Pub.

MARINE GRAVITY SOFTWARE

BY

H.D. VALLIANT

INTERNAL REPORT 84-5

Gravity, Geothermics & Geodynamics Division
Earth Physics Branch
Energy, Mines & Resources
1984

This document was produced by scanning the original publication.

Ce document est le produit d'une numérisation par balayage de la publication originale.

MARINE GRAVITY SOFTWARE

H.D. Valliant

PREFACE

Three FORTRAN programs, in routine use at the Earth Physics Branch, Department of Energy, Mines and Resources, for processing data from Lacoste and Romberg AIR/SEA gravimeters are documented herein. Two of these, LSIGRV and LSINAV were developed in-house while the third is a FORTRAN version of Dr. L.B.J. Lacoste's original BASIC program CCP. All programs are documented with imbedded comment lines and introductory notes. This software is provided for information purposes only. The Earth Physics Branch does not warrant that it will be free from error nor does it provide any on-going support.

TROIS PROGRAMMES D'ORDINATEUR AU SUJET DES DONNEES DU CHAMP DE GRAVITE MARITIME

H.D. Valliant

PREFACE

Trois programmes FORTRAN, lesquels sont disponibles à la Direction de la physique du globe, Energie, Mines et Ressources Canada, sont conçus pour le traitement des données des gravimètres maritimes et aéronautiques de LaCoste et Romberg. Deux de ces programmes, LSIGRV et LSINAV ont été conçus à la Direction de la physique du globe et le troisième est basé sur le programme CCP écrit en BASIC par M. L. B.J. LaCoste. Tous les trois contiennent les notes d'introductions et les lignes de commentaires. Ces programmes son offerts pour renseignements seulement. La Direction de la physique du globe ne peut garantir ces programmes, ni assurer qu'ils ne contiennent pas d'erreurs.

Publications for which the attached software has been used

Open file Gravity Maps:

76-1	Queen Charlotte Sound
76-2	La Perouse Bank
77-1	Nootka Island
77-2	Brooks Peninsula
77-3	Coast of Labrador
82-3 (A)	Hudson Bay (Free air)
82-3 (B)	Hudson Bay (Bouger)
82-15	B.C. Coast

Other Related Publications:

"Theory and evaluation of the LaCoste and Romberg three-axis inertial platform for marine gravimetry". H.D. Valliant and L.J.B. LaCoste, Geophysics, 41, pp. 459-467, 1976.

"Sea-gravimeter Trials on the Halifax test range aboard CSS HUDSON, 1972". H.D. Valliant, J. Halpenny, R. Beach and R.V. Cooper, Geophysics, 41, pp. 700-711, 1976.

"Re-evaluation of the effects of horizontal accelerations on the LaCoste and Romberg gravimeter platform". H.D. Valliant, Geophysics, pp 137-141, 1979.

"Marine navigation with the LaCoste and Romberg inertial platform". H.D. Valliant, Geophysics, 46, p. 1469, 1981.

"Position measurements with the LaCoste and Romberg Air/Sea gravimeter". H.D. Valliant and R.V. Cooper, Geophysics, 46, pp. 40-44, 1981.

"Field Trials with the LaCoste & Romberg straight-line gravimeter". H.D. Valliant, Geophysics, 48, pp. 611-617, 1983.

PROGRAM LSIGRV

BY

H.D. Valliant, R. Beach & J. Halpenny

Gravity, Geothermics and Geodynamics Division

Earth Physics Branch

Energy, Mines & Resources

24 July 1982

LSIGRV

Purpose: To read gravimeter tapes and reduce data to observed g.

To Execute: RUN DY0: LSIGRV and follow console instructions.

Utilities: The executor program (EXEC) allows the operator to choose a number of utilities as follows:

1) Process data

Processes data, producing observed g, until the end time or a time gap is observed and returns to EXEC for further instructions. The process is self explanatory through console prompts. A choice of console listing or not is provided. Data is reduced using the beam-slope method.

2) Reread/Restart

This utility is left over from the days of our fine DIGIDATA tape recorders which provided many read errors but no re-read capability. The utility functions and permits the tape to be rewound (now under program control) a specified number of blocks with automatic restart if the missing time is recovered. You probably won't use this feature.

3) Interpolate

Allows semi-automatic interpolation through missing values or bad data. Unfortunately the computer is not clairvoyant and needs good data on either side of the gap before it can interpolate. Read ahead in provided to solve this problem.

4) TDUMP

Allows examination of data from a given starting time to a given end time. For a quick look give times 0 and 999999. respectively for start and end. You can

exit with CNTRL "C" when you have seen enough. Remember before trying this to close files (path-6) or you may lose your data. A normal exit on finding the end-time returns to EXEC and any open files remain open.

5) Read Ahead

Permits the tape recorder to read one or more records to get past bad data. When you are satisfied that a good record has been reached, take path 3 and the computer does the rest.

6) Close files and Exit

Exactly as the title implies. No rewind of the tape is performed allowing you to terminate and restart part way through a tape (e.g. line-by-line processing). If after a break to EXEC (End time reached, bad data etc.) you choose path 1 the new data will be appended to the old file. If you choose 6 and rerun the program you will get a new file. Use the switches on the tape drive to rewind the tape as necessary.

Input

Mag Tape records at 10 sec sample rate. S - meter standard format.

Output

One file in list directed format containing DAY, TIME, OBSG

List Option

An option permits surpressing terminal listing. This has two advantages and two disadvantages.

Advantages 1) saves paper

2) program runs faster

Disadvantages 1) Listed on the terminal but not included in the disk file is the Total Correction and its difference from that recorded on tape.

2) If you get a bad time interrupt you may have trouble figuring out where you are. However, using READ AHEAD and INTERPOLATE should let you recover.

If you select no listing you can still get a listing of GOBS by typing the file using the commmand.

TYPE filename

Note

Subroutine RECRD constitutes an IEEE-488 bus handler. Normal FORTRAN I/O statements may be substituted as desired.

```
      PROGRAM LSIGRV

C             ****************************************
C             ****    INITIALIZE AND EXECUTE    ****
C             ****************************************
C             ************  GRVSYS.06  ************
C             ************  22/07/82   ************
C
C
      DIMENSION IFILE(5)
      DIMENSION TC(60)
      DIMENSION ST(60),FDATA(9), CC(60),AB(60)
      COMMON /RECORD/ IBUF(240)
      DIMENSION DREC(30)
      COMMON /TABLES/ STFW(60),TCFW(60),ABFW(60)
C
C****** SPRING TENSION FILTER WEIGHTS:
C       ( DIGITAL + 3*20 )
C
      DATA STFW/-.00011,.00002,.0002,.00045,.00077,.00119,
     1  .00172,.00237,.00317,.00414,.00528,.00662,
     1  .00816,.00991,.01187,.01404,.0164,.01892,
     1  .02158,.02434,.02713,.02989,.03257,.03509,
     1  .03738,.03936,.04098,.04217,.04291,.04316,
     1  .04291,.04217,.04098,.03936,.03738,.03509,
     1  .03257,.02989,.02713,.02434,.02158,.01892,
     1  .0164,.01404,.01187,.00991,.00816,.00662,
     1  .00528,.00414,.00317·.00237,.00172,.00119,
     1  .00077,.00045,.0002,.00002,-.00011,-.00019/
C
C****** TOTAL CORRECTION FILTER WEIGHTS:
C       ( DIGITAL )
C
      DATA TCFW/-.00034,-.00038,-.00041,-.00044,-.00046,-.00046,
     1  -.00044,-.00039,-.0003,-.00015,.00007,.00037,
     1  .00079,.00133,.00202,.00289,.00396,.00526,
     1  .00679,.00859,.01066,.01299,.01558,.01841,
     1  .02143,.0246,.02785,.0311,.03426,.03723,
     1  .03992,.04223,.04408,.04539,.04613,.04626,
     1  .04579,.04474,.04315,.04109,.03864,.03589,
     1  .03292,.02984,.02671,.02362,.02063,.0178,
     1  .01516,.01274,.01056,.00863,.00694,.00548,
     1  .00424,.00321,.00235,.00166,.00111,.00068/
C
C****** BEAM FILTER WEIGHTS CONVOLVED WITH FIRST DERIVATIVE:
C       ( DIGITAL + 1ST DERIVATIVE )
C
      DATA ABFW/-.00036,-.00003,-.00003,-.00002,-.00001,.00001,
     1  .00003,.00007,.00012,.00018,.00026,.00036,
     1  .00048,.00061,.00078,.00097,.00118,.00141,
     1  .00166,.00193,.00220,.00246,.00271,.00292,
     1  .00310,.00321,.00324,.00320,.00307,.00283,
     1  .00249,.00208,.00158,.00102,.00043,-.00016,
     1  -.00076,-.00132,-.00182,-.00225,-.00260,-.00286,
     1  -.00302,-.00311,-.00311,-.00304,-.00291,-.00273,
     1  -.00253,-.00230,-.00205,-.00181,-.00157,-.00135,
     1  -.00113,-.00094,-.00077,-.00062,-.00049,-.00089/
C
C
C
```

```
C
C****** ZERO DATA BUFFERS AND SWITCHES:
C
        TYPE *, 'OUTPUT FILENAME ?: (------,---)'
        ACCEPT 88, IFILE
        TYPE *, 'KILL LISTING 1=YES 0=NO'
        ACCEPT *, ISUPP
        CALL ASSIGN (8,IFILE)
88      FORMAT (5A2)
        CALL ASSIGN (4,'TT:')
        DO 3 J=1,4
    3   FDATA(J)=0.
        ISTART=0
C
C****** CHOOSE ROUTINE TO EXECUTE:
C
    5   CONTINUE
        WRITE(4,9900)
9900    FORMAT(' **EXEC?**     1=PROCESS DATA '/15X,'2=REREAD/RESTART'
     1/15X,'3=INTERPOLATE'/15X,'4=TDUMP'/15X,'5=READ AHEAD'/
     215X,'6=CLOSE FILES,EXIT')
        READ (4,*) IPATH
        IF((IPATH.LT.1).OR.(IPATH.GT.7)) GO TO 5
        GO TO (9,100,150,200,300,90), IPATH
C
C
C
C
C               ****************************
C               ****    GRAV2  (MAIN)   ****
C               ****************************
C
C****** CONSTANTS LOCAL TO GRAV2:
C
    9   X=0.
        IFIRST=0
        IDCNT=0
        IFLDS=0
        IERR=0
        J1=0
        TYPE *, 'CROSS-COUPLING? 1=YES;0=NO'
        ACCEPT *, ICCSW
        CCSW=ICCSW
        TYPE *,' METER NUMBER? '
        ACCEPT *,MNO
        CALL ASSIGN (11,'DY0:BASEG.DAT')
801     READ (11,*,END=803) MMNO,G1,B1,B0
        IF (MMNO.NE.MNO) GO TO 801
        GO TO 807
803     TYPE *,' METER ',MNO,' NOT ON FILE DK:BASEG.DAT'
        TYPE *, 'SUPPLY GMETER CONSTANT: S-56=0.9968: S-41=0.9903:
     1 SL-1=0.926'
        ACCEPT *, G1
        IF(ISTART.NE.0) GO TO 10
C
C****** ENTER BASE INFORMATION:
        TYPE *, 'ENTER BASE VALUES: SMETER & GRAVITY-980000.'
        WRITE(4,8801)
 8801   FORMAT(' ','------. ----.--')
        READ(4,*) B1,B0
        WRITE(4,8804)
 8804   FORMAT(' ','** GRAVITY DATA REDUCTION **')
807     WRITE(4,8803) B1,B0,MNO,G1
```

```
 8803   FORMAT('0','BASE DIAL READING :',F8.0,/,' BASE G-VALUE :',F8.2/,
        1    ' CONSTANT FOR ',I4,' = ',F8.4)
        B1=(B1/10)*G1
        ISTART=1
C
C****** DISPLAY FIRST START TIME FROM TAPE:
C
   10   CONTINUE
        CALL RECRD  (0,DREC)              ! READ
C++++++ INSERT READ OF RAW DATA HERE.
        DAY=AINT(DREC(1)/1000.)+.1
        TIME=DREC(2)
        CALL TIMER(TIME,510.,DAY)
        WRITE (4,11) TIME,DAY
11      FORMAT (' CURRENT RECORD AT TIME:',F7.0,2X,F4.0)
   12   WRITE(4,900)
  900   FORMAT(' ','OK?')
        TYPE *, '1=CONTINUE; 2=EXEC; 3=READ ANOTHER RECORD; 4=DO ALL'
        READ(4,*) IPATH
        IF((IPATH.LT.1).OR.(IPATH.GT.4)) GO TO 12
        GO TO (15,5,10,1300), IPATH
1300    I = TIME/500 +2.001                       !next full 5 min block
        T0 = I*500.
        I = DAY
        D0 = I
        T9 = 0                                    !Default end time
        D9 = DAY +3
        GO TO 17
C
C****** ENTER START AND END TIMES:
C
   15   CONTINUE
        WRITE (4,8805)
8805    FORMAT (' START-TIME? START-DAY? END-TIME? END-DAY')
        WRITE(4,8802)
 8802   FORMAT(' ','------, ---, ------, ---,')
        READ(4,*) T0,D0,T9,D9
        WRITE(4,*) T0,D0,T9,D9
C
C****** CALCULATE ACTUAL TAPE START AND END TIMES:
C
17      D8=D0
        T2=T0
        CALL TIMER(T2,-450.,D8)
        CALL TIMER(T9,510.,D9)
C
C****** READ TAPE TO THIS START TIME:
C
   20   CONTINUE
        CALL RECRD (0,DREC)
C++++++ INSERT READ OF RAW DATA HERE.
        DAY=AINT(DREC(1)/1000.)
        BEAML = DREC(5)
        IF(DREC(2).LT.T2) GO TO 20
        IF((DREC(2).EQ.T2).AND.(DAY.EQ.D8)) GO TO 25
C
C****** ERROR - TAPE TIME GREATER THAN START TIME:
C
C
        TYPE *, 'TAPE TIME:',DREC(2),DAY,'GREATER THAN'
        TYPE *, 'START TIME:',T2,D8
        GO TO 10
   25   CONTINUE
```

```
      WRITE(4,26) DREC(2)
   26 FORMAT(' ','START TIME = ',F7.0,' + 4 MIN. AND 50 SECS.')
   27 CONTINUE
      WRITE(4,900)
      TYPE *, '1=OK;2=EXEC;3=REENTER START-TIME'
      READ(4,*) IPATH
      IF((IPATH.LT.1).OR.(IPATH.GT.3)) GO TO 27
      GO TO (40,5,9), IPATH
C
C****** READ NEXT RECORD AND CHECK TIMES MATCH:
C
   30 CONTINUE
      CALL RECRD(0,DREC)
C++++++ INSERT READ FOR RAW DATA HERE.
      IF (DREC(1).EQ.999000.) GO TO 80                          !End of file
   40 IF(DREC(2).EQ.T2) GO TO 45
      IF (DREC(2).LT.240000.) GO TO 43
      DREC(2) = DREC(2)-240000.
      DREC(1) = DREC(1)+1
      GO TO 40
43    TYPE *, 'BAD TIME:',DREC(2),'SHOULD BE:',T2
      GO TO 5
C
C****** CHECK FOR END TIME AND EDIT DATA:
C
   45 IF((DREC(2).EQ.T9).AND.(D8.EQ.D9)) GO TO 80
      IF(ABS(DREC(4)).NE.0.) GO TO 55
      TYPE *, 'ZERO SPRING TENSION DETECTED AT:',DREC(2)
      DREC(4)=ST0
      DREC(5)=AB0
      DREC(2)=CC0
      DREC(7)=TC0
C
C****** CALCULATE NEXT VALID TIME AND DAY:
C
   55 A5=DREC(2)
      CALL TIMER(T2,10.,D8)
C
C****** RESOLVE ST , TC FROM TAPE:
C
      ST0=DREC(4)
      AB0=DREC(5)
      CC0=DREC(6)
      TC0=DREC(7)
C **   COMPUTE TC FROM BEAM VELOCITY AND CC
C     TC0 = (DREC(5)-BEAML)*30. -DREC(6)
      BEAML = DREC(5)
C
C****** FILTER AND FORM DIGITAL G-OBSERVED:
C
   58 J1=J1+1
      IF(J1.GT.1) GO TO 59
      SMALL=ST0
   59 ST(J1)=ST0-SMALL
      TC(J1)=TC0
      AB(J1)=AB0
      CC(J1)=CC0
      IF((IFIRST.EQ.5).AND.(J1.LT.60)) GO TO 180
      IF(J1.LT.60) GO TO 30
      J1=30
      SUMAB=0.
      SUMCC=0.
      SUMST=0.
```

```
        SUMTC=0.
        DO 60 I=1,60
        SUMAB=SUMAB+AB(I)*ABFW(I)
        SUMCC=SUMCC+CC(I)*TCFW(I)
        SUMST=SUMST + ST(I) * STFW(I)
   60   SUMTC=SUMTC + TC(I) * TCFW(I)
        CALCTC=(SUMAB*3.)+(SUMCC*.1*CCSW)
        SUMTC=-SUMTC*.1
        GOBS=((SUMST+SMALL)*.1)*G1+SUMTC*G1
        GOBS1=((SUMST+SMALL)*.1)*G1+CALCTC*G1
        DO 65 I=1,30
        TC(I)=TC(I+30)
        AB(I)=AB(I+30)
        CC(I)=CC(I+30)
   65   ST(I)=ST(I+30)
C
C
C****** STORE RESULT AND RETURN FOR MORE INPUT:
C
        IF(IFIRST.NE.0) GO TO 70
        IDCNT=1
        IFIRST=1
   70   TIME=DREC(2)
        DAY8=D8
        CALL TIMER(TIME,-500.,DAY8)
        FDATA(1)=TIME/100.
        FDATA(2)=GOBS1-B1+B0+.05
        FDATA(4)=GOBS-GOBS1
        FDATA(3)=CALCTC
        IF (ISUPP .EQ. 1) GO TO 75
        WRITE(4,998) DAY8,FDATA(1),FDATA(2),FDATA(3),FDATA(4)
75      WRITE(8,998) DAY8,FDATA(1),FDATA(2)
C       IF(IDCNT.EQ.299) GO TO 80                          !Why stop?
        IDCNT=IDCNT + 1
        IF(IFIRST.EQ.5) GO TO 180
        GO TO 30
C
C****** END OF DIGITAL GRAVITY ROUTINE:
C
   80   CONTINUE
C
        TYPE *, 'IDCNT= ',IDCNT-1
        TYPE *, 'OUTPUT: DAY;TIME;OBSERVED GRAVITY WITH COMPUTED TC;'
        TYPE *, '        CALC TC; RECRDED  MINUS  CALC TC'
        GO TO 5
  998   FORMAT(' ',F4.0,1X,F5.0,1X,F9.2,1X,F7.2,1X,F7.2)
   90   CONTINUE
        TYPE *, 'END OF GRAV2'
        CALL CLOSE (8)
        STOP
C
C
C
C               ****************************************
C               ****    GRAV2  (RESTART, REREAD)    ****
C               ****************************************
C
C****** RESTART AT LAST UNUSED TIME (A5+1):
C
  100   CONTINUE
C
        WRITE(4,9992)
 9992   FORMAT(' ','BACKSPACE HOW MANY?')
```

```
        READ(4,*) NN
        IF(NN.EQ.0) GO TO 105
        CALL RECRD (-NN)
C++++++ INSERT BACKSPACE FOR RAW DATA HERE.
  105   CONTINUE
        NNN=NN+10
        DO 110 J=1,NNN
C++++++ INSERT READ FOR RAW DATA HERE.
        CALL RECRD (0,DREC)
        IF(DREC(2).EQ.A5) GO TO 115
  110   CONTINUE
        WRITE(4,9904) A5,DREC(2)
 9904   FORMAT(' NO TIME MATCH AFTER 50 READS',F7.0,1X,F7.0)
        GO TO 5
  115   CONTINUE
        WRITE(4,9905)
C
 9905   FORMAT (' ','GOT IT!')
        GO TO 30
C
C
C
C               ****************************************
C               ****    GRAV2  (INTERPOLATION)    ****
C               ****************************************
C
C****** INTERPOLATION ROUTINE:
C
  150   CONTINUE
  180   IF(IFIRST.EQ.5) GO TO 185
        IFIRST=5
        TCOLD=TC0
        STOLD=ST0
        CCOLD=CC0
        ABOLD=AB0
        TNEW=DREC(2)
        TOLD=A5
        THOLD=DREC(2)
        CALL TIMER(TNEW,-TOLD,-5.)
        FACT=1./(TNEW/10.)
C
  185   CALL TIMER(TOLD,10.,DAY8)
        IF(TOLD.EQ.THOLD) GO TO 190
        ST0=ST0+(DREC(4)-STOLD)*FACT
        TC0=TC0+(DREC(7)-TCOLD)*FACT
        AB0=AB0+(DREC(5)-ABOLD)*FACT
        CC0=CC0+(DREC(6)-CCOLD)*FACT
        DREC(2)=TOLD
        GO TO 58
C
  190   IFIRST=1
        DREC(2)=THOLD
        T2=THOLD
        GO TO 45
C
C
C               **********************************
C               ****   READ  AHEAD           ****
C               **********************************
C
300     CONTINUE
        TYPE *, 'HOW MANY RECORDS'
305     ACCEPT *, J
```

```
        IF (J .EQ. 0) GO TO 5
        DO 310 I=1,J
        CALL RECRD (0,DREC)
        DAY=AINT(DREC(1)/1000.)
        WRITE (4,9914) DAY,DREC(2),DREC(4),DREC(7),DREC(6)
310     CONTINUE
        TYPE *, 'MORE ?; 0=NO'
        GO TO 305
C
C
C               ******************************
C               ****    TDUMP  (MAIN)    ****
C               ******************************
C
C****** TAPE PRINT ROUTINE:
C
  200   CONTINUE
        TYPE *, 'REWOUND TAPE ?'
        WRITE(4,8805)
        WRITE(4,8802)
        READ(4,*) ST1,SD1,ST9,SD9
        WRITE(4,8800) ST1,SD1,ST9,SD9
 8800   FORMAT(' ',4F10.2)
  216   CONTINUE
C++++++ INSERT READ FOR RAW DATA HERE.
        CALL RECRD (0,DREC)
        DAY=AINT(DREC(1)/1000.)
        IF((DREC(2).LT.ST1).OR.(DAY.LE.SD1)) GO TO 216
        IF((DREC(2).GT.ST9).OR.(DAY.GE.SD9)) GO TO 225
C++++++ WRITE DATA HERE.
        WRITE(4,9914) DAY,DREC(2),DREC(4),DREC(7),DREC(6)
 9914   FORMAT(' ',5F10.1)
        GO TO 216
C
  225   CONTINUE
        WRITE(4,9915)
 9915   FORMAT(' ','END DUMP.')
        GO TO 5
        END
C
C
C
        SUBROUTINE TIMER(TIME1,TIME2,DAY)
C
C****** SPLIT TIMES TO HOURS, MINUTES, SECONDS AND KEEP SIGNS.
C
        TH1=AINT(TIME1/10000.)
        TM1=AINT((TIME1-TH1*10000.)/100.)
        TS1=TIME1-(TH1*10000.+TM1*100.)
        TH2=AINT(TIME2/10000.)
        TM2=AINT((TIME2-TH2*10000.)/100.)
        TS2=TIME2-(TH2*10000.+TM2*100.)
C
        AMNS=0.
        AHRS=0.
C
C****** RESOLVE SECONDS TO MINUTES AND SECONDS.
C
        TS1=TS1+TS2
        IF(TS1.GE.0.)GO TO 10
        TS1=TS1+60.
        TM2=TM2-1.
   10   IF(TS1.LT.60.) GO TO 20
```

```
        AMNS=AINT(TS1/60.)
        TS1=TS1-AMNS*60.
C
C****** RESOLVE MINUTES TO HOURS AND MINUTES.
C
   20   TM1=TM1+AMNS+TM2
        IF(TM1.GE.0.) GO TO 30
        TM1=TM1+60.
        TH2=TH2-1.
   30   IF(TM1.LT.60.) GO TO 40
        AHRS=AINT(TM1/60.)
        TM1=TM1-AHRS*60.
C
C****** RESOLVE HOURS AND ADJUST DAY.
C
   40   TH1=TH1+AHRS+TH2
        IF(TH1.LT.24.) GO TO 50
        TH1=TH1-24.
        DAY=DAY+1.
   50   IF(TH1.GE.0.) GO TO 60
        TH1=TH1+24.
        DAY=DAY-1.
C
C****** BUILD NEW TIME AND RETURN IT IN TIME1.
C
   60   TIME1=TH1*10000.+TM1*100.+TS1
C
C****** IF NEGATIVE DAY SENT IN SEND BACK TOTAL SECONDS.
C
        IF(DAY.GE.0.) RETURN
        TIME1=TH1*3600.+TM1*60.+TS1
        RETURN
        END
        SUBROUTINE RECRD(NPOS,DREC)
C    1982 VERSION WITH SKIP FWD, BKD, AND EOF SUPPORT
C     NPOS=     0       READ 9-track gravimeter tape
C               +N      SPACE FORWARD N RECORDS
C               -N      BACKSPACE N RECORDS
C               -999    REWIND
        COMMON / RECORD / BUFF(240)
        DIMENSION DREC(30)
        BYTE BUFF,XTAB(255), XBEG(256),BIN(240),IBGR(8),IST
        EQUIVALENCE (XTAB,XBEG(2))
C**             TRANSLATE 9-TRACK EBCIDC  TO ASCII
        DATA XBEG/13*0,13,23*0,10,26*0,1H ,10*0,1H.,1H<,1H(,
        9     1H+,1H!,1H&,
        1  10*0,1H$,1H*,1H),1H;,1H^,1H-,1H/,8*0,1H|,1H,,1H%,1H_,
        2  83*0,1HA,
C       2  1H>,1H?,9*0,1H`,,1H:,1H#,1H@,1H',1H=,1H",65*0,1HA,
        3  1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,7*0,1HJ,1HK,1HL,1HM,1HN,
        4  1HO,1HP,1HQ,1HR,8*0,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ,6*0,
        5  1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,6*0 /
        DATA ISTART /0/
        IF (ISTART.EQ.1.AND.NPOS.EQ.0) GO TO 150        !ALREADY SET UP
        CALL IBSEND ('SD(1)RC(0)',10,2) ! 7-TRACK ON DRIVE 1
        IF (NPOS.EQ.0) GO TO 150
        ISTART = 0
        IF (NPOS.NE.-999) GO TO 100
        CALL IBSEND ('RW',2,2)                  ! REWIND
        NREC = 0
        RETURN
100     ENCODE (8,4,IBFR) NPOS                  ! SKIP COUNT
        CALL IBSEND(IBFR,8,2)
```

```
      NREC = NREC + NPOS
      RETURN
150   LENO= IBRECV (BIN,240,3)                ! READ
      I = IBRECV(IST,1,2)                     !Get status byte
      IF ((IST.AND.'100).EQ.0) GO TO 160      !End of file
      DREC(1) = 999000.                       !Flag it
      RETURN
160   DO 180 I=1,LENO
      K= BIN(I)
      IF(K.LT.0) K=K+256                      !UPPER BIT IS NOT SIGN
180   BUFF(I) = XTAB(K)                       ! BCD TO ASCII
      NREC = NREC + 1
      ISTART = 1
      DECODE (LENO,1,BUFF,ERR=220) DREC
      RETURN
220   TYPE 3,' READ ERROR RECORD ',NREC,' LENGTH ', LENO,IPEEK(MTS)
      TYPE 2,(BIN(I),I=1,LENO)
      TYPE 7,(BUFF(I),I=1,20)
      TYPE 7,(BIN(I),I=1,20)
7     FORMAT (1X,20I3)
      TYPE 2,(BUFF(I),I=1,LENO)
1     FORMAT (F6.0,3F8.0,17(3X,F5.0),9F8.0)
2     FORMAT(1X,64A1)
3     FORMAT (1X,A20,I5,A10,I5,07)
4     FORMAT('MB(',I4,1H))
      END
      SUBROUTINE IBCMD (NTALK,NLIST)
      BYTE ARR(3)
      COMMON /IBLINK/ IBSR,IBDR,LTALK,LLIST
      DATA ARR /1H?, 1H ,1H /
      DATA  IBSR,IBDR,LTALK,LLIST / '160150, '160152, 0,0/
      DATA IFCW /'10/                                 !SET IFC AT START ONLY
100   CALL IPOKE (IBSR,IFCW)                          !IFC OR TAKE CONTROL
      IFCW = 1                                        !TAKE CONTROL
      LLIST = NLIST                                   !RESET LIST
      LTALK = NTALK
      IBY = 1                                         !BYTE COUNTER
      ARR(2) =NLIST +32
      ARR(3) = NTALK +64
      NW = -32767                                     ! 5 SEC TIMEOUT
      GO TO 140                                       ! WAIT FOR CMD DONE
120   IF (IBY.GT.3) RETURN                            ! LAST CHAR ACCEPTED
      CALL IPOKEB (IBDR,ARR(IBY))                     ! SEND BYTE
      IBY =IBY +1
      NW = -10000                                     !WAIT COUNT
140   IF ((IPEEK(IBSR).AND.'2000).NE.0) GO TO 120     !CMD DONE
      NW = NW +1
      IF (NW.LT.0) GO TO 140
      PAUSE 'CONTROLLER NOT READY, CHECK H-WARE,
     1INPUT 'C' TO CONTINUE'
      IFCW = '10                                      !Set IFC
      GO TO 100                                       ! TRY AGAIN
      END
      SUBROUTINE  IBSEND (ARR,LEN,NLIST)
      BYTE ARR(LEN)
      COMMON /IBLINK/ IBSR,IBDR,LTALK,LLIST
100   IF (NLIST.NE.LLIST.OR.LTALK.NE.0) CALL IBCMD(0,NLIST)
      CALL IPOKE (IBSR,'44)                           !TALK STATUS
      IBY = 1                                         !BYTE COUNT
      NW = -32767                                     ! 5 SEC TIMEOUT
      GO TO 220                                       !WAIT FOR TALKER READY
200   CALL IPOKEB (IBDR,ARR(IBY))                     !SEND DATA
      IBY = IBY + 1
```

```
        IF (IBY.GT.LEN) RETURN                                  !LAST BYTE SENT
        NW = -1000                                              !TIMEOUT COUNTER
220     IF ((IPEEK(IBSR).AND.'1000).NE.0) GO TO 200             !READY AGAIN
        NW = NW + 1
        IF (NW.LT.0) GO TO 220
        PAUSE 'TAPE NOT READY, CHECK H-WARE,
        1INPUT 'C' TO CONTINUE'
        GO TO 100                                               ! TRY AGAIN
        END
        FUNCTION IBRECV (ARR,LEN,NTALK)
C       ARRAY REFRENCES TAKE 177 US/READ VS 237 US/READ FOR IPEEK
        COMMON /IBLINK / IBSR,IBDR,LTALK,LLIST
        BYTE ARR(LEN),IPEB(1)
        DIMENSION IPEEKA(1)
        NBDR = IBDR-IADDR(IPEB) + 1                             ! BYTE ARR
        NBSR = ( IBSR - IADDR(IPEEKA))/2 +1                     ! WORD ARRAY
90      IF(NTALK.NE.LTALK.OR.LLIST.NE.0) CALL IBCMD(NTALK,0)
        IPEEKA(NBSR)= '20                                       !SET LISTENER ON
        IBRECV = 0
        NW =-32767                                              ! 5 SEC TIMEOUT
        GO TO 120                                               !WAIT LOOP
100     IBRECV = IBRECV + 1
        ARR(IBRECV) =IPEB(NBDR)                                 !STORE NEW BYTE
        IF (IBRECV.GE.LEN) RETURN                               ! FINISHED
        NW = 0                                      !NO WAIT IF BLOCK STARTED
120     IF ((IPEEKA(NBSR).AND.'400).NE.0) GO TO 100
        NW = NW + 1                                             !NOT READY, COUNT
        IF (NW.LT.0) GO TO 120
        IF (IBRECV.GT.0) RETURN                                 !END OF BLOCK
        PAUSE 'TAPE DRIVE NOT RESPONDING, CHECK-H-WARE,
        1INPUT 'C' TO CONTINUE'
        CALL IBSEND('SD(1)',5,2)                                !Select drive again
        LTALK=0
        GO TO 90                                                !TRY AGAIN
        END
```

PROGRAM LSINAV

by

H.D. Valliant & R. Beach

Gravity, Geothermics and Geodynamics Division

Earth Physics Branch

Energy, Mines & Resources

1 July, 1980

LSINAV

Purpose: To reduce observed gravity to free-air anomalies.

Execution: RUN DYO:LSINAV and respond to prompts

Inputs: 1) Disk file on DY1: containing:

DAY, TIME, LAT (Deg), LAT (min), LONG (deg), LONG (min), DEPTH, MAGNETICS in list directed format.

2) Disk file on DY1: containing: DAY, TIME, GOBS. (ouput from LSIGRV)

Outputs: 1) Disk file on DY1: containing:

DAY, TIME, GOBS, EOTVOS, FREE-AIR, LAT, LONG, DEPTH

Notes: 1) The program expects data to be at five minute intervals on the five minute mark. It will however interpolate through missing navigation fixes.

2) It also requires, to allow for filtering, 15 minutes of navigation data prior to first gravity value and 15 min. after. If not, the result is a FORTRAN run-time error, 'Invalid Logical Unit No' as the program tries to close files that never opened.

```
      PROGRAM LSINAV
C             ****************************
C             ****    NAV2  (MAIN)    ****
C             ****************************
C
C
C****** CONSTANTS LOCAL TO NAV2:
C
      DIMENSION IFILE(6),JFILE(6),KFILE(6)
      DIMENSION  SAVEP(3,4),FLOATF(7),FLONGF(7),FDATA(7),FLATF(7)
      DIMENSION FWNAV(7)
      DATA FWNAV/0.,-.2,-.1,0.,.1,.2,0./
      TYPE *, 'NAVIGATION FILENAME ? (------,---)'
      ACCEPT 88, IFILE
      TYPE *, 'GRAV2 OUTPUT FILENAME ? (------,---)'
      ACCEPT 88, JFILE
      TYPE *, 'NAV2 OUTPUT FILENAME ? (------,---)'
      ACCEPT 88, KFILE
D     TYPE 999, IFILE,JFILE,KFILE
D999  FORMAT (' ',5A2)
88    FORMAT (6A2)
  300 C0=.0174533
      C2=5185.9
      C3=5.7704
      C4=7.50259
      C5=.004154
      C6=5.278895E-3
      C7=2.3462E-5
      R1=3444.05
      J1=0
      J2=0
      IFIRST=0
      X=0.
D     TYPE 999, IFILE,JFILE,KFILE
      CALL ASSIGN (4,'TT:')
      CALL ASSIGN (8,JFILE)
      CALL ASSIGN (9,IFILE)
      CALL ASSIGN (10,KFILE)
      WRITE (4,381)
C
C
  323 J2=J2+1
      READ(8,*,END=385) DAY,FDATA(1),FDATA(2)
      GOBS=FDATA(2)
      IF(GOBS.EQ.0..AND.FDATA(1).EQ.0.) GO TO 385
      IF(J2.GT.1) GO TO 324
      TTIM=FDATA(1)*100.
      CALL TIMER(TTIM,-1500.,DAY)
      GO TO 325
C
  324 CONTINUE
      CALL TIMER(TTIM,500.,DAY)
  325 CONTINUE
C
  340 CONTINUE
D     TYPE *, 'DEBUG MSG #1',DAY,TTIM
      CALL POSN(DAY,TTIM,FDAY,FTIME,FLATD,FLONGD,DEPTH,IEOF)
      IF(IEOF.EQ.0) GO TO 350
      IF(IEOF.EQ.2) GO TO 385
      WRITE(4,9920) TTIM,DAY,FTIME,FDAY
```

```
 9920   FORMAT(' ','EOF ON POSITION FILE...',/,'LOOKING FOR.........',
     1  F6.0,1X,F4.0,/,'LAST TIME READ........',F6.0,1X,F4.0)
9922    FORMAT(' ','END JOB')
        GO TO 385
C
  350   J1=J1+1
C
C****** SAVE POSITIONS FOR OUTPUT:
C
        DO 354 I=1,3
        SAVEP(1,I)=SAVEP(1,I+1)
        SAVEP(2,I)=SAVEP(2,I+1)
  354   SAVEP(3,I)=SAVEP(3,I+1)
C
C****** FILTER POSITIONS FOR VELOCITIES:
C
        DO 356 I=1,6
        FLATF(I)=FLATF(I+1)
  356   FLONGF(I)=FLONGF(I+1)
        FLATF(7)=FLATD
        FLONGF(7)=FLONGD
        VELE=0.
        VELN=0.
        DO 357 I=1,7
        VELE=VELE+FLONGF(I)*FWNAV(I)
  357   VELN=VELN+FLATF(I)*FWNAV(I)
        SAVEP(1,4)=FLATD
        SAVEP(2,4)=FLONGD
        SAVEP(3,4)=DEPTH
        IF(J1.GT.6) GO TO 360
        GO TO 324
C
C****** CONVERT VELOCITIES TO KNOTS AND CALCULATE:
C
  360   TLAT=SAVEP(1,1)*C0
        VELE=(-VELE*C0)*R1*COS(TLAT)*12
        VELN=VELN*C0*R1*12
        S2=SIN(TLAT)*SIN(TLAT)
        S3=SIN(2*TLAT)*SIN(2*TLAT)
        THGRAV=978032.-980000.+(C2*S2)-(C3*S3)
        EOTV=(C4*VELE*COS(TLAT))+C5*(VELE*VELE+VELN*VELN)
        FRAIR=GOBS-THGRAV+EOTV
C
C****** ROUND RESULTS AND STORE:
C
        FDATA(3)=EOTV+.05
        FDATA(4)=FRAIR+.05
        FDATA(5)=SAVEP(1,1)+.00005
        FDATA(6)=SAVEP(2,1)+.00005
        FDATA(7)=SAVEP(3,1)+.00005
        WRITE (4,384) DAY,FDATA(1),980000.+FDATA(2),(FDATA(J),J=3,7)
        WRITE (10,384) DAY,FDATA(1),980000.+FDATA(2),(FDATA(J),J=3,7)
        GO TO 323
C
C
C****** END UP:
C
  381   FORMAT('1','DAY',6X,'TIME',5X,'GOBS',5X,'EOTVOS',5X,'FRAIR',
     1  4X,'LAT',6X,'LONG',5X,'DEPTH')
  384   FORMAT(' ',F4.0,5X,F5.0,1X,F9.1,1X,F8.1,2X,F7.1,2X,F8.3,1X,
     1    F9.3,1X,F7.1)
  385   CONTINUE
C
```

```
      CLOSE (UNIT=8,DISPOSE='SAVE')
      CLOSE (UNIT=9,DISPOSE='SAVE')
      CLOSE (UNIT=10,DISPOSE='SAVE')
      WRITE (4,9922)
      STOP
      END
C*****************************************
      SUBROUTINE TIMER(TIME1,TIME2,DAY)
C*****************************************
C
C****** SPLIT TIMES TO HOURS, MINUTES, SECONDS AND KEEP SIGNS.
C
      TH1=AINT(TIME1/10000.)
      TM1=AINT((TIME1-TH1*10000.)/100.)
      TS1=TIME1-(TH1*10000.+TM1*100.)
      TH2=AINT(TIME2/10000.)
      TM2=AINT((TIME2-TH2*10000.)/100.)
      TS2=TIME2-(TH2*10000.+TM2*100.)
C
      AMNS=0.
      AHRS=0.
C
C****** RESOLVE SECONDS TO MINUTES AND SECONDS.
C
      TS1=TS1+TS2
      IF(TS1.GE.0.)GO TO 10
      TS1=TS1+60.
      TM2=TM2-1.
   10 IF(TS1.LT.60.) GO TO 20
      AMNS=AINT(TS1/60.)
      TS1=TS1-AMNS*60.
C
C****** RESOLVE MINUTES TO HOURS AND MINUTES.
C
   20 TM1=TM1+AMNS+TM2
      IF(TM1.GE.0.) GO TO 30
      TM1=TM1+60.
      TH2=TH2-1.
   30 IF(TM1.LT.60.) GO TO 40
      AHRS=AINT(TM1/60.)
      TM1=TM1-AHRS*60.
C
C****** RESOLVE HOURS AND ADJUST DAY.
C
   40 TH1=TH1+AHRS+TH2
      IF(TH1.LT.24.) GO TO 50
      TH1=TH1-24.
      DAY=DAY+1.
   50 IF(TH1.GE.0.) GO TO 60
      TH1=TH1+24.
      DAY=DAY-1.
C
C****** BUILD NEW TIME AND RETURN IT IN TIME1.
C
   60 TIME1=TH1*10000.+TM1*100.+TS1
C
C****** IF NEGATIVE DAY SENT IN SEND BACK TOTAL SECONDS.
C
      IF(DAY.GE.0.) RETURN
      TIME1=TH1*3600.+TM1*60.+TS1
      RETURN
      END
C*****************************************
```

```
      SUBROUTINE POSN(DAY,CTIM,FDAY,FTIME,FLATD,FLOND,DEPTH,IEOF)
C*****************************************
      DATA DTIME/0./
C
      IEOF=0
   10 CONTINUE
C++++++ INSERT READ FOR NAVIGATION HERE.
      IF(DTIME.GE.CTIM) GO TO 52
      READ(9,*,END=200) IDAY,ITIME,LAT,DLATM,LON,DLONM,DEPT,DUM
C
   20 CONTINUE
      DDAY=IDAY
      DTIME=ITIME
      DTIME=DTIME*100.
      DLAT=LAT
      DLON=LON
C
C++++++ IS THIS THE RIGHT DAY?
C
      IF(DDAY.LT.DAY) GO TO 10
      IF(DDAY.EQ.DAY) GO TO 40
C
C++++++ MUST BE DONE.
C
      IEOF=2
      GO TO 201
C
C++++++ CHECK THE TIME.
C
   40 CONTINUE
      IF(DTIME.GE.CTIM) GO TO 52
      PTIM=DTIME
      PLAT=DLAT
      PLATM=DLATM
      PLON=DLON
      PLONM=DLONM
      PDEPT=DEPT
      GO TO 10
C
C++++++ NEEED MORE POSITIONS.
C
  200 IEOF=2
  201 FTIME=PTIM
      RETURN
C
C++++++ GOT A HIT.
   52 CONTINUE
      IF(DTIME.EQ.CTIM) GO TO 60
C
      CNTIM=CTIM
      FNTIM=DTIME
      CALL TIMER(CNTIM,-PTIM,-5.)
      CALL TIMER(FNTIM,-PTIM,-5.)
      FACT=CNTIM/FNTIM
      FLAT=PLAT+(DLAT-PLAT)*FACT
      FLATM=PLATM+(DLATM-PLATM)*FACT
      FLON=PLON+(DLON-PLON)*FACT
      FLONM=PLONM+(DLONM-PLONM)*FACT
      DEPTH=PDEPT+(DEPT-PDEPT)*FACT
      FDAY=DAY
      FTIME=CTIM
C
      WRITE(4,900) FTIME
```

```
  900   FORMAT(' ','FIX INTERPOLATED AT :',F8.0)
        GO TO 65
C
   60   CONTINUE
        PTIM=DTIME
        PLAT=DLAT
        PLATM=DLATM
        PLON=DLON
        PLONM=DLONM
        PDEPT=DEPT
C
        FDAY=DDAY
        FTIME=DTIME
        FLAT=DLAT
        FLATM=DLATM
        FLON=DLON
        FLONM=DLONM
        DEPTH=DEPT
C
   65   CONTINUE
        FLATD=(FLAT+FLATM/60.)
        FLOND=(FLON+FLONM/60.)
C
C
        RETURN
        END
```

PROGRAM CCP42

translated to FORTRAN by

L.R. Dumontier

NORPAK Ltd.

Pakenham, Ont.

30 June 1980

CCP42

L.R. Dumontier

NORPAK LTD.

Purpose: To compute cross-correlation coefficients between GOBS and platform monitors.

To Execute: Load mag tape unit

Run CCP42

Input: Mag Tape

Output: file containing correlation matrix for future summation with other similar martrices

Hard copy

Reference:

LaCoste, L.B.J, 1973. Crosscorrelation method for evaluating and correcting shipboard gravity data: Geophysics, v. 38, pp 701-709.

Note:

Subroutine RECRD constitutes an IEEE-488 bus handler. Normal FORTRAN I/O statements may be substituted as desired.

```
C   ******************************************************
C   *                                                    *
C   *                                                    *
C   *   GRAVITY DATA CROSS-CORRELATION PROGRAM           *
C   *                                                    *
C   *   DATE:   14-JUN-76                                *
C   *                                                    *
C   *   AUTHOR: L. R. DUMONTIER, NORPAK LTD              *
C   *                                                    *
C   *                                                    *
C   ******************************************************
C
C
C
C       COMPILATION AND LOADING PROCEDURES FOR THE
C       CROSS-CORRELATION PROGRAM
C
C
C
C       FORT CCP         ;COMPILE CROSSCORRELATION PROGRAM
C
C       FORT NFORT       ;COMPILE TIME TO INTERVAL CONVERSION
C                        ;PROGRAM
C
C       FORT FINT        ;COMPILE BASIC INT FUNCTION SIMULATOR
C
C       FORT TAB         ;COMPILE TABBING ROUTINE
C
C
C       RLDR CCP DIGDA42 NFORT FINT TAB FORT.LB
C
C                        ;LOAD CCP WHERE DIGDA42 IS FORTRAN
C                        ;CALLABLE MAGTAPE ROUTINE AS SUPPLIED
C                        ;BY R. BEACH (EMR)
C
C
C       THREE VERSION OF THIS PROGRAM CAN BE GENERATED WITH
C       SLIGHT MODIFICATIONS TO THE SOURCE CODE. THE VARIATIONS
C       ARE INDICATED IN THE CODE WHERE APPLICABLE. TO AID IN
C       SORTING OUT THE DIFFERENCES, THE REGIONS OF INTEREST ARE
C       DELIMITED THUS:
C
C-------------------------
C
C       RDOS
C
C-------------------------
C
C
C
C.........................
C
C       SOS WITH DIGIDATA
C
C.........................
C
C
C
C+++++++++++++++++++++++++
C
```

```
C       RDOS READING DATA FROM DISC
C
C++++++++++++++++++++++++
C
C
C:::::::::::::::::::::::
C
C       RDOS READING DATA FROM DIGIDATA
C
C:::::::::::::::::::::::
C
C
C
C
C
C
C   ************************************************
C
C
C
C
C
C       TO RUN PROGRAM, SIMPLY TYPE CCP (CARRIAGE RETURN)
C
C
C       MAKE SURE GRAVITY DATA TAPE IS LOADED ONTO DIGIDATA
C       READER BEFORE ANSWERING THE QUESTIONS
C
C       THE PROGRAM WILL ASK FOR THE VARIOUS RUN-TIME
C       PARAMETERS AND WILL PROCEED TO PROCESS THE RAW
C       DATA.  SAMPLE OUTPUT IS INCLUDED IN THE PACKAGE.
C
C       THE PROGRESS OF THE PROGRAM IS INDICATED BY TYPING
C       THE TIME PERIODICALLY.
C
C       AFTER THE SYSTEMATIC CORRECTION HAS BEEN COMPUTED,
C       THE PROGRAM WILL TYPE A NUMBER OF USER OPTIONS.
C       THE USER SIMPLY TYPES THE NUMBER CORRESPONDING TO THE
C       SELECTED OPTION FOLLOWED BY CARRIAGE RETURN. THE
C       PROGRAM GUIDES THE USER ALONG, ASKING FOR PARAMETERS
C       AS THEY ARE REQUIRED. WHEN THE USER IS SATISFIED
C       THAT THE DEFECTIVE MONITOR HAS BEEN ISOLATED, THE CORRECTED
C       GRAVITY IS PLOTTED ALONG WITH THE UNCORRECTED GRAVITY.
C
C       THE PROGRAM WILL STAY IN AN INFINITE LOOP UNTIL ^A IS TYPED
C
C
C
C
C   ************************************************************
C
C
      REAL K1,K2,N0,N9,N,M,M1,M2,L
C
      INTEGER UDIM,TTO,TTI,F2,G2
C
      DIMENSION IFILE(5),JFILE(5)
      DIMENSION C(11),D1(8),P(3),Q(3),R(3),S(8)
      DIMENSION D(10,9),A(30),U(11,12),V1(8,8)
      DIMENSION M(8,8),M1(8,6),M2(8,8),Y(8,8),V(8,8)
      DIMENSION L(8),O(8),FILTER(60)
      DIMENSION H(10),U2(8,8),W(7,7),U1(8,8),B(3,67)
C
```

```
      COMMON/CCP1/TTI,TTO,N8,UDIM,FILTER
      DATA TTI/5/,TTO/7/,N8/0/,UDIM/12/
C
C     FILTER WEIGHING FUNCTION TAKES INTO ACCOUNT
C     PREVIOUS FILTERING OF 6 20-SEC STAGES
C
C     FILTER TIME LAG = 7 MIN.
C
      DATA FILTER/-.008,-.00086,-.00089,-.0009,-.00086,
     *  -.00074,-.00054,.00021,.00027,.00093,
     *  .00183,.003,.00448,.00629,.00848,
     *  .01106,.01403,.01735,.02099,.02488,
     *  .02893,.03298,.03694,.04063,.04391,
     *  .04666,.04875,.05011,.05068,.05048,
     *  .04951,.04785,.04558,.04282,.03966,
     *  .03625,.0327,.02911,.02557,.02216,
     *  .01897,.01601,.01332,.01092,.00882,
     *  .00699,.00544,.00412,.00304,.00216,
     *  .00146,.0009,.00048,.00016,-.00007,
     *  -.00022,-.00033,-.00039,-.00042,-.00042/
C
C     ALLOW USER TO INITIALIZE VARIABLE PARAMETERS
C
C
C     PROCESS RAW DATA FIRST TIME THRU
C
C
C     ZERO Y ONLY FIRST TIME THRU
C
C     CHOOSE OPTION OF PROCESSING NEW DATA OR
C     COMBINING TAPES FROM PREVIOUS REDUCTIONS
C
C
2     WRITE(TTO,1010)
1010  FORMAT(//,' ENTER OPTION:',/,T15,' 1 - PROCESS RAW DATA',/,T15,
     *  ' 5 - PREPARE FOR ENTERING Y(I,J) FROM PAPER TAPE')
      ACCEPT *, G2
      GO TO (10,2,2,2,720,2,2) G2
C
10    DO 5 I=1,8
        DO 4 J=1,8
          Y(I,J)=0.
4       CONTINUE
5     CONTINUE
      G2=0
C
1     WRITE(TTO,1000)
1000  FORMAT(/' ENTER REEL NUMBER,DAY,START TIME(HHMMSS.), END TIME')
      ACCEPT *, IREEL,IDAY,T0,T9
C
      WRITE(TTO,1020)
1020  FORMAT(/' ENTER SAMPLING TIME,...(SEC.)')
      ACCEPT *, T4
C
C     COUNTER FOR TYPEOUT OF T
C
      TOLD=0.
C
C     DATA COUNTER FOR COMPUTING AVERAGES
C
      F2=0
C
C     **** ALLOW USER TO CHANGE LATER...
```

```
C
C       INITIALIZE ALLOWABLE CURVATURES
C
        DO 20 I=1,10
          C(I)=32.
20      CONTINUE
        C(6)=128.
C
C       INITIALIZE U-MATRIX TO IMPOSSIBLE VALUES
C       TO ALLOW DETECTION OF MISSING DATA
C
        DO 50 I=1,11
          DO 40 J=1,UDIM
            U(I,J)=0.
40        CONTINUE
50      CONTINUE
C
C       SET DATA VALIDITY INDICATORS TO ZERO
C
        DO 55 I=1,10
          DO 54 J=1,9
            D(I,J)=0.
54        CONTINUE
55      CONTINUE
C
C       INITIALIZE Y,L,O,M & S ARRAYS TO ZERO
C
        DO 65 I=1,8
          DO 60 J=1,8
          M(I,J)=0.
60        CONTINUE
65      CONTINUE
        DO 70 I=1,8
          L(I)=0.
          O(I)=0.
          S(I)=0.
70      CONTINUE
C
C       OPEN DATA FILE TO ALLOW COMPUTING
C       OF AVERAGE SYSTEMATIC CORRECTION
C
C
C+++++++++++++++++++++++++
C
C       CALL OPEN(2,'TAPE',1,IERR)
C
C+++++++++++++++++++++++++
C
C-----------------------
C
C       CALL OPEN(1,'ASYSC',3,IERR)
C
C----------------------
C
C       NUMBER INTERVALS CORRESPONDING TO THE VARIOUS T'S
C
        CALL NFORT(T0,N0,T4)
        CALL NFORT(T9,N9,T4)
C
C       IF JUST ACROSS 24 HR BOUNDARY,
C       MAKE SURE N9>N0
C
        IF (N9.GT.N0) GO TO 85
```

```
        N9=N9+86400./T4
C
C       WRITE N0,N9 FOR USER'S INFORMATION..
C
85      WRITE(TTO,1030) N0,N9
1030    FORMAT(' N0= ',F6.0,'    N9= ',F6.0)
C
C       SKIP FIRST 7 DATA SETS (AS PER BASIC PROGRAM)
C
        N0=N0+6
C
C       READ MAGTAPE DATA UNTIL T=T0
C
C       IF IER=-1, MAGTAPE UNIT WASN'T READY...
C       SIMPLY TRY AGAIN
C
C
C.........................
C:::::::::::::::::::::::::
C
90      CONTINUE
        CALL RECRD (0,A)
        IF(T.EQ.A(2)) GO TO 90
C
C:::::::::::::::::::::::::
C.........................
C
C+++++++++++++++++++++++++
C
C90     READ BINARY(2) (A(I),I=1,21)
C
C+++++++++++++++++++++++++
C
        T=A(2)
C
C       SUBROUTINE NFORT CONVERTS TIME TO THE NUMBER
C       OF TIME INTERVALS (AS DEFINED BY SAMPLING TIME)
C       SINCE THE BEGINNING OF THE DAY
C
        CALL NFORT(T,N,T4)
C
C       IGNORE DATA IF THERE ARE LARGE GAPS
C       IN TIME SEQUENCE
C
        IF(N.LT.N0) GO TO 90
        IF(N.GT.(N0+20)) GO TO 90
C
C       SET BLOCK COUNTER FOR MODULO 30 TOTAL DATA SET
C
        N30=1
C
C       JUMP INTO MIDDLE OF MAIN DATA LOOP
C       SINCE FIRST SET OF DATA AVAILABLE
C
        GO TO 120
C
C
C       MAIN DATA LOOP STARTS HERE
C
C       SHIFT WHOLE BUFFER FOR EVERY NEW SET OF DATA
C
110     DO 112 I=1,11
          ITEMP=UDIM-1
```

```
          DO 111 J=1,ITEMP
            U(I,UDIM-J+1)=U(I,UDIM-J)
111       CONTINUE
          U(I,1)=999999.
112     CONTINUE
C
C       & SHIFT DATA VALIDITY INDICATORS TO FOLLOW U MATRIX...
C
        DO 115 I=1,10
          DO 114 J=1,8
            D(I,10-J)=D(I,9-J)
114       CONTINUE
          D(I,1)=0.
115     CONTINUE
C
C       READ NEXT SET OF DATA
C
C
C:::::::::::::::::::::::::::
C...........................
C
117     CONTINUE
        CALL RECRD (0,A)
        IF(T.EQ.A(2)) GO TO 117
C
C...........................
C:::::::::::::::::::::::::::
C
(+++++++++++++++++++++++++++
C
C       READ BINARY(2) (A(I),I=1,21)
C
C+++++++++++++++++++++++++++
C
        T=A(2)
        CALL NFORT(T,N,T4)
C
C       CHECK FOR 24 HR BOUNDARY
C
120     N=N-N0
125     IF(N.GE.0.) GO TO 130
        N=N+86400./T4
        GO TO 125
C
C
C       REMOVE CHANNEL NUMBERS
C
C       CHANNEL NUMBERS STORED IN FIFTH DECIMAL DIGIT
C       FROM THE RIGHT
C
130     DO 135 I=6,14
          A(I)=A(I)-AINT(A(I)/10000.)*10000.
135     CONTINUE
C
C       MOVE DATA INTO U MATRIX
C
C       INCOMING DATA IS FORMATTED AS FOLLOWS:
C
C       A(2)    TIME
C       A(4)    SPRING TENSION
C       A(5)    AVERAGE BEAM VOLTAGE
C       A(6)    CROSS COUPLING
C       A(8)    INHERENT CROSS-COUPLING
```

```
C       A(9)    LONGITUDINAL IMPERFECTION CROSS COUPLING
C       A(10)   CROSS IMPERFECTION CROSS COUPLING
C       A(11)   AVERAGE VALUE OF SQUARE OF VERTICAL ACCELERATION
C       A(12)   AVERAGE VALUE OF ABSOLUTE MAGNITUDE OF CROSS
C                             ACCELERATION
C       A(13)   AVERAGE VALUE OF ABSOLUTE MAGNITUDE OF
C                             LONGITUDINAL ACCELERATION
C       A(14)   SECOND ORDER CROSS IMPERFECTION CROSS COUPLING
C
C       DATA WILL BE STORED IN THE FOLLOWING ORDER:
C
C       U(1,X)  S.T.
C       U(2,X)  B
C       U(3,X)  CC
C       U(4,X)  VCC
C       U(5,X)  AL
C       U(6,X)  AX
C       U(7,X)  VE
C       U(8,X)  X..
C       U(9,X)  Y..
C       U(10,X) AX2
C       U(11,X) T
C
        U(1,1)=A(4)
        U(2,1)=A(5)
        U(3,1)=A(6)
        DO 140 I=4,10
          U(I,1)=A(I+4)
140     CONTINUE
        U(11,1)=A(2)
C
C       NOW CHECK DATA FOR IMPOSSIBLE VALUES
C       PROBABLY CAUSED BY TAPE ERRORS
C
        DO 150 I=1,10
          IF(I.EQ.1) GO TO 145
          IF(ABS(U(I,1)).LT.9999.) GO TO 150
          D(I,1)=-10.
          GO TO 150
145       IF(ABS(U(I,1)).LT.999998.) GO TO 150
          D(I,1)=-10.
150     CONTINUE
C
C       CAN'T DO ANYMORE UNLESS 3 SETS AVAILABLE
C
        IF(N.LT.1.9) GO TO 110
C
C       CHECK ALLOWABLE CURVATURES
C
        DO 160 I=1,10
          C(11)=U(I,1)-2*U(I,2)+U(I,3)
          IF(ABS(C(11)).GT.C(I)) GO TO 160
C
C         CURVATURE OK, SET DATA VALID FLAGS
C
          D(I,3)=D(I,3)+1.
          D(I,2)=D(I,2)+1.
          D(I,1)=D(I,1)+1.
160     CONTINUE
C
C       CAN'T INTERPOLATE DATA UNLESS 9 SETS AVAILABLE
C
        IF(N.LT.7.9) GO TO 110
```

```
C
C       INTERPOLATE WHERE REQUIRED/POSSIBLE
C
        DO 200 I=1,10
          IF(D(I,8).GT.0.5) GO TO 200
C
C         TYPE OUT DATA CAUSING PROBLEMS
C
          TEMP=N-5+NO
          WRITE(TTO,1050)TEMP,U(I,9),U(I,8),
     *        U(I,7),U(I,6),U(I,5)
1050      FORMAT(/' N= ',F6.0,' INVALID DATA IS: ',5F10.3)
C
C         CHECK ADJACENT DATA VALIDITY FLAGS TO SEE IF
C         INTERPOLATION IS POSSIBLE...
C
C         IF PREVIOUSLY INTERPOLATED DATA INVALID,
C         OR MORE THAN FIVE POINTS IN SUCCESSION INVALID,
C         INTERPOLATION NOT POSSIBLE.
C
          IF(D(I,9).LE.0.5) GO TO 170
C
          K=1
          IF(D(I,3).GT.0.5) K=2
          IF(D(I,4).GT.0.5) K=3
          IF(D(I,5).GT.0.5) K=4
          IF(D(I,6).GT.0.5) K=5
          IF(D(I,7).GT.0.5) K=6
C
          IF(K.NE.1) GO TO 180
170       WRITE(TTO,1060) I, TEMP
1060      FORMAT(/' TOO MANY ERRORS TO INTERPOLATE VARIABLE '
     *        ,I2,' AT N= ',F6.0)
C
C         TO ALLOW PROGRAM TO CONTINUE
C         SET DATA AS VALID
C
          D(I,8)=1.
          GO TO 200
C
C         FORM OF INTERPOLATION DEPENDS ON NUMBER OF
C         POINTS MISSING.
C
180       U(I,8)=(U(I,9)*(7-K)+U(I,K+1))/(8-K)
        WRITE(TTO,1065) U(I,8)
1065    FORMAT(' INTERPOLATED VALUE = ',F12.4)
          D(I,8)=1.
C
200     CONTINUE
C
C       NOW COMPUTE GRAVITY FROM SPRING TENSION AND BEAM.
C
C       FILTER ALL MONITORS...
C
C       NEGATIVE OF BEAM SLOPE CONSTANT (MV/MGAL)
C       FROM CALIBRATION INFORMATION SHEET
C
        F=2.
        F=60./F
C
C       MAKE SURE NO INVALID DATA REMAINS
C
C
```

```
C
      DO 208 I=1,11
        IF(ABS(U(I,12)-U(I,11)).LT.1000.) GO TO 208
        IF(ABS(U(I,12)-U(I,10)).GT.1000.
     *    .AND.ABS(U(I,12)-U(I,9)).GT.1000.) GO TO 208
        IF(ABS(U(I,12)-U(I,10)).GT.1000.) GO TO 204
        U(I,11)=(U(I,12)+U(I,10))/2.
        GO TO 208
204     U(I,11)=(U(I,12)*2.+U(I,9))/3.
208   CONTINUE
C
C     COMPUTE AVERAGE OF LAST TWO SPRING TENSION READINGS
C
      S(2)=(U(1,12)+U(1,11))/2.
      IF(N.GT.11.1) GO TO 220
C
C     SET UP INITIAL CONDITIONS FOR FILTERING
C
      DO 210 J=1,7,2
        S(J)=S(2)
210   CONTINUE
C
C     FILTER SPRING TENSION WITH 3 20-SEC STAGES OF RC LOW-PASS
C     FILTERING.
C
220   DO 230 J=1,5,2
        S(J+3)=S(J+2)*.6+(S(J+1)+S(J))/5.
230   CONTINUE
C
C     SHIFT DATA
C
      DO 240 J=1,7,2
        S(J)=S(J+1)
240   CONTINUE
C
C     COMPUTE GRAVITY
C
      M(1,2)=S(8)-(U(2,11)-U(2,12))*F+(U(3,12)+U(3,11))/2
C
C     & MONITORS
C
      DO 250 I=2,8
        M(I,2)=(U(I+2,12)+U(I+2,11))/2
250   CONTINUE
C
C     FILTER GRAVITY AND MONITORS WITH THREE MORE 20-SEC
C     STAGES OF RC LOW-PASS FILTERING
C
      DO 300 I=1,8
        IF(N.GT.11.1) GO TO 270
        DO 260 J=1,7,2
          M(I,J)=M(I,2)
260     CONTINUE
270     DO 280 J=1,5,2
          M(I,J+3)=M(I,J+2)*.6+(M(I,J+1)+M(I,J))/5
280     CONTINUE
        DO 290 J=1,7,2
          M(I,J)=M(I,J+1)
290     CONTINUE
300   CONTINUE
C
      IF(N.LT.11.1) GO TO 110
C
```

```
C       PARAMETERS FOR GETTING CURVATURES AND FILTERING
C
        P(1)=(1.-T4/300.)/(1.+T4/300.)
        Q(1)=1./(1.+T4/300.)
C
C       NOW MOVE M TO M1 TO ISOLATE
C       VARIOUS STAGES OF COMPUTATION
C
        M1(1,2)=M(1,8)/10
        DO 310 I=2,8
          M1(I,2)=-M(I,8)/10
310     CONTINUE
C
C       WRITE DATA TO ALLOW COMPUTING OF AVERAGE
C       SYSTEMATIC CORRECTION
C
C-------------------------
C
C
C       WRITE BINARY (1) (M1(I,2),I=1,8),U(11,11)
C
C-------------------------
C
C       & CHECK FUNCTION TO BE PERFORMED
C
        F2=F2+1
        IF(G2.GT.0) GO TO 600
C
C       PROCESS RAW DATA
C
        IF(N.GT.12.1) GO TO 330
C
C       INITIAL CONDITIONS
C
C       FIRST RECORD TIME CORRESP. TO FIRST VALID DATA ITEM
C
        T1=U(11,11)-T4
        DO 320 I=1,8
          M1(I,1)=M1(I,2)
          M1(I,3)=0.
          M1(I,5)=0.
320     CONTINUE
C
C       GET CURVATURES OF GRAVITY AND MONITORS AND FILTER THEM
C
330     DO 360 I=1,8
          DO 340 K=1,3,2
            M1(I,K+3)=M1(I,K+2)*P(1)+(M1(I,K+1)-M1(I,K))*Q(1)
340       CONTINUE
          DO 350 K=1,5,2
            M1(I,K)=M1(I,K+1)
350       CONTINUE
360     CONTINUE
C
C       COMPUTE CROSS-CORRELATION INTEGRALS
C
        DO 380 I=1,8
          DO 370 K=1,8
            Y(I,K)=Y(I,K)+M1(I,6)*M1(K,6)
370       CONTINUE
380     CONTINUE
C
C       COMPUTE SUMS AND SUMS OF PRODUCTS
```

```
C
        DO 390 I=2,8
          L(I)=L(I)+M1(I,2)*M1(I,2)
          O(I)=O(I)+M1(I,2)
390     CONTINUE
C
C       CHECK IF ALL DATA PROCESSED
C
C       GET MORE DATA IF T<T9
C
        N30=N30+1
C
C
C       CHECK FOR LAST BLOCK - IT CONTAINS ONLY 26 DATA ITEMS
C
        IF(N30.NE.26) GO TO 395
C
C       TOO CLOSE TO END?
C
        IF((N9-(N+N0)).LE.34) GO TO 396
395     IF(N30.NE.31) GO TO 110
        WRITE(TTO,1040) U(11,11)
1040    FORMAT(1X,'T=',F10.0)
        N30=1
        GO TO 110
C
C
C       YES...
C
396     F2=F2+1
        WRITE(TTO,1191) F2
1191    FORMAT(1X,'NUMBER OF DATA BLOCKS='I5)
C
C       RECORD TIME OF LAST DATA ITEM
C
        T2=U(11,11)-T4
        N5=F2
        N9=F2
C
C++++++++++++++++++++
C
C       ENDFILE 2
C
C++++++++++++++++++++
C
C---------------------
C
C       ENDFILE 1
C
C---------------------
C
C       ELIMINATE ANY Y(I,I)=0 WHICH WOULD
C       INTERFERE WITH COMPUTATIONS
C
        DO 400 I=1,8
          IF(Y(I,I).LT.0.9) Y(I,I)=1
400     CONTINUE
C
C       SET V(I,K)=Y(I,K) FOR PRINTING ROUTINE
C
405     N1=0
        N2=7
C
```

```
      DO 420 I=1,8
        DO 410 K=1,8
          V(I,K)=FINT(Y(I,K)+0.5)
          V1(I,K)=FINT(V(I,K)*1000.+0.5)/1000.
410     CONTINUE
420   CONTINUE
C
C     START PRINTING RESULTS
C
      WRITE(TTO,1070) IREEL
1070  FORMAT(/////,11X,'CROSS CORRELATION CANADA, REEL # ',I4,//)
C
C     SKIP SOME PRINTING DEPENDING ON G2
C
      IF(G2.EQ.1) GO TO 435
      WRITE(TTO,1080) T1,T2
1080  FORMAT(21X,'TIME: ',F7.0,' TO ',F7.0)
      IF(G2.EQ.3) GO TO 435
C
C     COMPUTE AVERAGE AND RMS VALUES OF MONITORS
C
      DO 430  I=2,8
        O(I)=O(I)/F2
        L(I)=SQRT(L(I)/F2)
        L(I)=FINT(L(I)+0.5)
C
C       SET D1(I)=O(I) FOR PRINTING ROUTINE
C
        D1(I)=FINT(O(I)+0.5)
430   CONTINUE
      WRITE(TTO,1090)
1090  FORMAT(/' AVERAGE AND RMS VALUES OF MONITORS: '/)
      WRITE(TTO,1091)
1091  FORMAT(T3,'VCC',T11,'AL',T20,'AX',T29,'VE',
     * T37,'X"',T46,'Y"',T56,'AX2'//)
      WRITE(TTO,1095) (D1(I),I=2,8)
1095  FORMAT(7(F5.0,4X)/)
      WRITE(TTO,1095) (L(I),I=2,8)
435   WRITE(TTO,1100)
1100  FORMAT(//,' CROSS CORRELATION INTEGRALS: '/)
      WRITE(TTO,1101)
1101  FORMAT(T3,'G',T10,'VCC',T18,'AL',T26,'AX',
     * T34,'VE',T42,'X"',T50,'Y"',T58,'AX2',//)
      WRITE(TTO,1105)((V1(I,J),J=1,8),I=1,8)
1105  FORMAT(8(F7.0,1X))
C
C     COMPUTE NORMALIZED CROSS-CORRELATION INTEGRALS
C
      DO 450 I=1,8
        DO 440 J=1,I
          V(I,J)=SQRT(Y(I,I)*Y(J,J))
          V(I,J)=Y(I,J)/V(I,J)
          V1(I,J)=FINT(V(I,J)*1000.+0.5)/1000.
          V1(J,I)=V1(I,J)
          V(J,I)=V(I,J)
440     CONTINUE
450   CONTINUE
      WRITE(TTO,1110)
1110  FORMAT(/,' NORMALIZED CROSS-CORRELATION INTEGRALS:'/)
      WRITE(TTO,1101)
      WRITE(TTO,1115)((V1(I,J),J=1,8),I=1,8)
1115  FORMAT(8(F7.3,1X))
      WRITE(TTO,1120)
```

```
1120    FORMAT(/' FIRST ROW OF ABOVE MATRIX ARE CROSS'
     *  ,' CORRELATION COEFFICIENTS'/)
C
C       SOLVE SIMULTANEOUS EQUATIONS CORRESPONDING TO
C       NORMALIZED CROSS CORRELATION INTEGRALS
C
C
        DO 550 NN=1,7
C
C       SHIFT SUBSCRIPTS TO SOLVE FOR EACH UNKNOWN
C
          DO 490 I=2,8
            K=I-NN+1
            IF(K.GT.1) GO TO 460
            K=K+7
460         U2(K,1)=V(I,1)
            U2(1,K)=U2(K,1)
            DO 480 J=2,8
              IG=J-NN+1
              IF(IG.GT.1) GO TO 470
              IG=IG+7
470           U2(K,IG)=V(I,J)
480         CONTINUE
490       CONTINUE
          DO 540 I1=1,6
            I=9-I1
            ITEMP=I-1
            DO 510 J=2,ITEMP
              DO 500 K=1,J
                W(J,K)=U2(J,K)*U2(I,I)-U2(I,J)*U2(I,K)
500           CONTINUE
510         CONTINUE
            ITEMP=I-1
            DO 530 J=2,ITEMP
              DO 520 K=1,J
                U2(J,K)=W(J,K)
520           CONTINUE
530         CONTINUE
540       CONTINUE
          H(NN+1)=W(2,1)/W(2,2)
550     CONTINUE
C
C       COMPUTE PERCENTAGE REDUCTION IN CURVATURE
C       OF COMPUTED GRAVITY BY USING A SINGLE MONITOR
C
        DO 560 I=2,8
          D1(I)=1.-SQRT(1.-V(1,I)**2)
          D1(I)=FINT(D1(I)*1000.+0.5)/10.
560     CONTINUE
C
C       PRINT RESULT
C
        WRITE(TTO,1130)
1130    FORMAT(/' PERCENTAGE REDUCTION IN CURVATURE OF ',
     *  'COMPUTED',/,' GRAVITY BY USING A SINGLE MONITOR:'/)
        WRITE(TTO,1091)
        WRITE(TTO,1135) (D1(I),I=2,8)
1135    FORMAT(7(F5.1,4X))
C
C       COMPUTE PERCENTAGE REDUCTION IN CURVATURE OF COMPUTED
C       GRAVITY BY USING ALL MONITORS
C
        H(1)=-1.
```

```
        G9=0
        DO 580 I=1,8
          DO 570 J=1,8
            G9=G9+H(I)*H(J)*V(I,J)
570       CONTINUE
580     CONTINUE
        G9=FINT((1.-SQRT(G9))*1000.+0.5)/10.
        WRITE(TTO,1140) G9
1140    FORMAT(//' PERCENTAGE REDUCTION IN COMPUTED GRAVITY ',
     *  'BY USING ALL MONITORS: ',F6.2,//,' FRACTIONS OF ',
     *  'MONITORS TO BE ADDED TO GET'/' ZERO CROSS-',
     *  'CORRELATION'/)
C
C       UNNORMALIZE H(I)
C
        DO 590 I=2,8
          D1(I)=SQRT(Y(1,1)/Y(I,I))
          H(I)=H(I)*D1(I)
          D1(I)=FINT(-H(I)*1000+0.5)/1000
590     CONTINUE
        WRITE(TTO,1091)
        WRITE(TTO,1145)(D1(I),I=2,8)
1145    FORMAT(7(F7.3,2X))
C
C       COMPUTE AVERAGE SYSTEMATIC CORRECTIONS ON SECOND
C       AND SUBSEQUENT PASSES
C
C
C       COMPUTE AVERAGE SYSTEMATIC CORRECTION
C
        G2=3
595     K1=0
        N1=0
        N2=7
C
C---------------------------
C
C       CALL OPEN(1,'ASYSC',1,IERR)
C
C---------------------------
C
        IF(G2.EQ.2) GO TO 599
C
C.........................
C
598     WRITE(TTO,1146)
1146    FORMAT(//' ENTER OPTION:'/,T15,'3 - COMPUTE AVERAGE SYSTEMA'
     *  'TIC CORRECTION (REWIND ',/T19,
     *  'DIGIDATA BEFORE CHOSING THIS OPTION!)',/,/T15,
     *'1 - LIST MORE OPTIONS')
        ACCEPT *, G2
        GO TO(690,598,1,598,598,598) G2
C
C.......................
C
599     WRITE(TTO,1148)
1148    FORMAT(//' SCALE: 1 INCH = 10 MGAL',//,T15,'X --- UNCOR',
     *  'RECTED GRAVITY',/,T15,'O --- CORRECTED GRAVITY',//,' TIME'//)
C
C.........................
C
        GO TO 1
C
```

```
C.......................
C
C
C-------------------------
C
C600    READ BINARY(1,END=680) (M2(I,2),I=1,8),T
C
C-------------------------
C
C.......................
C
C
C       COPY MATRIX TO SIMULATE DISK COPY
C
600     DO 605 I=1,8
          M2(I,2)=M1(I,2)
605     CONTINUE
        T=U(11,11)
C
C.......................
C
C       G4 IS UNCORRECTED GRAVITY (+4 TO SHIFT PLOT)
C
        G4=M2(1,2)
        G4=G4+4
        G3=G4
C
C       COMPUTE CORRECTED GRAVITY
C
        DO 610 I=2,8
        G3=G3-H(I)*M2(I,2)
610     CONTINUE
        K1=K1+G3-G4
C
C-------------------------
C
C       IF(G2.EQ.3) GO TO 600
C
C-------------------------
C
C.......................
C
        IF(F2.EQ.N5) GO TO 680
        IF(G2.EQ.3) GO TO 110
C
C.......................
C
C       G2=2: CORRECTED AND UNCORRECTED GRAVITY ARE FILTERED
C       AND TIME IS DELAYED
C
C
C       THE WEIGHTING TAKES INTO ACCOUNT 6, 20 SEC STAGES
C       OF PREVIOUS FILTERING
C
        B(1,N2)=G3
        B(2,N2)=G4
        B(3,N2)=T
C
C       FILTER ONCE A MINUTE
C
        IF(N2.LT.66) GO TO 670
        DO 620 I=1,60
C
```

```
C
C       MAKE SURE WE'RE FILTERING SMALL NUMBERS
C       TO AVOID INCORRECT NORMALIZATION
C       OF THE WEIGHTINGG FUNCTION
C
        B(1,I)=B(1,I+6)-B(1,66)
        B(2,I)=B(2,I+6)-B(2,66)
        B(3,I)=B(3,I+6)
620     CONTINUE
C
C       RESTORE PREVIOUS VALUES
C
        B(1,67)=B(1,66)
        B(2,67)=B(2,66)
        DO 630 I=1,60
          B(1,67)=B(1,67)+FILTER(I)*B(1,I)
        B(2,67)=B(2,67)+FILTER(I)*B(2,I)
630     CONTINUE
C
C       RESTORE ORIGINAL B(J,I)
C
        DO 640 I=1,60
          B(1,I)=B(1,I)+B(1,66)
        B(2,I)=B(2,I)+B(2,66)
640     CONTINUE
C
C       G3 FILTERED CORRECTED GRAVITY
C       G4 FILTERED UNCORRECTED GRAVITY
C
        N2=60
        G3=B(1,67)
        G4=B(2,67)
C
C       DELAY TIME =  7 MIN.
C
        T=B(3,18)
C
C       WRITE TIME PERIODICALLY
C
        IF(N1.LT.60) GO TO 650
        N1=0
        WRITE(TTO,1150) T
1150    FORMAT(1X,F8.0,1X,$)
C
C       PLOT THE DATA
C
650     G3=G3/60.-FINT(G3/60.+.000001)
        G4=G4/60.-FINT(G4/60.+.000001)
        G3=FINT(G3*60.+8.)
        G4=FINT(G4*60.+8.)
        IF(N1.NE.0) WRITE(TTO,1154)
1154    FORMAT(10X,$)
        WRITE(TTO,1155)
1155    FORMAT(' .',$)
        IF(G4.LT.G3) GO TO 660
        CALL TAB(G3)
        WRITE(TTO,1160)
1160    FORMAT(1X,'O',$)
        IF(G3.EQ.G4) GO TO 665
        CALL TAB(G4-G3)
        WRITE(TTO,1170)
1170    FORMAT(1X,'X',$)
        GO TO 665
```

```
660     CALL TAB(G4)
        WRITE(TTO,1170)
        CALL TAB(G3-G4)
        WRITE(TTO,1160)
665     WRITE(TTO,1171)
1171    FORMAT(2X)
670     N1=N1+1
        N2=N2+1
C
C-------------------
C
C       GO TO 600
C
C------------------
C
C
C.................
C
        GO TO 110
C
C.................
C
C
C       END OF FILE ON INPUT FILE
C
C
C----------------------
C
C680    ENDFILE 1
C
C----------------------
C
C.....................
C
680     N9=N5
C
C
C.....................
C
        IF(G2.NE.3) GO TO 690
        K2=FINT(10.*K1/(N9+1.)+0.5)/10.
        WRITE(TTO,1180)K2
1180    FORMAT(//' AVERAGE SYSTEMATIC CORRECTION= 'F5.2,' MGAL.')
        G2=2
C
C
C       DONE WITH AVERAGE SYSTEMATIC CORRECTION
C       WHAT IS NEXT?
C
690     WRITE(TTO,1200)
1200    FORMAT(//' ENTER OPTION:'/,T15,'2 - PLOT',/
     *  ,T15,'4 - PUNCH Y(I,J)',/,T15,'5 - PREPARE FOR ENTERING'
     *  ' Y(I,J) FROM PUNCHED TAPE',/,T15,'6 - MODIFY Y(I,J) FOR '
     *  'ONE MONITOR')
        ACCEPT *, G2
        GO TO (690,595,690,700,720,740,690,690,690), G2
C
C       PUNCH Y(I,J)
C
700     TYPE *, 'OUTPUT FILENAME ? (------.---)'
        ACCEPT 88, IFILE
        CALL ASSIGN (14,IFILE)
        DO 710 I=1,8
```

```
          DO 705 J=1,8
            WRITE(14,1205) Y(I,J)
1205        FORMAT(1X,F10.2)
705       CONTINUE
710     CONTINUE
        CALL CLOSE (14)
        GO TO 690
C
C       CLEAR Y(I,J) BEFORE ENTERING NEW DATA
C
720     DO 730 I=1,8
          DO 725 J=1,8
            Y(I,J)=0.
725       CONTINUE
730     CONTINUE
C
C       ASK FOR NEW OPTION...
C
735     WRITE (TTO,1210)
1210    FORMAT(//' ENTER OPTION:'/,T15,'1 - PROCESS DATA'/
     *  ,T15,'6 - MODIFY Y(I,J) FOR ONE MONITOR'/,T15,
     *  '7 - ADD TO Y(I,J) FROM PAPER TAPE')
        ACCEPT *, G2
        GO TO (780,735,735,735,735,740,760,735,735),G2
C
C       MODIFY Y(I,J)
C
740     WRITE(TTO,1220)
1220    FORMAT(//' ENTER MONITOR NUMBER:'/,T15,'2 - VCC',/
     *  ,T15,'3 - AL',/T15,'4 - AX',/,T15,'5 - VE'
     *  ,/,T15,'6 - X'',/,T15,'7 - Y'',/,T15,'8 - AX2')
        ACCEPT *, K
C
C       ZERO ALL MONITORS EXCEPT THE ONE SELECTED BY USER
C
        DO 750 I=1,8
          DO 745 J=1,I
            IF(I.EQ.J) GO TO 745
            IF(I.EQ.K.AND.J.EQ.1) GO TO 745
            Y(I,J)=0.
            Y(J,I)=0.
745       CONTINUE
750     CONTINUE
        GO TO 735
C
C       READ Y(I,J) FROM PAPER TAPE
C
760     TYPE *, 'INPUT FILENAME ? (------.---)'
        ACCEPT 88, JFILE
88      FORMAT (5A2)
        CALL ASSIGN (13,JFILE)
        DO 775 I=1,8
          DO 770 J=1,8
            READ(13,1225) Y1
1225        FORMAT(F10.2)
            Y(I,J)=Y(I,J)+Y1
770       CONTINUE
775     CONTINUE
        CALL CLOSE (13)
        GO TO 735
C
C       PROCESS DATA
C
```

```
780     WRITE(TTO,1230)
1230    FORMAT(//' ARE DATA AVAILABLE FOR SYSTEMATIC CORRECTION?',/
     *  T15,'1 - NO',/,T15,'3 - YES')
        ACCEPT *, G2
        GO TO 405
C
        END
C
C       SUBROUTINE NFORT COMPUTES THE N CORRESPONDING TO T
C       WHERE :
C
C       T=TIME
C       N=NUMBER OF TIME INTERVALS SINCE START OF DAY
C       T4=SAMPLING TIME
C
        SUBROUTINE NFORT(T,N,T4)
C
        REAL N
C
        T1=AINT(T/10000+.01)
        T2=T/10000-T1
        T2=AINT(T2*100+.01)
        T3=T/10-T1*1000-T2*10
        T3=AINT(T3+.01)
        N=AINT(T1*3600/T4+T2*60/T4+T3*10/T4+.01)
        RETURN
        END
C
C       FUNCTION TO SIMULATE THE
C       BASIC FUNCTION 'INT'
C
C       NOTE THAT THE BASIC INT TRUNCATES NEGATIVE NUMBERS
C       DIFFERENTLY THAN THE FORTRAN AINT
C
C       EG IN BASIC, INT(-13.5) = -14
C          IN FORT,  AINT(-13.5) = -13
C
C
        REAL FUNCTION FINT(ARG)
C
        IF(ARG.LT.0.) ARG=ARG-1.
        FINT=AINT(ARG)
        RETURN
        END

C       TABBING ROUTINE FOR PLOTTING GRAVITY DATA
C
C       THIS ROUTINE IS USED TO PLOT CORRECTED AND
C       UNCORRECTED GRAVITY ON THE EXECUPORT TERMINAL
C
        SUBROUTINE TAB(XN)
C
        N=AINT(XN)
        DO 10 I=1,N
        WRITE(7,1000)
1000    FORMAT(1X,' ',$)
10      CONTINUE
        RETURN
        END
```