------------------------------------------------------*
      SUBROUTINE BYE
$INCLUDE:'COMSIZE.I46'
$INCLUDE:'COMBLOC.I46'
      WRITE(*,2000) OUTFILE
 2000 FORMAT(5(/),
     & T10,' You are now leaving program PLANT46 [version 4.6]',
     & 3(/),T10,' You can retrieve your output from filename --> ',A,/,
     & T10,' or printer if you chose PRN: as output device.',///)
      RETURN
      END
```

```
* FILE: COMSIZE.I46                    Last revision:  January 12, 1989
C
C     NCL    = Number of compartments, lower index
C     NCU    = Number of compartments, upper index
C     NCPMT  = Number of compartments
C     NWL    = Number of walls, lower index
C     NWU    = Number of walls, upper index
C     NWALLS = Number of walls in total compartments
C     NQSTO  = Number of flow vectors between xylem (or phloem) lumen
C              and the storage compartments
C     NHOLD  = Number of values held for data caching, (for now)
C
      INTEGER    NCL, NCU, NCPMT, NQSTO, NWL, NWU, NWALLS, NHOLD
      PARAMETER (NCL = -1, NCU = 22, NCPMT = NCU-NCL+1,
     &           NWL =  0, NWU = 20, NWALLS= NWU-NWL+1,
     &           NQSTO=14, NHOLD=50)
C
C  Following are Logical Units used for files...as parameters they take
C  no space unless used, and cannot be accidentally altered.
C
      INTEGER  LUDAI, LUDAO, LUPRN, LUFMO, LULMO, LUGMO, LUDAC,
     &         LUXPI, LUXPO
      PARAMETER (LUDAI= 5, LUDAO= 7, LUPRN= 6, LUFMO= 8, LULMO= 9,
     &           LUGMO=10, LUDAC=50, LUXPI=30, LUXPO=31)


* FILE: COMBLOC.I46                    Last revision: January 26, 1989
C
      DOUBLE PRECISION  DIFU,           SIGMA
      COMMON /BLK1/     DIFU(NWL:NWU), SIGMA(NWL:NWU)
C
      DOUBLE PRECISION    DELTAX, LAMBDA, VOL, SB, QSTF, QSTB
      COMMON /BLK2/ DELTAX(NWL:NWU), LAMBDA(NCL:NCU), VOL(NCL:NCU),
     &   SB(NCL:NCU), QSTF(NWL:NQSTO), QSTB(NWL:NQSTO)
C
      INTEGER             NPAGE, ITIME, IBUFF
      LOGICAL             WIDE, WIDSET, LASJET, MDATA
      DOUBLE PRECISION    PRINTVL, TINITIAL, TSTOP, DT, TPRT, TOL
      COMMON /BLK3/ PRINTVL, TINITIAL, TSTOP, DT, NPAGE, ITIME, IBUFF,
     &  TPRT, TOL, WIDE, WIDSET, LASJET, MDATA
C
      INTEGER           NPLFT
      DOUBLE PRECISION  MASS, TIMES
      COMMON /BLK4/     NPLFT(NHOLD), MASS(NHOLD,NCL:NCU), TIMES(NHOLD)
C
      DOUBLE PRECISION  XNOW,           XOLD,           FRAC,    TR
      COMMON /BLK5/     XNOW(NCL:NCU), XOLD(NCL:NCU), FRAC(3), TR(3)
```

```
C
      DOUBLE PRECISION  A
      COMMON /BLK6/     A(NCL:NCU,NCL:NCU)
C
      CHARACTER*20  INFILE, OUTFILE, PGHD*70, DATE*8, TIME*8,
     &              WORDS(2)*72
      COMMON /BLK7/ INFILE, OUTFILE, PGHD,    DATE,   TIME, WORDS
C
      DOUBLE PRECISION    DCH, HC, CAIR, SOURC
      INTEGER             PLEFT, PSTART
      COMMON /BLK8/       DCH, HC, CAIR, SOURC(NCL:NCU), PLEFT, PSTART
C
      DOUBLE PRECISION  AREA,          Q
      COMMON /BOTH/     AREA(NWL:NWU), Q(NWU)
```

```
*UTIL1.FOR  Fortran Utilities (1)       Last Revision:  November 16, 1989
C                                       Source File:    UTIL1.FOR
*************************************************************************
*  A collection of generalized, hopefully useful, Fortran utilities.    *
*  They perform operations like clearing the screen, positioning        *
*  the cursor, and the like.  For the most general use, these should    *
*  be put into a library (.LIB) format.  Librarys only load the         *
*  routines which are actually referenced by the other object files     *
*  (rather than loading all object modules, whether used or not).       *
*     *NOTE* -- The cursor-type routines are updated versions of        *
*  those which appeared in earlier PLANTxx programs.  [They were up-    *
*  dated to conform more to ANSI.SYS]  If earlier PLANTxx programs are  *
*  re-linked, remove the cursor, setpr, etc, routines from PLANTxx.FOR;*
*  otherwise the linker will complain about multiple entry points.      *
*     See the individual routines for the calling parameters:           *
*                                                                       *
*-ANSI Routines ---------------------------------------------------------*
*     DAYSTR     --   Returns system date as an 8-character string.     *
*     TIMSTR     --   Returns system time as an 8-character string,     *
*                     based upon 12 hour clock, eg:  '11:56 pm'         *
*     CLS        --   Clears the screen through ANSI, replacement for   *
*                     CLRSCRN.                                          *
*     CURSOR     --   Positions the cursor to IRow, JColumn.  By trad-  *
*                     ition, Row is first.                              *
*     SETPR      --   Toggles the printer compressed mode.              *
*     SETLJ      --   Toggles the LaserJet compressed mode.             *
*-String Handling Routines ----------------------------------------------*
*     LENSTR     --   Returns the "actual" length of a character,      *
*                     ignoring trailing blanks.  [The standard LEN      *
*                     function returns the *defined* length.]           *
*     TOLOWER    --   Converts a single character to lower case.        *
*     TOUPPER    --   Converts a single character to upper case.        *
*     STRDN      --   Converts a character string to lower case.        *
*     STRUP      --   Converts a character string to upper case.        *
*-Precision Control -----------------------------------------------------*
*     ROUNDZ     --   Sets rounding control in the 8087, which is       *
*                     useful for error bounding.  Modes are selected by *
*                     an integer argument in {-2, -1, 0, +1, +2} for    *
*                     {none, down, nearest even, up, restore default}.  *
*                     The hardware default sets the '87 to round to     *
*                     nearest even when the machine is booted.          *
*     SECOND     --   Real function, returning elapsed time in seconds  *
*                     since midnight.                                   *
*-----------------------------------------------------------------------*
*                          David E. Cawlfield,  Spring '88               *
*************************************************************************
*DAYSTR                                        Last Update:  March 31, 1988
      SUBROUTINE DAYSTR(DAYT)
```

```
      CHARACTER        DAYT*8
      INTEGER*2        IYR, IMO, IDA
      INTEGER   MOD
      INTRINSIC MOD
      EXTERNAL  GETDAT
*****************************************************************************
*  Returns the system date, as a character string.  This routine is         *
*  very system dependent; furthermore the GETDAT call is specific to        *
*  MS Fortran 4.x, so this may have to be re-coded for other systems        *
*  or compilers.  If trouble is encountered, simply comment out the         *
*  call to this routine--the user may/should enter the time anyway.         *
*                          David E. Cawlfield,  Spring '88                  *
*****************************************************************************
      CALL GETDAT(IYR, IMO, IDA)
      IYR = MOD(IYR, 1900)
      WRITE(DAYT, 2000)  IMO, IDA, IYR
      RETURN
 2000 FORMAT(I2.2, '/', I2.2, '/', I2.2)
      END
*TIMSTR                                    Last Revision:  October 20, 1988
C
      SUBROUTINE TIMSTR(TIME)
      CHARACTER  TIME*8
      INTEGER*2 IHR, IMIN, ISEC, IHDT
      CHARACTER  CHR(0:23)*2, CAM(0:23)*2, CMIN*2
*****************************************************************************
*  Returns the system time as a character string.  This routine is          *
*  very system dependent; furthermore, the GETTIM call is specific to       *
*  MS Fortran 4.x, so this may have to be re-coded for other systems        *
*  or compilers.  If trouble is encountered, simply comment out the         *
*  call to this routine--the user may/should enter the time anyway.         *
*  A simple table look-up seems to be the best route to go on this.         *
*                          David E. Cawlfield,  Fall '88                    *
*****************************************************************************
      DATA CHR/
     &  '12','01','02','03','04','05','06','07','08','09','10','11',
     &  '12','01','02','03','04','05','06','07','08','09','10','11'/
      DATA CAM/ 12*'AM', 12*'PM'/
      CALL GETTIM(IHR, IMIN, ISEC, IHDT)
      WRITE(CMIN,'(I2.2)') IMIN
      TIME = CHR(IHR) // ':' // CMIN // ' ' // CAM(IHR)
      RETURN
      END
*CLS                                       Last Revision:  March 31, 1988
      SUBROUTINE CLS
      CHARACTER   ESC,  CHAR
      PARAMETER ( ESC = CHAR(27) )
*****************************************************************************
```

```
*  Clears the screen and homes the cursor by issuing an "ESC [2J"      *
*  sequence.  ANSI.SYS must be installed.                              *
************************************************************************
      WRITE(*, '(1X,A)') ESC // '[2J'
      RETURN
      END
*CURSOR                                        Last Revision:  April 1, 1988
      SUBROUTINE CURSOR(IR, JC)
      INTEGER   IR, JC
      CHARACTER   ESC,  CHAR, ESCSEQ*7
      PARAMETER ( ESC = CHAR(27) )
************************************************************************
*  Positions the cursor, using the ANSI "ESC [#;#f" sequence.          *
*  Writes positions IRow and JCol into the output string (ESCSEQ)      *
************************************************************************
      DATA ESCSEQ/ '[01;01f' /
      IF ( IR .LE. 0 .OR. IR .GT. 24 ) RETURN
      IF ( JC .LE. 0 .OR. JC .GT. 80 ) RETURN
C
C  The single, somewhat cryptic, WRITE statement below does the same as:
C           WRITE(ESCSEQ(2:3), '(I2.2)' ) IR
C           WRITE(ESCSEQ(5:6), '(I2.2)' ) JC
      WRITE(ESCSEQ, 2000) IR, JC
      WRITE(*, '(1X,A,\)') ESC // ESCSEQ
      RETURN
 2000 FORMAT('[', I2.2, ';', I2.2, 'f')
      END
*LENSTR                                        Last Revision:  April 2, 1988
      INTEGER FUNCTION LENSTR( STRING )
      CHARACTER STRING*(*), BLANK
      PARAMETER ( BLANK = ' ' )
      INTEGER   IC, L72, LEN
      INTRINSIC LEN
************************************************************************
*  Return the "actual" length of a character string, ignoring trailing *
*  blanks.  Uses intrinsic LEN to determine the *defined* length of    *
*  passed string; then marches backwards until the first non-blank     *
*  character is found.  If the string is empty, the value of IC will   *
*  "fall through" the loop and return a string length of zero.         *
************************************************************************
      L72 = LEN(STRING)
      DO  10, IC = L72, 1, -1
          IF ( STRING(IC:IC) .NE. BLANK ) GO TO 20
   10 CONTINUE
   20 LENSTR = IC
      END
*SETPR                                         Last Revision:  March 31, 1988
      SUBROUTINE SETPR(CMPRES, LUN)
```

```
      CHARACTER       CMPRES*3, CHAR
      INTEGER   LUN
      INTRINSIC CHAR
*************************************************************************
*  Issues either a Shift-In to put most ASCII printers into compressed *
*  mode; or a Shift-Out to put them back into standard mode.  SI and   *
*  SO are among the few codes which most printers agree upon, but      *
*  that still does not guarantee that this will work on all models.    *
*  Issues a question-mark if CMPRES is neither 'on ' or 'off'.         *
*                              D. E. Cawlfield,  Winter '88.           *
*************************************************************************
      IF      (CMPRES .EQ. 'ON ' .OR. CMPRES .EQ. 'on ') THEN
           WRITE(LUN,10) CHAR(15)
      ELSE IF (CMPRES .EQ. 'OFF' .OR. CMPRES .EQ. 'off') THEN
           WRITE(LUN,10) CHAR(18)
      ELSE
           WRITE(LUN,10) CHAR(63)
      ENDIF
   10 FORMAT(1X,A)
      RETURN
      END
*SETLJ                                        Last Revision:  April 17, 1989
      SUBROUTINE SETLJ(CMPRES, LUN)
      CHARACTER       CMPRES*3, CHAR, ESC
      PARAMETER ( ESC = CHAR(27) )
      INTEGER   LUN
      INTRINSIC CHAR
*************************************************************************
*  Issues a sequence which puts HP LaserJet (II's and +'s) into        *
*  compressed mode or back to "normal" (16.66 and 10 cpi).             *
*                              D. E. Cawlfield,  Spring '89.           *
*************************************************************************
      IF      (CMPRES .EQ. 'ON ' .OR. CMPRES .EQ. 'on ') THEN
           WRITE(LUN,10) ESC // '(s16.66H'
      ELSE IF (CMPRES .EQ. 'OFF' .OR. CMPRES .EQ. 'off') THEN
           WRITE(LUN,10) ESC // '(s10H'
      ELSE
           WRITE(LUN,10) CHAR(63)
      ENDIF
   10 FORMAT(A)
      RETURN
      END
*TOLOWER                                      Last Revision:  April 1, 1988
      CHARACTER FUNCTION TOLOWER( KHAR )
      CHARACTER KHAR, CHAR
      INTEGER*1 IBSET, ICHAR
      LOGICAL   LLT, LGT
*************************************************************************
```

```
*  Converts a single character to lower case by ORing it with          *
*  00100000B, which sets bit 5.  Does nothing if not passed an upper   *
*  case character.  Setting bit 5 [right from 0] does not alter lower  *
*  case letters, but it produces strange effects with non-alphabetic   *
*  characters.                                                         *
************************************************************************
      TOLOWER = KHAR
      IF ( LLT(KHAR,'A') .OR. LGT(KHAR,'Z') ) RETURN
      TOLOWER = CHAR( IBSET(ICHAR(KHAR), 5) )
      RETURN
      END
*TOUPPER                                        Last Revision:  April 1, 1988
      CHARACTER FUNCTION TOUPPER( KHAR )
      CHARACTER KHAR, CHAR
      INTEGER*1 IBCLR, ICHAR
      LOGICAL   LLT, LGT
************************************************************************
*  Converts a single character to upper case by "masking" it with      *
*  11011111B.  Does nothing if not passed a lower case character.      *
*  As is the case in TOLOWER, we want to avoid setting bit 5 in non-   *
*  alphabetic characters.                                              *
************************************************************************
      TOUPPER = KHAR
      IF ( LLT(KHAR,'a') .OR. LGT(KHAR,'z') ) RETURN
      TOUPPER = CHAR( IBCLR(ICHAR(KHAR), 5) )
      RETURN
      END
*STRDN                                          Last Revision:  April 5, 1988
      SUBROUTINE STRDN( STRING )
      CHARACTER         STRING*(*), TOLOWER
      INTEGER   L72, LENSTR, IC
      EXTERNAL  LENSTR, TOUPPER
************************************************************************
*  Converts a character string argument to all lower case by first     *
*  getting the "true" length via LENSTR and then doing the conversion  *
*  via TOLOWER.                                                        *
************************************************************************
      L72 = LENSTR(STRING)
      DO  10, IC = L72, 1, -1
          STRING(IC:IC) = TOLOWER(STRING(IC:IC))
   10 CONTINUE
      RETURN
      END
*STRUP                                          Last Revision:  April 5, 1988
      SUBROUTINE STRUP( STRING )
      CHARACTER         STRING*(*), TOUPPER
      INTEGER   L72, LENSTR, IC
      EXTERNAL  LENSTR, TOUPPER
```

```
************************************************************************
*  Converts character string argument to all upper case by first       *
*  getting the "true" length via LENSTR and then doing the conversion  *
*  via TOUPPER.                                                        *
************************************************************************
      L72 = LENSTR(STRING)
      DO  10, IC = L72, 1, -1
          STRING(IC:IC) = TOUPPER(STRING(IC:IC))
   10 CONTINUE
      RETURN
      END
*ROUNDZ                                    Last Revision:  April 8, 1988
      SUBROUTINE ROUNDZ( IARG )
      INTEGER   IARG
      INTEGER*2 CW, SCWRQQ, CWSAVE, IBSET, IAND
      LOGICAL   FIRST
      EXTERNAL  SCWRQQ, LCWRQQ
************************************************************************
*  Routine to change the method of rounding used by the 8087.  The     *
*  8087 has several modes available; the default is known as "unbiased *
*  round towards nearest even" [Palmer, "8087 Primer", 1984].  Integer *
*  argument IARG is used to select. IARG --> {-2, -1, 0, +1, +2} for:  *
*  {truncate, round down, default, round up, reset to default}.        *
*  The machine is always set to the default (round to nearest even)    *
*  at boot time, so modes 0 and 2 are redundant.                       *
*  In the 8087, the rounding usually takes place at the 64 bit         *
*  signifigand, although this, too, may be changed by the Control Word.*
*  Uses the MicroSoft SCWRQQ and LCWRQQ routines, which were provided  *
*  to modify the 8087 Control Word.  Also uses the Bit-Set routines.   *
************************************************************************
      DATA FIRST/ .TRUE. /
C
C  First, get original Control Word and save it...
C
      PRINT *
      PRINT *, 'Mode ', IARG
      IF ( FIRST ) THEN
           CWSAVE = SCWRQQ()
           FIRST  = .FALSE.
           WRITE(*,'(1X,A,Z4)') 'The original CW is: ', CWSAVE
      ENDIF
C
C  Now, select the option, and re-mask the CW...
C
      IF       (IARG .EQ. -2) THEN
           CW = SCWRQQ()
           CW = IAND( CW, 16#F3FF)
           CW = IBSET(CW,10)
```

```
            CW = IBSET(CW,11)
         ELSE IF (IARG .EQ. -1) THEN
            CW = SCWRQQ()
            CW = IAND( CW, 16#F3FF)
            CW = IBSET(CW,10)
         ELSE IF (IARG .EQ.  0) THEN
            CW = SCWRQQ()
            CW = IAND( CW, 16#F3FF)
         ELSE IF (IARG .EQ. +1) THEN
            CW = SCWRQQ()
            CW = IAND( CW, 16#F3FF)
            CW = IBSET(CW, 11)
         ELSE IF (IARG .EQ. +2) THEN
            CW = CWSAVE
         ELSE
            PRINT *,'Improper argument given to ROUNDZ -- ', IARG
            RETURN
         ENDIF
         CALL LCWRQQ(CW)
         WRITE(*,'(1X,A,Z4)') 'The altered  CW is: ', CW
         RETURN
         END
*SECOND    Time *function*          Last Revision:  July 15, 1989
C                                   Source File:  FSECOND.FOR
         REAL FUNCTION SECOND()
         INTEGER*2 IH, IM, IS, IHU
         REAL      START, DAY
         SAVE      START, DAY
************************************************************************
*  Returns the number of seconds and hundredths of seconds elapsed     *
*  since midnight.                      D. E. Cawlfield, July '89.      *
************************************************************************
         DATA START, DAY/ 2*0.0 /
         CALL GETTIM(IH, IM, IS, IHU)
         SECOND = 3600.*IH + 60.*IM + IS + 0.01*IHU
         IF (SECOND+DAY .LT. START) DAY = DAY + 86400.
         SECOND = SECOND + DAY
         START = SECOND
         RETURN
         END
```

```
*LSTVEC     Fortran Utilities (2)        Last Revision:  November 16, 1989
C                                        Source File:  UTIL2.FOR
      SUBROUTINE LSTVEC(LU, VECTOR, IB, IE, IM, IW, CNAME)
      INTEGER             LU, IB, IE, IM, IW
      DOUBLE PRECISION    VECTOR(IB:IE), TESTV
      CHARACTER*24   CFORM1, CFORM2, CFORM3, CFORM4*96, CNAME*(*)
      INTEGER   IBEG, IEND, ICOL, LEFT, LINES, LOOP, MCOL, I,
     &  LEN, LENOLD, LTAB, LMAX, LENSTR, IMOLD
      LOGICAL   SAME
*************************************************************************
*  LSTVEC prints a one dimensional array (vector) to the output unit.   *
*  The routine was made as general as possible so that it could be      *
*  used in a library of general support routines.                       *
*                                                                       *
*  Parameters:                                                          *
*     LU         --   Logical Unit number for report file.  Units 1    *
*                     and 6 are pre-connected in MS Fortran and do not  *
*                     need to be formally opened.                       *
*     VECTOR     --   DP vector array.  Dimensioned IB lower; IE upper. *
*     IB         --   Lower subscript dimension for VECTOR.             *
*     IE         --   Upper subscript dimension for VECTOR.             *
*     IW         --   Number of columns wide to print.  At 12 chars per *
*                     column, this would be 6 for normal print on 8x11; *
*                     10 for compressed print on 8x11; etc.             *
*     CNAME      --   Character string which is printed to identify     *
*                     VECTOR; usually the Fortran variable name.        *
*-----------------------------------------------------------------------*
*                          D. E. Cawlfield,  July '88                   *
*************************************************************************
      DATA LENOLD/0/, IMOLD/0/
      DATA CFORM1/ '(//, T08, A040)          '/
      DATA CFORM2/ '(/ , T08, 10(I8,4X) )    '/
      DATA CFORM3/ '(1P, T08, 10E12.3)       '/
      DATA CFORM4/ '(/,  T08,''...  All these values (in ['', I2, ''..'',
     & , I2, '']) were:  '', 1P, E12.3 ,''  ...'')     '/
C
      LEN = LENSTR(CNAME)
      IF (LEN .EQ. 0) GO TO 50
      IF (LEN .NE. LENOLD .OR. IM .NE. IMOLD) THEN
           LMAX = 12*IW + IM
           LTAB = (LMAX + LEN)/2
           LENOLD = LEN
           IMOLD  = IM
           WRITE(CFORM1(12:14), '(I3.3)') LTAB
           WRITE(CFORM1( 7: 8), '(I2.2)') IM
           WRITE(CFORM2( 7: 8), '(I2.2)') IM
           WRITE(CFORM3( 7: 8), '(I2.2)') IM
           WRITE(CFORM4( 7: 8), '(I2.2)') IM + 3
```

```
      ENDIF
      WRITE(LU,CFORM1) CNAME
C
C  See if VECTOR is constant and, if so, do a short form report.
C
   50 TESTV = VECTOR(IB)
      SAME  = .TRUE.
      DO  10, I = IB+1, IE
          IF (VECTOR(I) .NE. TESTV) THEN
              SAME = .FALSE.
              GO TO 20
          ENDIF
   10 CONTINUE
C
   20 IF ( SAME ) THEN
          WRITE(LU, CFORM4) IB, IE, TESTV
      ELSE
          MCOL = IE - IB + 1
          LOOP = MCOL / IW
          LEFT = MOD(MCOL, IW)
          IF (LEFT .NE. 0) LOOP = LOOP + 1
          IBEG = IB
          IEND = MIN(IBEG+IW-1, IE)
          DO  30, LINES = 1, LOOP
              WRITE(LU,CFORM2) (       ICOL,  ICOL = IBEG, IEND)
              WRITE(LU,CFORM3) (VECTOR(ICOL), ICOL = IBEG, IEND)
              IBEG = IEND + 1
              IEND = MIN(IBEG+IW-1, IE)
   30     CONTINUE
      ENDIF
      RETURN
      END
```

```
*UTIL3      Fortran Utilities (3)      Last Revision:  November 16, 1989
C                                      Source File:  UTIL3.FOR
*-------------------------------------------------------------------------*
*  A collection of routines to accept default values from the console.    *
*  Separate routines are written for each data class.  They are each      *
*  intended to obtain a *single* data item.  The usage is:                *
*     GET1xx(xxIN, xEFAULT, <alt-return>)                                 *
*  where the specific implementations are --                              *
*     GET1IN     --   Get one INTEGER; INTEGER parameters.                *
*     GET1SP     --   Get one single precision REAL; REAL parameters.     *
*     GET1DP     --   Get one double precision REAL, DP parameters.       *
*     GET1LO     --   Get one LOGICAL variable; LOGICAL parameters.       *
*                                                                         *
*  Although I dislike alternate returns, these routines were coded in     *
*  this fashion so that the calling program could control any error       *
*  messages to be generated.                                              *
*-------------------------------------------------------------------------*
*                          D. E. Cawlfield, Summer '89                    *
*                                                                         *
*-------------------------------------------------------------------------*
*GET1IN                                 Last Revision:  June 17, 1989
C                                      Source File:  UTIL3.FOR
      SUBROUTINE GET1IN(INTIN, IEFALT, *)
      INTEGER   INTIN, IEFALT, ITEM, IERR
      CHARACTER STRING*20
**************************************************************************
*  Get one integer from the console.  If the entry is comma or blank,    *
*  then the default value (IEFALT) is used.  The alternate return        *
*  allows the calling program to control any error processing.           *
**************************************************************************
  100 READ(*,1000,END=999) STRING
      IF (STRING .EQ. ' ' .OR. STRING(1:1) .EQ. ',') THEN
           INTIN = IEFALT
      ELSE
           READ(STRING,'(I20)',IOSTAT=IERR,END=999) ITEM
           IF (IERR .EQ. 0) THEN
                INTIN = ITEM
           ELSE
                WRITE(*,2000) STRING
                RETURN 1
           ENDIF
      ENDIF
      RETURN
  999 WRITE(*,2010)
      RETURN 1
 1000 FORMAT(A)
 2000 FORMAT(T5,'*Error* -- Illegal char in field: ', A)
 2010 FORMAT(T5,'*Error* -- End-of-File on Input.')
      END
```

```
*GET1SP                                 Last Revision:  June 17, 1989
C                                       Source File:  UTIL3.FOR
      SUBROUTINE GET1SP(SPIN, DEFALT, *)
      REAL      SPIN, DEFALT, TEMP
      INTEGER   IERR
      CHARACTER STRING*20
*************************************************************************
*  Get one SP REAL from the console.  If the entry is comma or blank,   *
*  then the default value (DEFALT) is used.  The alternate return       *
*  allows the calling program to control any error processing.          *
*************************************************************************
  100 READ(*,1000,END=999) STRING
      IF (STRING .EQ. ' ' .OR. STRING(1:1) .EQ. ',') THEN
          SPIN = DEFALT
      ELSE
          READ(STRING,'(F20.0)',IOSTAT=IERR,END=999) TEMP
          IF (IERR .EQ. 0) THEN
              SPIN = TEMP
          ELSE
              WRITE(*,2000) STRING
              RETURN 1
          ENDIF
      ENDIF
      RETURN
  999 WRITE(*,2010)
      RETURN 1
 1000 FORMAT(A)
 2000 FORMAT(T5,'*Error* -- Illegal char in field: ', A)
 2010 FORMAT(T5,'*Error* -- End-of-File on Input.')
      END
*GET1DP                                 Last Revision:  June 17, 1989
C                                       Source File:  UTIL3.FOR
      SUBROUTINE GET1DP(DPIN, DEFALT, *)
      DOUBLE PRECISION  DPIN, DEFALT, DTEMP
      INTEGER   IERR
      CHARACTER STRING*20
*************************************************************************
*  Get one DP REAL from the console.  If the entry is comma or blank,   *
*  then the default value (DEFALT) is used.  The alternate return       *
*  allows the calling program to control any error processing.          *
*************************************************************************
  100 READ(*,1000,END=999) STRING
      IF (STRING .EQ. ' ' .OR. STRING(1:1) .EQ. ',') THEN
          DPIN = DEFALT
      ELSE
          READ(STRING,'(F20.0)',IOSTAT=IERR,END=999) DTEMP
          IF (IERR .EQ. 0) THEN
              DPIN = DTEMP
```

```
          ELSE
               WRITE(*,2000) STRING
               RETURN 1
          ENDIF
     ENDIF
     RETURN
  999 WRITE(*,2010)
     RETURN 1
 1000 FORMAT(A)
 2000 FORMAT(T5,'*Error* -- Illegal char in field: ', A)
 2010 FORMAT(T5,'*Error* -- End-of-File on Input.')
     END
*GET1LO                                  Last Revision:  June 20, 1989
C                                        Source File:  UTIL3.FOR
     SUBROUTINE GET1LO(LOIN, LEFALT, *)
     LOGICAL          LOIN, LEFALT, LTEMP
     INTEGER   IERR
     CHARACTER STRING*20
*************************************************************************
*  Get one LOGICAL from the console.  If the entry is comma or blank,   *
*  then the default value (LEFALT) is used.  The alternate return       *
*  allows the calling program to control any error processing.          *
*************************************************************************
  100 READ(*,1000,END=999) STRING
     IF (STRING .EQ. ' ' .OR. STRING(1:1) .EQ. ',') THEN
          LOIN = LEFALT
     ELSE
          READ(STRING,'(L20)',IOSTAT=IERR,END=999) LTEMP
          IF (IERR .EQ. 0) THEN
               LOIN = LTEMP
          ELSE
               WRITE(*,2000) STRING
               RETURN 1
          ENDIF
     ENDIF
     RETURN
  999 WRITE(*,2010)
     RETURN 1
 1000 FORMAT(A)
 2000 FORMAT(T5,'*Error* -- Illegal char in field: ', A)
 2010 FORMAT(T5,'*Error* -- End-of-File on Input.')
     END
```

```
*LUFINV     Matrix Utilities (1)        Last Revision:  November 16, 1989
C                                       Source File:    LUFINV.FOR
      SUBROUTINE LUFINV(NROW, A, UL, AINV, IERR)
      INTEGER MAXN
      PARAMETER (MAXN = 50)
      INTEGER             NROW, IROW, JCOL, IERR
      DOUBLE PRECISION    A(NROW,NROW), UL(NROW,NROW), AINV(NROW,NROW)
      DOUBLE PRECISION    B(MAXN+1), X(MAXN)
      REAL                DIGITS
*************************************************************************
*  Invert matrix A, using the LU-factorization technique suggested      *
*  by Forsythe & Moler (see  DECOMP).  This routine has been further    *
*  optimized since only two elements are changed at each step through   *
*  the column loop.                                                     *
*                          D. E. Cawlfield,  Winter '88                 *
*************************************************************************
      IF (NROW .GT. MAXN) THEN
           PRINT *,'*LUFINV*  - NROW > MAXN, change and recompile'
           CALL SING(4)
      ENDIF
C
C  Generate first column of Ident...
C
      DO  10, IROW = 2, NROW
           B(IROW) = 0.D0
   10 CONTINUE
      B(1) = 1.D0
C
C  Call LU factorization, then step thru solution...
C
      CALL DECOMP(NROW, A, UL, IERR)
      DO  100, JCOL = 1, NROW
           CALL SOLVE(NROW, UL, B, X)
C  Iterative improvement (DIGITS not used here)
           CALL IMPRUV(NROW, A, UL, B, X, DIGITS, IERR)
           DO  20, IROW = 1, NROW
                AINV(IROW,JCOL) = X(IROW)
   20      CONTINUE
C
C  Generate next column of Ident for next pass...only two elements!
C
           B(JCOL  ) = 0.D0
           B(JCOL+1) = 1.D0
  100 CONTINUE
  999 RETURN
      END
```

```
*MATS       Matrix Utilities (2)        Last revision:  November 16, 1989
C                                       Source File:  MATS.FOR
      SUBROUTINE DECOMP(NN, A, UL, IERR)
      INTEGER   NN, MAXN, IERR
C     Double Precision Argument version
      DOUBLE PRECISION    A, UL
      INTEGER   IPS, N, I, J, NM1, K, IP, KP, KP1, IDXPIV
      DOUBLE PRECISION    SCALES, ROWNRM, BIG, SIZE, PIVOT, EM
      PARAMETER ( MAXN = 50 )
      DIMENSION A(NN,NN), UL(NN,NN), SCALES(MAXN)
      COMMON    /MATS/ IPS(MAXN)
*************************************************************************
*  DECOMP is a routine which performs an LU factorization of a matrix,  *
*  which is then solved by Gaussian elimination in routine SOLVE. This  *
*  is adapted from G. Forsythe & C. Mohler:  "Computer Solution of      *
*  Linear Algebraic Systems"; Prentice Hall; 1967.                      *
*  Parameters:                                                          *
*     NN         -  Order of matrix A (nrows = ncols)                   *
*     A          -  Matrix to be factored, must be declared NN X NN     *
*     UL         -  Upper-Lower factoriziation matrix output            *
*     IERR       -  Error flag.  Set if A is singular, or...see SING    *
*************************************************************************
      N  = NN
      IF (N .GT. MAXN) CALL SING(4, IERR)
C
C  Initialize IPS, UL, and SCALES
C
      DO  5, I = 1, N
          IPS(I) = I
          ROWNRM = 0.0
          DO  2, J = 1, N
              UL(I,J) = A(I,J)
              IF (ROWNRM .LT. ABS(UL(I,J)))  ROWNRM = ABS(UL(I,J))
    2     CONTINUE
          IF (ROWNRM .NE. 0.D0)  THEN
              SCALES(I) = 1.0D0 / ROWNRM
          ELSE
              CALL SING(1, IERR)
              SCALES(I) = 0.0D0
          ENDIF
    5 CONTINUE
C
C  Gaussian elimination with partial pivoting...
C
      NM1 = N - 1
      DO  17, K = 1, NM1
          BIG = 0.0D0
          DO  11, I = K, N
```

```
                IP = IPS(I)
                SIZE = ABS(UL(IP,K)) * SCALES(IP)
                IF (SIZE .GT. BIG) THEN
                     BIG    = SIZE
                     IDXPIV = I
                ENDIF
   11       CONTINUE
            IF (BIG .EQ. 0.D0) THEN
                CALL SING(2, IERR)
            ELSE
                IF (IDXPIV .NE. K) THEN
                     J           = IPS(K)
                     IPS(K)      = IPS(IDXPIV)
                     IPS(IDXPIV) = J
                ENDIF
                KP    = IPS(K)
                PIVOT = UL(KP,K)
                KP1   = K + 1
                DO  16, I = KP1, N
                     IP       =  IPS(I)
                     EM       = -UL(IP,K) / PIVOT
                     UL(IP,K) = -EM
                     DO  16, J = KP1, N
                          UL(IP,J) = UL(IP,J) + EM*UL(KP,J)
   16           CONTINUE
            ENDIF
   17 CONTINUE
      KP = IPS(N)
      IF (UL(KP,N) .EQ. 0.D0) CALL SING(2, IERR)
      RETURN
      END
*SOLVE
      SUBROUTINE SOLVE(NN, UL, B, X)
      INTEGER   NN, MAXN, IERR
      DOUBLE PRECISION    UL, X
      DOUBLE PRECISION    B, SUM
      INTEGER   N, NP1, IP, IPS, IM1, IP1, I, J, IBACK
      PARAMETER ( MAXN = 50 )
      DIMENSION UL(NN,NN), B(NN), X(NN)
      COMMON    /MATS/    IPS(MAXN)
*************************************************************************
*  SOLVE is the second stage of a Gaussian elimination program.  This   *
*  routine is usually called by IMPRUV, but may be used separately.     *
*  Solves the problem Ax = b, using the LU decomposition of A provided  *
*  by DECOMP.  Basically does two steps:  1)  solves Ly = b, then       *
*  2) back-substitution to find Ux = y.                                 *
*  Parameters:                                                          *
*     NN        -  Order of problem (nrows = ncols = NN)                *
```

```
*     UL        - The Upper-Lower decomposition of original matrix A  *
*     B         - The RHS of the system                                *
*     X         - The solution vector output                           *
*     IPS       - Pivoting information provided by DECOMP (in common) *
**************************************************************************
      N   = NN
      IF (N .GT. MAXN) CALL SING(4, IERR)
      NP1 = N + 1
      IP    = IPS(1)
      X(1)  = B(IP)
      DO  2, I = 2, N
           IP  = IPS(I)
           IM1 = I - 1
           SUM = 0.0
           DO  1, J = 1, IM1
                SUM = SUM + UL(IP,J) * X(J)
    1      CONTINUE
           X(I) = B(IP) - SUM
    2 CONTINUE
C
      IP   = IPS(N)
      X(N) = X(N) / UL(IP,N)
      DO  4, IBACK = 2, N
C  I goes (n-1) down to 1;
           I = NP1 - IBACK
           IP  = IPS(I)
           IP1 = I + 1
           SUM = 0.0
           DO  3, J = IP1, N
                SUM = SUM + UL(IP,J) * X(J)
    3      CONTINUE
           X(I) = (X(I) - SUM) / DBLE(UL(IP,I))
    4 CONTINUE
      RETURN
      END
*IMPRUV
      SUBROUTINE IMPRUV(NN, A, UL, B, X, DIGITS, IERR)
      INTEGER   NN, MAXN, IERR
      DOUBLE PRECISION    A, UL, X
      DOUBLE PRECISION    B, R, SUM
      REAL      DIGITS
      INTEGER   N, ITMAX, I, ITER, J
      DOUBLE PRECISION    EPS, XNORM, DXNORM, T, DX
      PARAMETER ( MAXN = 50 , EPS = 1.0E-8, ITMAX = 16 )
      DIMENSION A(NN,NN), B(NN), X(NN)
      DIMENSION UL(NN,NN), R(MAXN), DX(MAXN)
**************************************************************************
*  Carries out an iterative improvement to the solution obtained by    *
```

```
*  SOLVE.  See Forsythe & Mohler, reference in DECOMP routine.  EPS is *
*  the machine dependent round-off such that 1.0 + EPS = 1.0.  ITMAX   *
*  should be twice the number of digits in a floating point number.    *
*  The DIGITS parameter returns an estimate of the approximate digits  *
*  of accuracy.                                                        *
*  Parameters:                                                         *
*     NN         -  Order of problem (nrows = ncols = NN)              *
*     A          -  Original matrix A                                  *
*     UL         -  Upper-Lower decomposition of A (from DECOMP)       *
*     B          -  The RHS of the system                              *
*     X          -  The solution vector to be improved on (from SOLVE) *
*     DIGITS     -  As output, indicates the approximate digits of     *
*                   accuracy.                                         *
*     IERR       -  Error flag, see SING for conditions                *
*                                                                      *
*                                                                      *
*                                  David E. Cawlfield,  Fall '87       *
************************************************************************
      N     = NN
      IF (N .GT. MAXN) CALL SING(4, IERR)
      XNORM = 0.0
      DO  1, I = 1, N
           XNORM = AMAX1(XNORM, ABS(X(I)))
    1 CONTINUE
      IF (XNORM .EQ. 0.D0) THEN
           DIGITS = -LOG10(EPS)
      ELSE
           DO  9, ITER = 1, ITMAX
                DO  5, I = 1, N
                     SUM = 0.0D0
                     DO  4, J = 1, N
                          SUM = SUM + DBLE(A(I,J)) * DBLE(X(J))
    4                CONTINUE
                     SUM = B(I) - SUM
                     R(I) = SUM
    5           CONTINUE
C
C  It is essential that A(I,J)*X(J) yield a double precision result
C  and that the above + and - be double precision.
C
                CALL SOLVE(N, UL, R, DX)
                DXNORM = 0.0
                DO  6, I = 1, N
                     T    = X(I)
                     X(I) = X(I) + DX(I)
                     DXNORM = AMAX1(DXNORM,ABS(X(I)-T))
    6           CONTINUE
                IF (ITER .EQ. 1) THEN
```

```
                    DIGITS = -LOG10(AMAX1(DXNORM/XNORM, EPS))
                ENDIF
                IF (DXNORM .LE. EPS*XNORM) RETURN
    9       CONTINUE
        ENDIF
C
C  Iteration did not converge...
C
        CALL SING(3, IERR)
        RETURN
        END
*SING
        SUBROUTINE SING(IWHY, IERR)
************************************************************************
*  Catch-all error routine.  Given an error flag from DECOMP, IMPRUV   *
*  or SOLVE, this routine will print the appropriate error message.    *
*  The solution may or may not be allowed to continue depending upon   *
*  the severity of the error.                                          *
*                                  David E. Cawlfield,  Winter '88     *
************************************************************************
        INTEGER   IWHY
        IF      (IWHY .EQ. 1) THEN
             WRITE(*,2011)
        ELSE IF (IWHY .EQ. 2) THEN
             WRITE(*,2012)
        ELSE IF (IWHY .EQ. 3) THEN
             WRITE(*,2013)
        ELSE IF (IWHY .EQ. 4) THEN
             WRITE(*,2014)
             STOP 'Change DECOMP, IMPRUV & SOLVE'
        ELSE
             WRITE(*,2020)
        ENDIF
        IERR = -IWHY
        RETURN
 2011 FORMAT(1H0,'Matrix with zero row in DECOMPose.')
 2012 FORMAT(1H0,'Singular matrix in DECOMPose. Zero divide in SOLVE.')
 2013 FORMAT(1H0,'No convergence in IMPRUV. Matrix is nearly singular.')
 2014 FORMAT(1H0,'N > current MAXN (=50).  Change & re-compile.')
 2020 FORMAT(1H0,'The impossible has happened.  ELSE in SING.')
        END
```