



Technical Note 59
July 1990

Ocean Simulation Model for Internal Waves Computer Source Code

AD-A225 110

D. Rubenstein
Science Applications International Corporation
McLean, Virginia

DTIC
ELECTE
AUG 09 1990

Approved for public release; distribution is unlimited. Naval Oceanographic and Atmospheric Research Laboratory, Stennis Space Center, Mississippi 39529-5004.

**These working papers were prepared for the timely dissemination of information;
this document does not represent the official position of NOARL.**

Abstract

This technical note contains the source code for the first level ocean simulation model and associated test and display programs. This model provides simulations of internal wave activity based on average oceanographic conditions at a given location. The code is written in FORTRAN 77 and should be easily ported to a wide variety of computers and operating systems. This technical note is intended primarily for persons implementing and/or modifying the code on their own systems.

Acknowledgments

This work was supported under Program Element 602435N, under the direction of CDR Lee Bounds, office of Naval Technology.

The authors gratefully acknowledge Drs. Albert W. Green, Melvin Briscoe, Rudy Hollman, Lewis Dozier, Steven Borchardt, Michael Gregg, Terry Ewart, Chris Garrett and Eric D'Asaro, whose comments, advice and support contributed to the development of the first version of the ocean simulation model.

The mention of commercial products or the use of company names does not in any way imply endorsement by the U.S. Navy or NOARL.

**Ocean Simulation Model for Internal Waves
Computer Source Code**

Introduction

The computer code contained in this technical note is provided as a supplement to NOARL Report 10, "Ocean Simulation Model for Internal Waves." The philosophy and algorithms used in the code are documented in that report.

This technical note is primarily intended to provide a permanent record of the source code and a reference for anyone who wishes to implement or modify this code. The code can only run with a file containing the Levitus 5° climatic oceanographic data base. This file and the source code can be obtained on an ASCII tape or in VAX BACKUP format from NOARL, Code 331. To obtain this tape, a request should be submitted to

Commanding Officer
Attn: Code 331
Naval Oceanographic and Atmospheric Research Laboratory
Stennis Space Center, MS 39529-5004.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

```
C*****
C
C PROGRAM          MODEL1
C
C PURPOSE          FIRST LEVEL OCEAN SIMULATION MODEL.
C                  This model utilizes a coarse oceanographic data base
C                  to define the stratification. The variability is
C                  introduced through advecting the fields of temperature
C                  and salinity by vertical internal wave motions at
C                  given positions and times, based on a Garrett-Munk
C                  model.
C
C CURRENT DATE     03/23/89
C
C AUTHOR(S)        K.D. Saunders (NOARL)
C
C
C
C
C*****
```



```

C*****
C*****
C
C OUTPUT
C-----
C Unit   FILE           DATA
C-----
C   6     SYS$OUTPUT      « ephemeral file »
C
C           1. Diagnostic information
C
C   11     DIAGNOSTICS.LIS « ASCII file »
C
C           1. Diagnostic information
C
C   12     MODEL1.DAT      « Direct/Unformatted»
C
C           1. Displacement fields           (m)
C           2. Modified Temperature fields   (°C)
C           3. Modified Salinity fields       (psu)
C           4. Sound velocity fields          (m/s)
C
C   13     MODEL1.AUX      « ASCII »
C
C           Defining Parameters (labeled)
C
C   14     MODEL1.UV       « DIRECT »
C
C           1. U velocity field (m/s)
C           2. V velocity field (m/s)
C           3. W velocity field (m/s)
C
C   15     MODEL1.EIG      « DIRECT »
C
C           1. Modal Eigenvalues and Eigenfunctions for
C              W(j,z,x),k(j,x)
C
C   16     MODEL1.CTL      « DIRECT »
C
C           1. Control information relating to restart and
C              eigenmode/eigenvalue control and use
C
C   20     DEBUG.DAT       « ASCII »
C
C           1. DEBUGGING TOOL
C
C*****
C*****

```

```

C*****
C*****
C
C
C
C NOTES
C
C 1. The following assumptions are made for this first level
C    model:
C
C    ▫ no mean currents are assumed.
C      (this restriction will be relaxed in later versions)
C
C    ▫ only the internal wave part of the spectrum (including the M2
C      tidal contribution) affects the fields of temperature and
C      salinity.
C      (this restriction will be relaxed in later versions)
C
C    ▫ the temperature and salinity fields are initially defined
C      based on the Levitus 5° data base averages.
C      (this restriction will be relaxed in later versions)
C
C    ▫ if the BV frequency is imaginary, it is set to zero in the
C      mode calculations.
C
C    ▫ the internal wave field does not affect the modes for  $t > 0$ 
C      (this restriction may be relaxed in later versions)
C
C 2. The profile input section is derived from programs written
C    by William Teague, NOARL, Code 331 in conjunction with the
C    MOODS data base project.
C
C 3. The internal wave simulation section was originally derived
C    from programs written by Dr. David Rubenstein, SAIC, but
C    which have been since extensively modified at NOARL.
C
C 4. Record lengths differ on the VAX and the CONVEX. An
C    INTEGER*4 variable, RECCTL, addresses this difference.
C    For the VAX, RECCTL = 1
C    For the CONVEX, RECCTL = 4
C
C*****
C*****

```

```
C*****
C*****
C
C STRUCTURE
C
C CONTROL_INPUT
C   EIG_CONTROL
C   |
C   PROFILE_INPUT
C   |
C   INT_WAVE_SIMULATION
C   (TIME_LOOP)
C     (X - LOOP)
C       DISPLACE
C       PROFILE_CALC
C         (OUTPUT --> MODEL1.DAT)
C       (END_ZLOOP)
C     (END_X_LOOP)
C   (END_TIME_LOOP)
C*****
C*****
C
C PROGRAM LAYERING
C
C MAIN
C |
C -----
C |          |          |
C CONTROL_INPUT  PROFILE_INPUT  INT_WAVE_SIMULATION
C |          |          |
C EIG_CONTROL    |          |
C               |          |
C             -----      |
C             |LINT| INTRPL| DIST|
C             |          |          |
C                   BVFREQ           |
C                                   |
C                               -----
C                               |DISPLACE| PROFILE_CALC
C                               |          |
C                         -----
C                         |MODESUB| SZ| INTERP
C                         |          |
C                     -----
C                     |TURN| NUMEROV| INTERP| AVGINIT
C*****
C*****
```

```

C*****
C*****
C
C DEVELOPMENT STATUS and HISTORY
C
C-----
C ROUTINE          DATE          STATUS
C-----
C
C MAIN            10/25/88        Written
C
C CONTROL_INPUT   10/25/88        Written
C
C                10/26/88        Implicit NONE added
C
C                11/18/88        Added File for U,V output -
C                                unit 14
C
C EIG_CONTROL     2/8/89          Begun
C
C PROFILE_INPUT   10/25/88        Partially coded and tested
C
C                10/26/88        Profile Interpolation added
C                                Implicit NONE added
C                                BV Calculations added
C                                Diagnostic output file added
C
C                11/15/88        Compute min-max BV frequency
C
C LINT            10/25/88        Existed - modfied to include
C                                end points.
C
C                10/26/88        Implicit NONE added
C
C INTRPL          -              KDS LIBRARY ROUTINE
C
C DIST           -              KDS PROGRAM BASE
C
C                10/26/88        Documentation improved,
C                                Implicit NONE added
C BVFREQ          -              WHOI LIBRARY ROUTINE
C
C                10/26/88        Documentation improved
C                                Implicit NONE added
C
C INT_WAVE_SIMULATION 10/25/88    « Skeleton »
C
C                10/26/88        Coded - testing in progress
C

```

C				*
C	DISPLACE	10/26/88	*Rubenstein code	*
C				*
C		10/27/88	Single profile per call set up	*
C			Reinitialization of variables	*
C			before call to MODESUB	*
C			Added 3 parameters to	*
C			sequence: ix, nbvmax, nxmax	*
C				*
C		11/3/88	Added NDIR random directions	*
C			for each mode and frequency	*
C			contribution in isotropic case	*
C				*
C		11/8/88	Added random phase to each	*
C			contribution from a direction	*
C			frequency and mode	*
C				*
C			jstar set to 3 to agree with	*
C			GM spectrum: p57, Flatté	*
C				*
C		11/15/88	Use fixed frequencies for each	*
C			profile. This will help in	*
C			improving eigenvalue estimates	*
C				*
C		11/16/88	Changed weighting of each	*
C			directional component from	*
C			1/NDIR to 1/sqrt(NDIR)	*
C				*
C		11/18/88	Added U,V calculations and file	*
C				*
C		01/17/89	M2 Tidal Component added	*
C				*
C		02/03/89	DF replaced by sqrt(DF) in the	*
C			simulation of integration over	*
C			frequency.	*
C				*
C			Radian/sec frequency replaced	*
C			cph frequency in integration	*
C				*
C		02/06/89	ZD,U,V, rederived from W	*
C				*
C		03/02/89	Logarithmic frequencies steps	*
C			introduced.	*
C				*
C	SZ	10/25/88	*Rubenstein code	*
C				*
C		11/8/88	Summation of H(j) changed to use	*
C			all the j, not just the odd	*
C			modes	*
C				*
C		02/06/89	Scaling modified to reflect the	*
C			variation in the BV profile.	*
C				*

C	MODESUB	10/26/88	*Rubenstein code	*
C				*
C		11/15/88	Modified to use previous k's	*
C			as starting points for next	*
C			calculations	*
C				*
C		02/03/89	Normalization modified to create	*
C			orthonormal eigenmodes.	*
C				*
C	TURN	10/25/88	*Rubenstein code	*
C				*
C	NUMEROV	10/25/88	*Rubenstein code	*
C				*
C	INTERP	10/25/88	*Rubenstein code	*
C				*
C		10/27/88	Modified array declarations to	*
C			variable dimensions	*
C				*
C	AVGINT	10/25/88	*Rubenstein code	*
C				*
C	PROFILE_CALC	10/26/88	« Skeleton »	*
C				*
C		10/27/88	Coded - testing in progress	*
C				*
C				*
C	*****			*
C	*****			*

IMPLICIT NONE

CHARACTER*8 TIMEBUFF
CHARACTER*9 DATEBUFF

REAL TTT0,TTT1,DTTT,DTT1

INCLUDE 'MODEL1.INC'

CALL CONTROL_INPUT
CALL PROFILE_INPUT
CALL INT_WAVE_SIMULATION

STOP ' NORMAL END OF PROGRAM REACHED'
END

SUBROUTINE CONTROL_INPUT

```

C*****
C*****
C
C PROGRAM          CONTROL_INPUT
C
C PURPOSE          Reads in control data
C                  IO terminal files opened
C                  Auxilliary Latitude and Longitude computed
C
C HISTORY          10/25/88          1. Coding begun.
C
C AUTHOR(S)        K.D. Saunders (NOARL)
C
C*****
C*****
C
C INPUT
C-----
C      FILE          DATA
C-----
C      SYS$INPUT      1. Starting Latitude    (decimal °)
C                    2. Starting Longitude    (decimal °)
C                    3. Direction of section ( ° from north)
C                    4. Max Range (xmax)       ( km )
C                    5. Max Depth (zmax)       ( m )
C                    6. Delta x                ( km )
C                    7. Delta z                ( m )
C                    8. Max time               ( s )
C                    9. Delta time             ( s )
C
C*****
C*****
C
C OUTPUT
C-----
C      FILE          DATA
C-----
C      SYS$OUTPUT     1. Diagnostic information
C
C      COMMONS        1. All output is passed through named common
C*****
C*****

```

IMPLICIT NONE

CHARACTER*8 TIMEBUFF
CHARACTER*9 DATEBUFF
CHARACTER*3 ANS

INTEGER IANG,NT TMP
INTEGER NDIR,IT0,NT_DUM
INTEGER ITMP

REAL SINE,COSE

INCLUDE 'MODEL1.INC'

LOGICAL RESTART

COMMON /DIR/ NDIR,IT0
COMMON /EIG_COM/ RESTART

OPEN (FILE=TERMINAL_INPUT,UNIT=5,STATUS='UNKNOWN',DISP='DELETE')
OPEN (FILE=TERMINAL_OUT ,UNIT=6,STATUS='UNKNOWN',DISP='DELETE')
OPEN (FILE='DIAGNOSTICS.LIS',UNIT=11,STATUS='NEW',DISP='KEEP')
OPEN (FILE='DEBUG.DAT',UNIT=20,STATUS='NEW',DISP='KEEP')

WRITE(*,*) ' RESTART = ', RESTART

CALL EIG_CONTROL

WRITE(*,*) ' RESTART = ', RESTART

IF(.NOT. RESTART) THEN

* Read in control *
* data *

WRITE(6,100)
READ(5,*) LAT,LON,AZIMUTH
WRITE(6,105)
READ(5,1) SEASON
WRITE(6,110)
READ(5,*) XMAX,DX,ZMAX,DZ,TMAX,DT
WRITE(6,160)
READ(5,*) NDIR
WRITE(6,200)
READ(5,*) NEIG,NMODES,NF
WRITE(6,210)
READ(5,1) ANS
IF(ANS .EQ. 'YES') THEN
 GM_PROF = .TRUE.
ELSE
 GM_PROF = .FALSE.
END IF

IF(NEIG .EQ. 0) NEIG = 1000
IF(NF .EQ. 0) NF = 8
IF(NMODES .EQ. 0) NMODES = 5

WRITE(6,220)
READ(5,1) ANS
IF(ANS .EQ. 'YES') THEN
 TIDES = .TRUE.
ELSE
 TIDES = .FALSE.

C
C
C
C

```

END IF

IF( NDIR .GT. 30) NDIR = 30
NT = TMAX/DT + 1

NX = XMAX/DX + 1
IF( NX .GT. MAX) THEN
    NX = MAX
    DX = XMAX/(NX-1)
    WRITE( 6,140) NX,DX
    WRITE(11,140) NX,DX
END IF

NZ = ZMAX/DZ + 1
IF( NZ .GT. MAX) THEN
    NZ = MAX
    DZ = ZMAX/(NZ-1)
    WRITE( 6,150) NZ,DZ
    WRITE(11,150) NZ,DZ
END IF

WRITE(11,120) LAT,LON,AZIMUTH,XMAX,ZMAX,TMAX,NX,NZ,NT,NDIR
WRITE(11,170) DT,DX,DZ
WRITE( 6,120) LAT,LON,AZIMUTH,XMAX,ZMAX,TMAX,NX,NZ,NT,NDIR

```

C
C
C
C
C

```

*****
* Compute nearest pos.      *
* that is 5° away from*
* from the input pos.      *
*****

```

C
C
C
C
C

```

*****
* Reduce azimuth to      *
* nearest multiple of *
* 45°                    *
*****

```

```

IF( AZIMUTH .LT. 0) AZIMUTH = AZIMUTH + 360.0
IANG = AZIMUTH + 22.5
IANG = 45*(IANG/45)
IF( IANG .GE. 360) IANG = 0

AZIMUTH = IANG

```

C
C
C
C

```

*****
* Locate point on          *
* edge of 10° square *
*****

```

```

SINE = SIND(AZIMUTH)
COSE = COSD(AZIMUTH)
IF( SINE .NE. 0) THEN
    SINE = SINE/ABS(SINE)
END IF
IF( COSE .NE. 0) THEN
    COSE = COSE/ABS(COSE)
END IF

```

C
C
C
C

```

*****
* Lat1 and lon1 used *
* in profile_input *
*****

```

```

LAT1 = LAT + 5.*COSI
LON1 = LON + 5.*SINI

```

```

WRITE(11,130) LAT,LON,LAT1,LON1,AZIMUTH
WRITE(6,130)  LAT,LON,LAT1,LON1,AZIMUTH

```

C
C
C
C
C
C

```

*****
* DIFFERENT RECORD LENGTHS *
*
* For CONVEX: RECCTL = 4 *
* For VAX:      RECCTL = 1 *
*****

```

C

```

RECCTL = 1
RECCTL = 4

```

```

1 OPEN (FILE='MODEL1.DAT',UNIT=12,STATUS='NEW',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=RECCTL*NZ*4)
OPEN (FILE='MODEL1.AUX',UNIT=13,STATUS='NEW',DISP='KEEP')
1 OPEN (FILE='MODEL1.UV',UNIT=14,STATUS='NEW',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=3*NZ*RECCTL)
      itmp = NZ+1
1 OPEN (FILE='MODEL1.EIG',UNIT=15,STATUS='NEW',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=itmp*RECCTL)
1 OPEN (FILE='MODEL1.CTL',UNIT=16,STATUS='NEW',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=5*RECCTL)

```

```

T = 0
WRITE(16,REC=1) NX,DX,XMAX
WRITE(16,REC=2) NZ,DZ,ZMAX
WRITE(16,REC=3) NT,DT,TMAX
WRITE(16,REC=4) LAT,LON,LAT1,LON1,AZIMUTH
WRITE(16,REC=5) T,0,NDIR
WRITE(16,REC=6) NEIG,NMODES,NF

```

ELSE

```

1 OPEN (FILE='MODEL1.DAT',UNIT=12,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=4*NZ*RECCTL)
OPEN (FILE='MODEL1.AUX',UNIT=13,STATUS='OLD',DISP='KEEP')
1 OPEN (FILE='MODEL1.UV',UNIT=14,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=3*NZ*RECCTL)
      ITMP = NZ+1
1 OPEN (FILE='MODEL1.EIG',UNIT=15,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=ITMP*RECCTL)
1 OPEN (FILE='MODEL1.CTL',UNIT=16,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=5*RECCTL)

```

```

WRITE(6,190) NT
READ(5,*) NT TMP
IF( NT_TMP .GT. NT ) THEN
      NT = NT_TMP
ELSE
      STOP ' **** MAX TIME STEPS TOO SMALL **** '
END IF

```

```

READ(16,REC=3) NT_DUM,DT,TMAX
WRITE(16,REC=3) NT,DT,TMAX

```

```

READ (16,REC=5) T,IT0,NDIR
IF( NDIR .EQ. 0 ) THEN
      WRITE(*,*) ' ENTER NON-ZERO VALUE FOR NDIR '
      READ(5,*) NDIR

```

```
      WRITE (16,REC=5) T,IT0,NDIR  
END IF  
READ(16,REC=6) NEIG,NMODES,NF
```

```
END IF
```

```
RETURN
```

```

1      FORMAT(A)

100    FORMAT( //' *****OCEAN SIMULATION MODEL*****'//
1      '                               VERSION 1.0                               '///
2      ' Enter latitude, longitude and direction of section in'//
3      ' decimal degrees.                                     '///)

105    FORMAT(//' Enter season (WINTER,SPRING,SUMMER OR FALL)'//)

110    FORMAT(// ' Enter maximum and delta ranges in km, '//
1      '           maximum and delta depths in m, '//
2      '           maximum and delta times in s '///)

120    FORMAT(// ' INPUT DATA '// *****'//
1      ' Latitude = ',f13.3/
2      ' Longitude = ',f13.3/
3      ' Azimuth = ',f13.3//
4      ' Xmax = ',f13.3/
5      ' Zmax = ',f13.3/
6      ' Tmax = ',G13.3/
7      ' NX,NZ,NT,NDIR = ', 4i8//)

130    FORMAT(//' COMPUTED VALUES '//
1      ' LAT = ',F10.3,'          LON = ',F10.3/
2      ' LAT1 = ',F10.3,'         LON1 = ',F10.3/
3      ' Reduced Azimuth = ',f10.3//)

140    FORMAT( ' NX,DX Have been redefined to conform to storage'
1      ' requirements.'// NX = ',I10/' DX = ',G16.5//)

150    FORMAT( ' NZ,DZ Have been redefined to conform to storage'
1      ' requirements.'// NZ = ',I10/' DZ = ',G16.5//)

160    FORMAT( ' ENTER THE NUMBER OF DIRECTIONS TO USE ',
1      ' IN ISOTROPIC CASE'//)

170    FORMAT( ' DT,DX,DZ = ', 3F18.3///)

190    FORMAT( //' THE OLD VALUE FOR NT IS ', I5/
1      ' ENTER A NEW VALUE FOR NT > NT(OLD) '//)

200    FORMAT( //' ENTER NEIG, NMODES, NF ')
210    FORMAT( //' DO YOU WANT GENERIC GARRET-MUNK BVF PROFILE ?')

220    FORMAT( //' DO YOU WANT M2 INTERNAL TIDES ?')

      END

```

```

SUBROUTINE      EIG_CONTROL
C*****
C
C SUBROUTINE      EIG_CONTROL
C
C PURPOSE          Checks to see whether a restart is possible, and, if so
C                  requests information relating to whether it is desired.
C                  If the program is not to be restarted, the old files
C                  are closed and new ones opened. Otherwise, the old
C                  data and control files are reused.
C
C Author           K.D. Saunders
C
C History          2/8/89  - Coding begun
C
C*****
C*****
C
C Notes
C
C   The logical variable, RESTART is used to determine whether this
C   is a restart or a new run. It will also be used in determining
C   whether to recompute eigenfunctions later in the program.
C
C*****

```

```

CHARACTER*80      QUERY,ANSWER

INTEGER           DATRECL,EIGRECL,IDUM
INTEGER           LENG1,LENG2
INTEGER           NDIR,ITO

LOGICAL           RESTART,AUXEXIST,DATEXIST,CTLEXIST,EIGEXIST

COMMON /EIG_COM/  RESTART
COMMON /DIR/      NDIR,ITO

INCLUDE 'MODEL1.INC'

```

```

C
C *****
C * Test to see if the Eigenfunction
C * data and control files exist and are
C * compatible with any existing version
C * the MODEL1.AUX file.
C *
C * If they are, query to see if a restart
C * is desired. If not, close all the
C * files and start over.
C *****

```

```

RESTART = .TRUE.

```

```

INQUIRE(FILE='MODEL1.AUX',EXIST=AUXEXIST)
INQUIRE(FILE='MODEL1.DAT',EXIST=DATEXIST,RECL=leng1)
INQUIRE(FILE='MODEL1.EIG',EXIST=EIGEXIST,RECL=leng2)
INQUIRE(FILE='MODEL1.CTL',EXIST=CTLEXIST)

```

```

DATRECL = leng1*RECCTL
EIGRECL = leng2*RECCTL

```

```

1 IF( .NOT. (AUXEXIST .AND. DATEXIST .AND. CTLEXIST .AND.
            EIGEXIST) ) THEN
    WRITE(*,*) ' AUXEXIST ', AUXEXIST
    WRITE(*,*) ' DATEXIST ', DATEXIST

```

```
WRITE(*,*) ' CTLEXIST ', CTLEXIST
WRITE(*,*) ' EIGEXIST ', EIGEXIST
```

```
RESTART = .FALSE.
```

```
ELSE
```

```
1 OPEN( UNIT=16,FILE='MODEL1.CTL',STATUS='OLD',
      DISP='KEEP',ACCESS='DIRECT',RECL=5*RECCTL)
```

```
READ(16,REC=1) NX,DX,XMAX
READ(16,REC=2) NZ,DZ,ZMAX
READ(16,REC=3) NT,DT,TMAX
READ(16,REC=4) LAT,LON,LAT1,LON1,AZIMUTH
READ(16,REC=5) T,IT0,NDIR
READ(16,REC=6) NEIG,NMODES,NF
```

```
CLOSE( UNIT=16)
```

```
IF( DATRECL .NE. 4*RECCTL*NZ) THEN
    RESTART = .FALSE.
    WRITE(*,*) ' DATRECL ',DATRECL,4*RECCTL*NZ
END IF
```

```
IF( EIGRECL .NE. 4*(NZ+1)) THEN
    RESTART = .FALSE.
    WRITE(*,*) ' EIGRECL ',EIGRECL,4*(NZ+1)
END IF
```

```
IF ( RESTART ) THEN
    WRITE(6,100)
    READ(5,1) ANSWER
    IF ( INDEX(ANSWER,'YES') .NE. 0 .OR.
        INDEX(ANSWER,'yes') .NE. 0) THEN
        RESTART = .TRUE.
    ELSE
        RESTART = .FALSE.
    END IF
END IF
```

```
END IF
```

```
1 FORMAT(A)
100 FORMAT( ' DO YOU WANT TO RESTART THE PROGRAM WHERE IT',
1         ' WAS LEFT OFF ? '//
2         ' Answering YES will restart at that point '//
3         ' Answering NO will reinitialize the computation'//)
```

```
RETURN
END
```

SUBROUTINE PROFILE_INPUT

IMPLICIT NONE

INCLUDE 'MODEL1.INC'

```
C*****
C*****
C
C PROGRAM        PROFILE_INPUT
C
C PURPOSE        LOCATES PROFILES AT LAT,LON,LAT1,LON1 AND READS IN THE
C                TEMPERATURE AND SALINITY PROFILES AT BOTH LOCATIONS FROM*
C                LEVITUS 5° DATABASE.
C
C HISTORY        10/25/88        1. Program begun.
C
C AUTHOR(S)      K.D. Saunders (NOARL)
C
C*****
C*****
C INPUT
C
C        All input is via named common
C
C*****
C*****
C OUTPUT
C        SYS$OUTPUT        Diagnostic information
C
C        COMMONS        All data are returned via named common
C
C*****
C*****
C NOTES
C
C        The following notes are from the comments in Wm. Teague's program
C-----*
C        PROGRAM:    LEVFEB
C        PURPOSE:    THIS PROGRAM READS A DIRECT ACCESS FILE CREATED BY LEVRD AND
C                    WRITES AND WRITES THE DATA IN VFEB FORMAT.  THE OUTPUT GROUP
C                    CONSISTS OF 30 DEPTH LEVELS WITH DEPENDENT VARIABLES OF
C                    NO. OF TEMP OBSERVATIONS, MEAN TEMP, STANDARD DEVIATION OF
C                    TEMP, NO. OF SAL OBSERVATIONS, MEAN SAL, AND STANDARD DEVIATION
C                    OF SAL.
C-----*
C*****
```

```
INTEGER    ISHIF,
1           IPOSLOOP,
2           ISF,
3           IREC
```

```
REAL D(180),
1     ZLEV(30),
2     T_TEMP(300),
3     S_TEMP(300),
4     P(2),
```

```

5     PAV,
6     X_RATIO,
7     D_PROFILES,
8     E,
9     RLAT,
A     RLON,
B     DIST,
C     BVFRQ

```

```

CHARACTER*80    LEVFILE

```

```

1     DATA ZLEV/0,10,20,30,50,75,100,125,150,200,250,300,400,500,
2           600,700,800,900,1000,1100,1200,1300,1400,1500,
           1750,2000,2500,3000,3500,4000/

```

```

C
C
C
C

```

```

*****
* OPEN INPUT FILE. VAX *
* FILE SYSTEM TO BE USED*
*****

```

```

IF ( RECCTL .EQ. 1) THEN
  LEVFILE = 'MODELBASE$:LEVITUS.DAT'
END IF

```

```

C
C
C
C

```

```

*****
* OPEN INPUT FILE. CONVEX*
* FILE SYSTEM TO BE USED *
*****

```

```

IF ( RECCTL .EQ. 4) THEN
  LEVFILE = 'LEVITUS.DAT'
END IF

```

```

&     OPEN(UNIT=10,FILE=LEVFILE,
&     ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='OLD',
&     ERR=9091,RECL=180*RECCTL,READONLY)

```

```

C
C
C

```

```

* WINTER = FEB, MAR, APR *
* - USE MID MARCH FOR TIME *
* IN FDOC(1,1) *

```

```

IF (SEASON(1:2).EQ.'WI')THEN
  ISHIF=0

```

```

C

```

```

* SPRING = MAY, JUN, JUL *

```

```

ELSE IF (SEASON(1:2).EQ.'SP')THEN
  ISHIF=36

```

```

C

```

```

* SUMMER = AUG, SEP, OCT *

```

```

ELSE IF (SEASON(1:2).EQ.'SU')THEN
  ISHIF=72

```

```

C

```

```

* FALL = NOV, DEC, JAN *

```

```

ELSE IF (SEASON(1:2).EQ.'FA')THEN
  ISHIF=108
ELSE

```

```

C
C

```

```

* USE SUMMER IF SEASON *
* NOT CORRECTLY SPECIFIED *

```

```

  ISHIF = 72
END IF

```

```

DO 200 IPOSLOOP = 1,2

```

```

  IF(IPOSLOOP .EQ. 1) THEN
    RLAT = LAT
    RLON = LON

```

```

      ELSE
        RLAT = LAT1
        RLON = LON1
      END IF

      IF(RLON.LT.0)RLON=RLON+360.
      RLAT=RLAT+90.

```

```

*****
* CHECK LAT LON VALUES *
*****

```

```

      IF(ABS(RLON).GE.360.)THEN
        WRITE(6,*)'LONGITUDE NOT BETWEEN -180 AND 180 ',RLON
        STOP ' LONGITUDE ERROR - PROGRAM STOPPED '
      END IF
      IF(ABS(RLAT).GT.180)THEN
        WRITE(6,*)'LATITUDE NOT BETWEEN -90 AND 90 ',RLAT
        STOP ' LATITUDE ERROR - PROGRAM STOPPED '
      ENDIF

```

```

*****
* COMPUTE DIRECT ACCESS RECORD NO.S *
*****

```

```

      I=RLON/5.+1.
      J=RLAT/5.+1.
      IREC=(I-1)*144+J+ISHIF

```

```

*****
* READ DATA RECORD - NUMOBS, TEMP,*
* SIGMA, NUMOBS, SAL, SIGMA      *
*****

```

```

      READ(10,REC=IREC,ERR=9092)D

```

```

      K=0
      ISF=0

```

```

      WRITE(11,130)
      DO 50 L=1,90,3

```

```

        K=K+1
        BUF(1)=ZLEV(K)
        BUF(2)=D(L)
        BUF(3)=D(L+1)
        BUF(4)=D(L+2)
        BUF(5)=D(L+90)
        BUF(6)=D(L+91)
        BUF(7)=D(L+92)

```

```

*****
* CHECK FOR 0 OBSERVATIONS *
* INSERT MISSING RECORD   *
* FLAG THEN -999.0        *
*****

```

```

      IF(BUF(2).LE.0.1)THEN
        BUF(3)=-999.0
        BUF(4)=-999.0
      END IF
      IF(BUF(5).LE.0.1)THEN
        BUF(6)=-999.0
        BUF(7)=-999.0
      END IF
      Z_IN(IPOSLOOP,K) = BUF(1)
      TEMP_IN(IPOSLOOP,K) = BUF(3)
      SAL_IN(IPOSLOOP,K) = BUF(6)

```

```

      IF ( K.GT.1 .AND. TEMP_IN(IPOSLOOP,K) .LE.-998.0) THEN
        TEMP_IN(IPOSLOOP,K) = TEMP_IN(IPOSLOOP,K-1)
      END IF

```

```

      IF ( K GT.1 .AND. SAL_IN(IPOSLOOP,K) .LE. -998.0) THEN
        SAL_IN(IPOSLOOP,K) = SAL_IN(IPOSLOOP,K-1)
      END IF

```

```

      WRITE(11,140) IPOSLOOP,K,TEMP_IN(IPOSLOOP,K),SAL_IN(IPOSLOOP,K)

```

```

50    CONTINUE

```

```

200  CONTINUE

```

```

C                                     *****
C                                     * CLOSE THE LEVITUS FILE *
C                                     *****
      CLOSE(UNIT=10)

```

```

C*****
C                                     *
C      INTERPOLATE TEMPERATURE AND SALINITY PROFILES FROM THE INPUT *
C      PROFILES ONTO THE SECTION *
C                                     *
C      1. First, compute the distance between the profiles and use as *
C      input distance. *
C      2. Second, fill T,S to desired depth if required *
C      3. Interpolate to the z-grid *
C      4. Interpolate to the x-grid *
C      5. Compute Brunt-Väisälä frequencies *
C                                     *
C*****

```

```

      D_PROFILES = DIST(LAT1,LON1,LAT,LON)
      X_BASE(1) = 0.0
      X_BASE(2) = D_PROFILES

```

```

      DO 210 I = 1,NX
        X_OUT(I) = (I-1)*DX
210    CONTINUE

```

```

      DO 220 I = 1,NZ
        Z_OUT(I) = (I-1)*DZ
        ZBV(I) = Z_OUT(I)
220    CONTINUE

```

```

      X_RATIO = XMAX/D_PROFILES

```

```

      IF( GM_PROF) THEN

```

```

        DO 230 K = 1,NX
          BVMAX(K) = 2.99
          DO 240 I = 1,NZ
            BVF(I,K) = 3.0*EXP(-ZBV(I)/1300.0)
            IF( I.LT.4) BVF(I,K) = 2.99
240          CONTINUE
230        CONTINUE

```

```

        FMAX = 2.99
        RETURN

```

```

      ELSE

```

```

C                                     *****
C                                     * Set up starting *
C                                     * temperature *
C                                     * and salinity *
C                                     * profiles *
C                                     *****

```

```

DO 250 I = 1,30
  T_TEMP(I) = TEMP_IN(1,I)
  S_TEMP(I) = SAL_IN(1,I)
CONTINUE

CALL INTRPL(6,30,ZLEV,T_TEMP,NZ,Z_OUT,DUMMY)

DO 260 I = 1,NZ
  TEMP(I,1) = DUMMY(I)
CONTINUE

CALL INTRPL(6,30,ZLEV,S_TEMP,NZ,Z_OUT,DUMMY)

DO 270 I = 1,NZ
  SAL(I,1) = DUMMY(I)
CONTINUE

*****
* Set up ending T,S *
* profiles, scaled *
* by ratio of xmax to *
* distance between *
* input profiles *
*****

DO 280 I = 1,30
  T_TEMP(I) = TEMP_IN(2,I)
  S_TEMP(I) = SAL_IN(2,I)
CONTINUE

CALL INTRPL(6,30,ZLEV,T_TEMP,NZ,Z_OUT,DUMMY)

DO 290 I = 1,NZ
  TEMP(I,NX) = (DUMMY(I)-TEMP(I,1))*X_RATIO + TEMP(I,1)
CONTINUE

CALL INTRPL(6,30,ZLEV,S_TEMP,NZ,Z_OUT,DUMMY)

DO 300 I = 1,NZ
  SAL(I,NX) = (DUMMY(I)-SAL(I,1))*X_RATIO + SAL(I,1)
CONTINUE

*****
* Fill the temp, sal *
* arrays. Use linear *
* interpolation. *
*****

DO 310 I = 1,NZ
  CALL LINT(TEMP(I,1),TEMP(I,NX),NX,DUMMY)
  DO 320 K = 2, NX-1
    TEMP(I,K) = DUMMY(K)
  CONTINUE

  CALL LINT(SAL(I,1),SAL(I,NX),NX,DUMMY)
  DO 330 K = 2, NX-1
    SAL(I,K) = DUMMY(K)
  CONTINUE
CONTINUE

*****
* Compute BV Freqs *
*****

* Set min of max bvf's *

FMAX = 1.0E10
DO 340 K = 1,NX
  BVMAX(K) = 0.0
DO 350 I = 1,NZ-1

```

```

      T_TEMP(1) = TEMP(I,K)
      T_TEMP(2) = TEMP(I+1,K)
      S_TEMP(1) = SAL(I,K)
      S_TEMP(2) = SAL(I+1,K)
      P(1)      = Z_OUT(I)
      P(2)      = Z_OUT(I+1)
      BVF(I,K)  = BVFRQ(S_TEMP,T_TEMP,P,2,PAV,E)
      IF(BVMAX(K).LT.BVF(I,K))BVMAX(K) = BVF(I,K)
350    CONTINUE
      BVF(NZ,K) = BVF(NZ-1,K)
      IF(FMAX.GT.BVMAX(K))FMAX = BVMAX(K)
340    CONTINUE

      END IF
999    RETURN

```

```

9091      STOP 'ERROR IN OPENING LEVITUS FILE'
9092      STOP 'ERROR IN READING LEVITUS FILE'

100      FORMAT( ' PROFILE ',I4,'      X = ',F10.3//
1         '          Z          T          S          BV '//)
110      FORMAT( 4F12.3)
120      FORMAT( '***** INITIAL INTERPOLATE PROFILES TEMP,SAL,BVF',
1         ' ARRAYS *****'//)
130      FORMAT('// INPUT TEMPERATURE AND SALINITY PROFILES '//)
140      FORMAT( 1X,2I4,2F12.3 )

```

END

SUBROUTINE INT_WAVE_SIMULATION

```

C*****
C*****
C
C PROGRAM      INT_WAVE_SIMULATION
C
C PURPOSE      Does most of the calculations for MODEL1. It is based
C              on the Garrett-Munk internal wave model.
C
C HISTORY      10/26/88      Coding begun
C
C AUTHOR(S)    K.D. Saunders (NOARL)
C
C
C*****
C*****
C
C INPUT        All interprocess communication is via named common.
C
C
C*****
C*****
C
C OUTPUT      All output is done in subroutine calls.
C
C*****
C*****
C
C Notes
C   Subroutines called:
C
C               MODE_CALC
C               DISPLACEMENTS
C               PROFILE_CALC
C
C*****
C*****

```

IMPLICIT NONE

INCLUDE 'MODEL1.INC'

```

LOGICAL*1      BV_CHANGED / .TRUE. /
LOGICAL        RESTART
INTEGER        IDIR,NBV,NNZ,IIX,IT_T,IT0
INTEGER        NDIR
REAL           ZT(MAX),BVT(MAX)

```

```

COMMON /EIG_COM/RESTART
COMMON /DIR/      NDIR,IT0

```

```

NMODES = 5
NF      = 8
NBV     = NZ
NNZ     = NZ

```

```

READ(16,REC=3) NT,DT,TMAX
READ(16,REC=5) T0, IT0 ,NDIR
READ(16,REC=6) NEIG,NMODES,NF

```

```
WRITE(13,120) NT,NX,NZ,DT,DX,DZ,T0,LAT,LON,AZIMUTH
```

```
TMAX = NT*DT
```

```
WRITE(16,REC=3) NT,DT,TMAX
```

```
WRITE(*,*) ' NT,DT,TMAX,IT0 ', NT,DT,TMAX,IT0
```

```
IF (IT0 .GT. NT) IT0 = NT
```

```
DO 360 IT = IT0 + 1 , NT
```

```
T = ( IT - 1 ) * DT
```

```
DO 370 IX = 1,NX
```

```
DO 380 K = 1,NBV  
  BVT(K) = BVF(K,IX)  
  ZT(K) = ZBV(K)
```

```
380 CONTINUE
```

```
IDIR = 0
```

```
*****  
* This is the same call*  
* as in Rubenstein's, *  
* except for the IX *  
* parameter. The *  
* displacements are *  
* computed at each *  
* range. *  
*****
```

```
CALL DISPLACE(ZD,NNZ,NX,1000.*XMAX,T,  
1 AZIMUTH,IDIR,NF,NMODES,  
2 LAT,NBV,ZT,BVT,IX,  
3 MAX,MAX,FMAX,IT,NEIG,  
4 TIDES)
```

```
CALL PROFILE_CALC
```

```
*****  
* END IX-LOOP *  
*****
```

```
370 CONTINUE
```

```
RESTART = .TRUE.
```

```
WRITE(16,REC=5) T,IT,NDIR
```

```
*****  
* END IT-LOOP *  
*****
```

```
360 CONTINUE
```

```
RETURN
```

```
100 FORMAT( ' IX,NX,NZ,NBV,NF,NMODES =',7I10/  
1 ' XMAX = ',G15.4,' T = ',G15.4,  
2 ' LAT = ',G15.4//)
```

```
120 FORMAT( ' ***** OCEAN SIMULATION MODEL VERSION 1.0 '//  
1 ' NT ',I5/  
2 ' NX ',I5/  
3 ' NZ ',I5/  
4 ' DT ',G20.5/  
5 ' DX ',G20.5/  
6 ' DZ ',G20.5/
```

7 ' TO ',G20.5/
8 ' LAT ',G20.5/
9 ' LON ',G20.5/
A ' AZ ',G20.5//)

END

```

C*****
C*****
C INFORMATIONAL DOCUMENTATION ONLY !
C*****
C*****
C PROGRAM      DPERT
C
C PURPOSE      COMPUTES RANDOM INTERNAL WAVE DISPLACEMENTS
C
C AUTHOR       D. RUBENSTEIN (SAIC) - ORIGINAL AUTHOR (LOWER CASE)
C              K.D. SAUNDERS (NOARL)- MODIFIER (IN CAPS)
C
C HISTORY      9/25/88          - RECEIVED AT NOARL
C              9/26/88          - INITIAL MODIFICATION TO RUN ON UVAX
C                               a. Changed rands, rnd to RAN
C                               b. Set up output file for displacement
C                               data.
C              10/24/88          - Added documentation
C
C INPUT        FILE NAME « filename »(CHARACTER)
C              DATA IN « filename »
C              Line 1
C                  nf          - number of frequencies in expansion
C                  nmodes      - number of modes
C                  nbv         - number of points in bv profile
C                  nz          - number of points in vertical
C                  lat         - latitude
C              Line 2
C                  nx          - number of points in horizontal
C                  totx        - total distance in x-direction (m)
C                  nt          - number of time steps
C                  dt          - delta time (sec)
C                  angle       - Azimuth angle of vertical plane (°)
C                  idir        - directionality flag
C                               0 = isotropic
C                               1 = Along-range propagation ky=0
C                               2 = Cross-range propagation kx=0
C              Line 3 - 2+nbv
C                  z(j)        - depth (m)
C                  bv(j)       - BV frequency (cph)
C
C NOTES:       1. This program computes internal wave displacements in
C              the x-z plane at equally spaced times. The program
C              reads a data file containing the control parameters
C              and a single Brunt-Väisälä frequency profile. All the
C              displacements are computed for this single profile.
C*****

```

```

1  SUBROUTINE DISPLACE ( Z, NZ, NX, TOTX, T, ANGLE, IDIR,
2  NF, NMODES, LAT, NBV, ZT, BVT, IX,
   NBVMAX, NXMAX, FMAX, IT, NEIG, TIDES)

```

```

C*****
C
C PROGRAM          DISPLACE
C
C PURPOSE:
C
C Calculate random vertical displacements (correlated in time) due to
C internal waves. A Garrett-Munk type of spectrum is used to generate the
C proper energy levels. The displacements are packed into array Z(NZ,),
C which covers a vertical plane.
C
C Input Parameters
C
C NZ          Number of points in the vertical used in
C              calculating modes (Integer*4)
C NX          Number of points in the horizontal (Integer*4)
C TOTX        Total distance in x-direction, in meters (Real*4)
C T           Time in seconds (Real*4)
C ANGLE       Azimuth angle of the vertical plane, in degrees
C IDIR        Flag for directionality of internal waves
C              = 0  Isotropic
C              = 1  Along-range propagation (ky = 0)
C              = 2  Cross-range propagation (kx = 0)
C NF          Number of frequencies in expansion (Integer*4)
C NMODES      Number of modes in expansion (Integer*4)
C LAT         Latitude, in degrees (Real*4)
C NBV         Number of points in BV profile, and in output array Z(Integer*4)
C ZT          Depths of BV frequencies, and of output displacements Z,
C              in meters (Real*4 array of length NBV)
C BVT         Set of BV frequencies, in cph (Real*4 array of length NBV)
C
C Output parameter
C
C Z           Array of vertical displacements, in meters (Real*4 2-D array
C              of size NBVMAX x NXMAX)
C
C Note: The BV-frequency array BVT is of length NBV, which is interpolated
C onto a regularly spaced grid of length NZ. The output array WM from
C subroutine MODESUB is interpolated back into a grid of length NBV.
C
C MAX is the maximum number of depth points allowed in MODESUB calculations
C
C*****

```

```

PARAMETER ( MAX = 8000, MODEMAX = 50, NFMAX = 50, NDIRMAX=50)

REAL Z(NBVMAX), BVT(NBVMAX), ZT(NBVMAX), LAT
REAL F(NFMAX), FR(NFMAX), ZDEP(MAX), WMINT(MAX)
REAL K(0:MODEMAX), WM(MAX,0:MODEMAX), KSTART(0:MODEMAX,NFMAX)
REAL KX, KY, FMAX
REAL COST(0:MODEMAX,NFMAX,NDIRMAX), SINT(0:MODEMAX,NFMAX,NDIRMAX)

REAL PHASE(0:MODEMAX,NFMAX,NDIRMAX),
1     U(1000),V(1000),W(1000),FF,DZ,UF,VF,UMAG,VMAG,WMAG,ZMAG,
2     DCZMAG,DFF(NFMAX),YFK(NFMAX),YFL(NFMAX),
3     EPSILON,QQU,QQV,QQF,QQZ,QW,QDW

COMPLEX A(0:MODEMAX,NFMAX), I1, FACTOR, DCZ
1     ,VTT,UTT,ZTT,WTT,QF1,QFU,QFV,QFZ

INTEGER ISEED0 /191531459/,
1     ISEED,
2     IX,
3     IT,
4     NX,
5     LOC_REC,
6     LOC_REC1,
7     NDIR,
8     IID

LOGICAL*1 FIRSTPASS /.TRUE./,
1     FIRSTGO /.TRUE./,
2     TIDES
LOGICAL RESTART

COMMON /DIR/ NDIR
COMMON /EIG_COM/ RESTART

DATA JSTAR / 3 /

```

```

C*****
C
C  FI and BVMAX are inertial frequency and maximum BV frequency, in cph.*
C
C*****
C  INITIALIZE VARIABLES ON EACH PASS THROUGH THIS ROUTINE
C*****

```

```

DZ = ZT(2)-ZT(1)

```

```

ISEED = ISEED0
I1 = (0., 1.)
PI = 4.*ATAN(1.)
DEGRAD = PI/180.

```

```

*****
* Added 11/8/88 - kds to
* to conform to Garrett-Munk
* form - p57 in Flatté
*****

```

```

JSTAR = 3

```

```

FACTOR = (0,0)
KX = 0
KY = 0

```

```

IF( FIRSTPASS ) THEN
  FIRSTPASS = .FALSE.

```

```

*****
* COMPUTE THE INTEGRAL OF N(z)*
*****

```

```

390  VV = 0
      DO 390 I = 2,NBV-1
        VV = VV + BVT(I)
      CONTINUE
      VV = VV + 0.5*(BVT(1)+BVT(NBV))
      VV = DZ*VV*2*PI/3600.0

```

```

SQNDIR = NDIR
SQNDIR = sqrt(SQNDIR)
WRITE(11,7020) NDIR,SQNDIR

```

```

400  DO 400 M = 0,MODEMAX
      K(M) = 0
      CONTINUE

```

```

*****
* Get array of frequencies, in cph *
*****

```

```

FI = SIN(DEGRAD*LAT)/12.0
FIR = 2.*PI*FI/3600.
DX = TOTX / (NX-1.0)

```

```

*****
* COMPUTE CENTER FREQUENCIES *
* AND DF'S
*****

```

```

DY = LOG10(FMAX/FI)/NF

```

```

DO 410 I = 1,NF+1
  YFK(I) = (I-0.5)*DY
  YFL(I) = (I-1)*DY

```

```

410          CONTINUE

C          *****
C          * Fixed frequencies *
C          * used to facilitate *
C          * eigenvalue comp's *
C          *****

C          DO 420 I = 1, NF
C              F(I) = FI + (I-0.5)*DF
C              F(I) = FI*10.0**(YFK(I))
C              DFF(I) = FI*(10.0**YFL(I+1) - 10.0**YFL(I) )
C              FR(I) = F(I)*2.*PI/3600.0

420          CONTINUE

C*****
C Note: Ordinarily, both F and DF should both be in units of rad/sec. *
C Since DF is being divided by F, and they are both in the same units *
C of cph, their units do not need to be converted (except in the *
C exponential). *
C*****

COSANG = COS ( ANGLE * DEGRAD )
SINANG = SIN ( ANGLE * DEGRAD )

ISEED = ISEED0
PHI = RAN(ISEED)

DO 430 IID = 1,NDIR

DO 140 IFREQ = 1, NF

DO 141 M = 0, NMODES

C*****
C ADJUST TO CONVENTIONAL DEFINITION OF MODE NUMBER FOR THE GARRETT-MUNK *
C SPECTRAL POWER ROUTINE. (THE LOWEST MODE, WITH NO ZERO-CROSSINGS, *
C IS M=0, OR J=1) *
C*****

J = M + 1
JMODES = NMODES + 1

C          *****
C          * DIRECTIONALITY OF PROPAGATION *
C          *****

C          IF ( IDIR .EQ. 0 ) THEN
C              THETA = 2.*pi*RAN(ISEED)
C              WRITE(6,7100) M,IFREQ,IID,ISEED,THETA
C              WRITE(11,7100) M,IFREQ,IID,ISEED,THETA
C              COST(M,IFREQ,iid) = COS(THETA)
C              SINT(M,IFREQ,iid) = SIN(THETA)
C          ELSE
C              COST(M,IFREQ,iid) = 0.
C              SINT(M,IFREQ,iid) = 0.
C              IF ( IDIR .EQ. 1 ) COST(M,IFREQ,iid) = 1.0
C              IF ( IDIR .EQ. 2 ) SINT(M,IFREQ,iid) = 1.0
C          ENDIF

C          THETA = 2.*PI*RAN(ISEED)
C          PHASE(M,IFREQ,IID) = THETA

C          IF(IID .EQ. 1) THEN
C              PHI = 2.*PI*RAN(ISEED)
C              *****
C              * NO = 3 CPH = 0.005236 RAD/S *
C              *****

C              VN = 0.005236
C              SPEC = 2.*VN*VV*SZ(FR(IFREQ),J,

```

1

JSTAR,JMODES,FIR)
 A(M,IFREQ) = SQRT(SPEC) * CEXP(I1*PHI)

END IF

 * END MODES LOOP, M *

141

CONTINUE

 * END IFREQ-LOOP *

140

CONTINUE

 * END DIRECTION LOOP, IDD *

430

CONTINUE

 * END FIRST-PASS *

END IF

 * Initialize Z *

J = IX
 DO 160 I = 1, NBVMAX
 Z(I) = 0.0
 U(I) = 0.0
 V(I) = 0.0
 W(I) = 0.0

160 CONTINUE

IPRINT = 0
 EPSILON = 0.001

 * MOVED OUTSIDE IZ LOOP *
 * 10/27/88-KDS *

RANGE = (IX-1.)*DX
 X = COSANG * RANGE
 Y = SINANG * RANGE

WRITE(*,*) ' IX,RANGE',IX,RANGE

 * TEST FOR VARIABLE BV FREQ *

IF(NEIG .GT. 1) THEN
 LOC_REC = NF*(NMODES+1)*(IX-1)
 ELSE
 LOC_REC = 0
 END IF

DO 300 IFREQ = NF, 1, -1

FACTOR=CEXP(-I1*F(IFREQ)*T*2.*PI/3600.)

 * sqrt(df/ndir) added 2/2/89 to *
 * take care of convergence in the *
 * stochastic integral sense. (See *
 * Kinsman, 1965, p368 ff for *
 * details. *

C

```

DO 440 M = 0,NMODES
  IF(FIRSTGO) THEN
    K(M) = 0.0
  ELSE
    K(M) = KSTART(M,IFREQ)
  END IF
440 CONTINUE

IF( .NOT. RESTART ) THEN
  WRITE(*,*)
  1 ' CALLING MODESUB : RESTART,IFREQ = ',RESTART,IFREQ
  CALL MODESUB ( NMODES, F(IFREQ), NBV, ZT, BVT, NZ, LAT,
  1 EPSILON,
  2 IPRINT , K, ZDEP, WM ,FIRSTGO)

  IF( IPRINT .GE. 1) THEN
    WRITE(11,*)'IFREQ=',IFREQ,F(IFREQ), ' K = '
    WRITE(11,*)(K(M),M=0,NMODES)
  END IF

END IF

IF(IX.EQ.1) THEN
  WRITE(20,6002) IFREQ,F(IFREQ),FR(IFREQ)
END IF

6002 FORMAT( // '*****'
  1 // ' Ifreq = ', i5/
  2 ' Freq = ', g20.5/
  3 ' Rad. Freq', g20.5//)

DO 250 M = 0, NMODES

  IF(IX .EQ.1) WRITE(20,6001) M,K(M)
6001 FORMAT( ' Mode = ', I5 /
  1 ' k(m) = ', g20.8//)

  IF( .NOT. RESTART ) THEN
    KSTART(M,IFREQ) = K(M)

C C C C *****
C C C C * UPDATE INITIAL K *
C C C C * MATRIX *
C C C C *****

    IF ( K(M) .LE. 1.E-18 ) THEN
      WRITE(11,170)M,IFREQ
      WRITE(6,170)M,IFREQ

170 FORMAT(' Mode ',i3,' of Frequency #',i3,
  1 ' did not converge.',/, ' Increase NZ. Program aborting')
      STOP ' ERROR - NO MODAL CONVERGENCE'
    ENDIF

  END IF

C *****
C INTERPOLATE WM, OF LENGTH NZ INTO WMINT, OF LENGTH NBV *
C *****

LOC_REC = LOC_REC + 1

```

```

IF( .NOT. RESTART) THEN
  CALL INTERP ( NZ,ZDEP,WM(1,M),NBV,ZT(NBV),ZT,WMINT)
  INQUIRE(UNIT=15,RECL=NREC15)
  WRITE(15,rec=LOC_REC) K(M),(WMINT(KK),KK=1,NBV)

```

```

ELSE
  INQUIRE(UNIT=15,RECL=NREC15)
  READ (15,rec=LOC_REC) K(M),(WMINT(KK),KK=1,NBV)

```

```

END IF

```

7990

```

FORMAT( 1X,G20.8,(1X,I5,G20.5))

```

```

DO 450 IID = 1,NDIR

```

```

  KX = COST(M,IFREQ,IID) * K(M)
  KY = SINT(M,IFREQ,IID) * K(M)
  IF(KX .EQ. 0) KX = 1.0E-9
  IF(KY .EQ. 0) KY = 1.0E-9
  IF(IX.EQ.1) THEN
    WRITE(20,6003) KX,KY,abs(a(m,ifreq))
  END IF

```

6003

1

```

  FORMAT( ' KX = ',G20.5/' KY = ',G20.5/
    'A(m,ifreq) = ',g20.5//)

```

```

  IF(IPRINT .EQ. -1) THEN
    WRITE(11,4000) M,IFREQ,IID,K(M),KX,KY,
      A(M,IFREQ),FACTOR
    FORMAT(1X,3I4,7G15.4)
  END IF

```

4000

1

```

  XXXK = KX*KX + KY*KY

```

```

  FF = FR(IFREQ)
  DFR = 2*PI*DFF(IFREQ)/3600.
  BFACT = sqrt(dfr/ndir)

```

1

2

```

  QF1 = BFACT* A(M,IFREQ) * FACTOR *
    CEXP(I1*(KX*X + KY*Y
    + PHASE(M,IFREQ,IID) ) )

```

```

  QFU = (-I1) * (I1*FIR*KY+FF*KX) / (FF*XXXK)
  QFV = (-I1) * (I1*FIR*KX+FF*KY) / (FF*XXXK)
  QFZ = 1.0/(I1*FF)

```

```

  QQU = REAL(QF1*QFU)/DZ
  QQV = REAL(QF1*QFV)/DZ
  QQF = REAL(QF1)
  QQZ = REAL(QF1*QFZ)

```

```

  DO 220 IZ = 1, NBV
    QW = WMINT(IZ)
    IF(IZ .GT. 1 .AND. K(M) .GT. 1.0E-5) THEN
      QDW= WMINT(IZ-1)-QW
      U(IZ-1) = U(IZ-1) + QQU*QDW
      V(IZ-1) = V(IZ-1) + QQV*QDW
    END IF
    W(IZ) = W(IZ) + QQF*QW
    Z(IZ) = Z(IZ) + QQZ*QW
  CONTINUE

```

220

C

C

C

450

C

CONTINUE

```

*****
* END DIRECTION LOOP, IID *
*****
*****

```

C
C
250 CONTINUE
C
C
C
300 CONTINUE

* END MODE LOOP, M *

* END FREQUENCY LOOP, IFREQ *

```

C*****
C
C      M2 TIDAL COMPONENT GOES HERE
C
C*****

```

```

IF( TIDES) THEN

```

```

DO 460 M = 0,NMODES
      K(M) = 0.0
CONTINUE

```

```

460
C
C
C

```

```

*****
* M2 TIDAL FREQUENCY *
*****

```

```

FM = 1.0/12.4
FMR = 2*PI*FM/3600.0
FIRSTGO = .TRUE.

```

```

IF ( .NOT. RESTART ) THEN

```

```

      CALL MODESUB ( NMODES, FM, NBV, ZT, BVT, NZ, LAT,
1              EPSILON,
2              IPRINT , K, ZDEP, WM ,FIRSTGO)

```

```

DO 470 M =0,NMODES
      WRITE(6,8010) M,K(M)
8010   FORMAT(1X,I5,5X,G20.8)
470   CONTINUE

```

```

END IF

```

```

DO 240 M = 0,NMODES

```

```

IF( .NOT. RESTART ) THEN
      KSTART(M,IFREQ) = K(M)

```

```

C
C
C
C

```

```

*****
* UPDATE INITIAL K *
* MATRIX *
*****

```

```

      IF ( K(M) .LE. 1.E-18 ) THEN
        WRITE(11,170)M,IFREQ
        WRITE(6,170)M,IFREQ
        STOP ' ERROR - NO MODAL CONVERGENCE'
      ENDIF
END IF

```

```

C*****
C
C      Interpolate WM, of length NZ into WMINT, of length NBV
C
C*****

```

```

      LOC_REC = LOC_REC + 1

```

```

IF( .NOT. RESTART) THEN
      CALL INTERP ( NZ,ZDEP,WM(1,M),NBV,ZT(NBV),ZT,WMINT)

```

```

C
C
C
C

```

```

*****
* NORMALIZE INTERNAL M2 *
* TIDAL EIGENFUNCTIONS *
*****

```

```

      ANORM = 1.0
      AMPLITUDE = 1.0/(M+1)

```

```

C
C
C

```

```

*****
* Assume M2 tide has *
* a base amplitude of *

```

```

C                                     * 1 meters scaled by *
C                                     * the mode no. *
C                                     *****
ANORM = AMPLITUDE*ANORM
WMAX = 0

C                                     * Apply normalization *
DO 480 IZ = 1,NBV
    WMINT(IZ) = ANORM*WMINT(IZ)
480 CONTINUE

    INQUIRE(UNIT=15,RECL=NREC15)
    WRITE(15,REC=LOC_REC) K(M),(WMINT(KK),KK=1,NBV)

ELSE
    INQUIRE(UNIT=15,RECL=NREC15)
    READ (15,REC=LOC_REC) K(M),(WMINT(KK),KK=1,NBV)

END IF

C*****
C
C    NORMALIZE INTERNAL TIDE EIGEN FUNCTIONS
C
C*****
C                                     * For now, set ky *
C                                     * to zero *
C
    KX = K(M)
    KY = 0.0
    XXKK = K(M)*K(M)
    FACTOR_M2 = CEXP(-I1*FM*T*2.*PI/3600.)

    FF = FM
    QF1 = FACTOR_M2 * CEXP( I1*(KX*X + KY*Y) )
    QFU = (-I1) * (I1*FIR*KY+FF*KX) / (FF*XXKK)
    QFV = (-I1) * (I1*FIR*KX+FF*KY) / (FF*XXKK)
    QFZ = 1.0/(I1*FF)

    QQU = REAL(QF1*QFU)/DZ
    QQV = REAL(QF1*QFV)/DZ
    QQF = REAL(QF1)
    QQZ = REAL(QF1*QFZ)

    DO 230 IZ = 1, NBV
        QW = WMINT(IZ)
        IF(IZ .GT. 1 .AND. K(M) .GT. 1.0E-5) THEN
            QDW = WMINT(IZ-1)-QW
            U(IZ-1) = U(IZ-1) + QQU*QDW
            V(IZ-1) = V(IZ-1) + QQV*QDW
        END IF
        W(IZ) = W(IZ) + QQF*QW
        Z(IZ) = Z(IZ) + QQZ*QW
230 CONTINUE

C                                     *****
C                                     * END MNODES-LOOP *
C                                     *****
240 CONTINUE

C                                     *****
C                                     * FOR TIDES *
C                                     *****

END IF

LOC_REC1 = NX*(IT-1) + IX
WRITE(14,REC=LOC_REC1) (U(IZ),V(IZ),W(IZ),IZ=1,NBV)

IF( NEIG.EQ.1) RESTART = .TRUE.

```

FIRSTGO = .FALSE.

RETURN

7000 FORMAT(//5X,'M',4X,'IF',3X,'IID',15X,'ISEED',11X,'THETA'//)

7020 FORMAT(// ' NDIR = ',I5//
 1 ' SQNDIR = ',F12.4//)

7100 FORMAT(2X,3I5,5X,I20,G16.5)

END

FUNCTION SZ (F, J, JSTAR, JMODES, FIR)

```
C*****
C
C
C  COMPUTE SPECTRAL DENSITY OF VERTICAL DISPLACEMENT
C
C  INPUT PARAMETERS:
C
C  F      Frequency, in rad/sec
C  J      Vertical mode number
C  FIR    Inertial frequency, in rad/sec
C*****
```

DATA B/1300./

PI = 4.*ATAN(1.)

E = 6.3 E-5

BW = (2.0*FIR)/(PI*F*SQRT(F**2 - FIR**2))

```
C *****
C      GET H(J)      *
C *****
```

SUM = 0.

```
C*****
C      Removed 11/8/88 - kds      *
C*****
C      do 20 jj = 1, jmodes, 2      *
C*****
```

JMAX = JMODES

JMAX = 100

DO 20 JJ = 1, JMAX

SUM = SUM + 1./(JJ*JJ + JSTAR*JSTAR)

20 CONTINUE

HJ = (1./(J*J + JSTAR*JSTAR)) / SUM

SZ = (B**2) * BW * HJ * E

RETURN

END


```
C  subroutine tests for sufficient resolution, and the constraint is greatest*
C  at high frequencies.*
```

```
PARAMETER ( MAX = 8000, MODEMAX = 50 )
```

C*****

```

      G(J) = B(BV,J,F,FI)
60    continue

      CALL TURN( G, NZ, JA, JB, JM )

      IF ( NMODES*NRES .GT. JB-JA ) THEN
        IF(IPRINT .GE. 1) WRITE(11,70) NMODES, NRES, JB-JA
70      FORMAT(' NMODES = ',i3,' * ',i3,' > JB-JA = ',i3,/,
1        ' Region between turning points insufficiently resolved.')

        NZ = NZ * 1.25
        IF(IPRINT .GE. 1) WRITE(11,25) NZ

        GO TO 28

      ENDIF

```

* (B Z) *

```

DO 80 J = 1, NZ
      G(J) = B(BV,J,F,F1)
80    CONTINUE

      CALL TURN( G, NZ, JA, JB, JM )
      CALL AVGINT ( G, NZ, JA, JB, DZ, GAVG )
      AFACTOR = (PI/GAVG)**2
      I1 = 1
      N = NZ

```

```
*****
* Loop through modes *
*****
```

```
*****
* First guess for k**2 *
*****
```

```

ITERATE = ITERATE + 1

1 CALL NUMEROV ( M, I1, N, JA, JB, JM, G, W, DZ,
                IPRINT, ICOUNT, ICROSS, K2OLD, K2NEW )

IF ( IPRINT.GE.1 .OR. MOD(ITERATE,20).EQ.0 ) THEN
  WRITE(11,*)'ITERATE,M,ICOUNT,K2OLL,K2NEW,K2MIN,K2MAX = '
  110 WRITE(11,110)ITERATE,M,ICOUNT,K2OLD,K2NEW,K2MIN,K2MAX
      FORMAT(1X,I6,2I4,2D15.4,5X,2D14.4)
ENDIF

IF ( ICOUNT .NE. M ) THEN
  IF ( ICOUNT .LT. M ) THEN
    IF ( ICROSS .EQ. 0 ) THEN
      K2MIN = AMAX1(K2MIN,K2OLD)
    ELSE
      K2MIN = AMAX1(K2MIN,0.5*(K2MIN+K2OLD))
    ENDIF
  ELSE
    K2MAX = AMIN1(K2MAX,K2OLD)
  ENDIF

  IF ( ICROSS .EQ. 1 ) THEN
    NUMBER = NUMBER + 1
    DELTA1 = 0.5*(K2MAX+K2MIN) - K2OLD
    DELTA2 = K2OLD*((2.0*M+1.)/(2.0*ICOUNT+1) - 1.0)
    IF ( ABS(DELTA1) .GT. 2.*ABS(DELTA2) ) THEN
      K2OLD = K2OLD + DELTA2
    ELSE
      K2OLD = K2OLD + DELTA1
    ENDIF
    GO TO 100
  ENDIF

  IF ( ICOUNT .LT. M ) THEN
    K2OLD = 0.5*(K2MAX+K2OLD)
  ELSE
    K2OLD = 0.5*(K2MIN+K2OLD)
  ENDIF
  GO TO 100
ENDIF

IF ( ABS((K2OLD-K2NEW)/K2NEW) .GT. EPSLON ) THEN
  IF ( ICROSS .EQ. 1 ) GO TO 100
  IF ( K2NEW .GT. K2OLD ) THEN
    K2MIN = K2OLD
  ELSE
    K2MAX = K2OLD
  ENDIF
  K2OLD = K2NEW
  GO TO 100
ENDIF

IF ( IPRINT .GE. 1 ) THEN
  WRITE(11,*)
  WRITE(11,*)'CONVERGED: K2OLD, K2NEW=',K2OLD,K2NEW
ENDIF

```

C
C
C
C

 * Pad with zeros if bottom or top *
 * were brought in *

```

115  CONTINUE
      IF ( I1 .GT. 1 ) THEN
        IF ( IPRINT .GE. 1 ) WRITE(11,*)' ZERO-Pad:  I1 = ',I1
        DO 135 I = 1, I1-1
          W(I) = 0.
135  CONTINUE
      ENDIF

      IF ( NZ .GT. N ) THEN
        IF ( IPRINT .GE. 1 ) WRITE(11,*)' ZERO-Pad:  N, NZ = ',N,NZ
        DO 140 I = N+1, NZ
          W(I) = 0.
140  CONTINUE
      ENDIF

```

C
C
C

```

*****
*   NORMALIZE W   *
*****

```

```

      SUM = 0.
      SUM2 = 0.

      DO 160 J = 2, NZ
        QT2 = (BV(J)**2 - FI**2) / (F**2 - FI**2)
        QT1 = (BV(J-1)**2 - FI**2) / (F**2 - FI**2)

        WT2 = b(bv,j ,f,fi)
        WT1 = b(bv,j-1,f,fi)

        SUM = SUM + 0.5*(WT1*W(J-1)**2 + WT2*W(J)**2)*DZ
        SUM2 = SUM2 + 0.5*(WT1+WT2)*DZ
160  CONTINUE

      ANORM = SQRT(ABS(1.0/SUM))

      WM(1,M) = 0.
      WM(NZ,M) = 0.

      DO 180 J = 2, NZ-1
        WM(J,M) = ANORM*W(J)
180  CONTINUE

      K2(M) = K2NEW
      K(M) = SQRT(K2(M))

      IF ( IPRINT .GE. 1 ) THEN
        WRITE(11,*)' FOUND MODE #',M
        WRITE(11,*)' Frequency=',f,'      K**2 = ',K2NEW
        IF(IPRINT.GE.2) write(11,200)(wm(i,m), i = 1, nz)
200  FORMAT(1X,6F12.7)
      ENDIF

```

C
C
C
C

```

*****
* Adjust afactor for next *
* higher mode             *
*****

```

```

      AFACTOR = K2OLD/(M+1.5)**2

```

C
C
C

```

*****
* END NMODES-LOOP, M *
*****

```

```

400  CONTINUE

```

```

      RETURN
      END

```

SUBROUTINE TURN (G, NZ, JA, JB, JM)

```

C*****
C
C          SUBROUTINE TURN
C
C*****

```

REAL G(NZ)

```

C
C
C          *****
C          *   FIND MAXIMUM   *
C          *****

JM = 1
GMAX = G(1)

DO 10 I = 2, NZ
  IF ( G(I) .GT. GMAX ) THEN
    JM = I
    GMAX = G(I)
  ENDIF
10 CONTINUE

IF ( JM .LE. 2 ) THEN
  IF ( G(2) .GT. 0.25*GMAX ) THEN
    JM = 3
  ELSE
    WRITE(11,*)'*** Peak too close to surface. Increase NZ.***'
    WRITE(11,*)'jm = ',jm,' gmax=',gmax,' g:'
    WRITE(11,*)(g(i),i=1,nz)
    WRITE(6,*)'*** Peak too close to surface. Increase NZ.***'
    WRITE(6,*)'jm = ',jm,' gmax=',gmax,' g:'
    WRITE(6,*)(g(i),i=1,nz)
    STOP 'ERROR - PEAK TOO CLOSE TO SURFACE'
  ENDIF
ENDIF

C
C
C          *****
C          * Find upper turning point *
C          *****

DO 20 I = JM, 1, -1
  IF ( G(I) .GT. 0. ) JA = I
20 CONTINUE

C
C
C          *****
C          * Find lower turning point *
C          *****

DO 30 I = JM, NZ
  IF ( G(I) .GT. 0. ) JB = I
30 CONTINUE

C
RETURN
END

```

SUBROUTINE AVGINT (G, NZ, JA, JB, DZ, GAVG)

```
C*****
C
C  Title      subroutine avgint
C
C  Purpose    Integrate g(z) from index j = ja to jb, and get average
C
C*****
```

REAL G(NZ)

GAVG = 0.

DO 20 J = JA, JB

GAVG = GAVG + SQRT(ABS(G(J)))

20 CONTINUE

GAVG = GAVG*DZ

RETURN

END

```

1      SUBROUTINE NUMEROV (M, I1, N, JA, JB, JM, G, W, DZ,
                          IPRINT, ICOUNT, ICROSS, K2OLD, K2NEW)

```

```

C*****
C
C  Title      subroutine avgint
C
C*****

```

```

      PARAMETER ( MAX = 8000 )

      REAL G(N), W(N), K2OLD, K2NEW

      DOUBLE PRECISION T(MAX), PHIP(MAX), PHIM(MAX)

1     DOUBLE PRECISION   S, FACT, PHI2, A1P, A2P, A1M, A2M,
                          B1P, B2P, B1M, B2M

      DOUBLE PRECISION   PHIPPR, PHIMPR

      DATA S/1./

```

```

C
C
C
C*****
C  Initialize end points
C*****

```

```

10    CONTINUE

      PHIM(I1) = 0.
      PHIM(I1+1) = S*DZ
      PHIP(N) = 0.

      IF ( MOD(M,2) .EQ. 0 ) THEN
        PHIP(N-1) = S*DZ
      ELSE
        PHIP(N-1) = -S*DZ
      ENDIF

```

```

C
C
C
C*****
C  Icross is a flag, which is set = 1
C  * if there is a zero-crossing at the
C  * match point, but no sign match
C*****

```

```

      ICROSS = 0
      ICOUNT = 0
      FACTOR = (DZ**2)*K2OLD/12.

```

```

      DO 20 I = I1, N
        T(I) = -FACTOR * G(I)
20    CONTINUE

```

```

      JBOT = I1 + 2

```

```

      DO 40 J = JBOT, JM + 2
        PHIM(J) = ( (2. + 10.*T(J-1))*PHIM(J-1) +
1         (T(J-2) - 1.)*PHIM(J-2) ) / (1.-T(J))
        IF( J.LE.JM .AND. PHIM(J-1)*PHIM(J).LE.0. ) ICOUNT = ICOUNT+1

```

```

C
C
C
C*****
C  Bring in top if exponential
C  * growth is sufficiently strong*
C*****

```

```

1      ARG = (ABS(PHIM(J))+ABS(PHIM(J-1))) /
           (ABS(PHIM(I1+1))+ABS(PHIM(I1+2)))

      IF ( ARG .GT. 1.E12 ) THEN
        IBOT = I1 + 1
        DO 30 I = J, IBOT, -1
          ARG = (ABS(PHIM(J))+ABS(PHIM(J-1))) /
                (ABS(PHIM(I))+ABS(PHIM(I-1)))
          IF ( ARG .GT. 1.E6 ) THEN
            I1 = I
            GO TO 10
          ENDIF
30      CONTINUE
      ENDIF

      IF ( ICOUNT .GT. M ) THEN
        IF ( IPRINT .GE. 1 ) WRITE(11,*)'ICOUNT > M IN PHIM'
        GO TO 600
      ENDIF

40     CONTINUE

      JTOP = N - 2

      DO 60 J = JTOP, JM-2, -1

        PHIP(J) = ( (2. + 10.*T(J+1))*PHIP(J+1) +
1          (T(J+2) - 1.)*PHIP(J+2) ) / (1.-T(J))
        IF( J.GE.JM .AND. PHIP(J+1)*PHIP(J).LE.0. ) ICOUNT = ICOUNT+1

```

```

*****
* Bring in bottom if exponential *
* growth is sufficiently strong *
*****

```

```

C
C
C
C
1      ARG = (ABS(PHIP(J))+ABS(PHIP(J+1))) /
           (ABS(PHIP(N-1))+ABS(PHIP(N-2)))
      IF ( ARG .GT. 1.E12 ) THEN
        ITOP = N - 1
        DO 50 I = J, ITOP
          ARG = (ABS(PHIP(J))+ABS(PHIP(J+1))) /
                (ABS(PHIP(I))+ABS(PHIP(I+1)))
          IF ( ARG .GT. 1.E6 ) THEN
            N = I
            GO TO 10
          ENDIF
50      CONTINUE
      ENDIF

      IF ( ICOUNT .GT. M ) THEN
        IF ( IPRINT .GE. 1 ) WRITE(11,*)'ICOUNT > M IN PHIP'
        GO TO 600
      ENDIF

60     CONTINUE

```

```

*****
* Does zero-crossing occur at *
* match-point? *
*****

```

```

1      IF ( PHIP(JM)*PHIM(JM) .LE. 0.0 ) THEN
        ICROSS = 1
        ICOUNT = ICOUNT + 1
        IF ( IPRINT .GE. 1 ) WRITE(11,*)'zero-crossing at j=jm=',
          jm,' icount=',ICOUNT

```

```

IF ( ICOUNT .NE. M ) THEN
  IF ( IPRINT .GE. 1 ) WRITE(11,*) 'icount <> m at match point'
  GO TO 600
ENDIF

```

```

C
C
C
*****
*   Look for sign-match   *
*****

```

```

1 IF ( PHIP(JM-1)*PHIM(JM).GT.0.0 .OR.
2   PHIP(JM-2)*PHIM(JM).GT.0.0 .OR.
3   PHIP(JM)*PHIM(JM+1).GT.0.0 .OR.
  PHIP(JM)*PHIM(JM+2).GT.0.0 ) THEN
  JM1 = JM + 0.5*(JB-JA)/(M+2.)
  JM2 = JM - 0.5*(JB-JA)/(M+2.)
  IF ( JM1 .GT. JA .AND. JM1 .LT. JB ) THEN
    JM = JM1
  ELSE
    IF ( JM2.GT.JA .AND. JM2.LT.JB ) THEN
      JM = JM2
    ELSE
      JM = 0.5*(JM + JB)
    ENDIF
  ENDIF
  IF ( IPRINT .GE. 1 ) WRITE(11,*)
1    'Sign match found.  New jm=',JM
  GO TO 700
ELSE
  ICOUNT = M - 1
  GO TO 600

```

```

C
C
C
*****
* End Sign-Match *
*****

C
C
C
*****
* End Zero Crossing *
*****

ENDIF

```

```

C
C
C
*****
*   Early return   *
*****

IF ( ICOUNT .NE. M ) GO TO 600

```

```

C
C
C
*****
* Adjust phi by a factor *
*****

```

```

FACT = PHIP(JM)/PHIM(JM)
IF ( FACT .GT. 1. ) THEN
  DO 70 J = JM-2, N
    PHIP(J) = PHIP(J)/FACT
70  CONTINUE
  ELSE
    DO 80 J = 2, JM+2
      PHIM(J) = FACT*PHIM(J)
80  CONTINUE
ENDIF

```

```

C
C
C
*****
* Integrate phi**2 *
*****

```

```

PHI2 = 0.
DO 100 J = 2, JM
  PHI2 = PHI2 + G(J-1)*PHIM(J-1)**2+G(J)*PHIM(J)**2
100 CONTINUE
DO 110 J = JM+1, N

```

```

      PHI2 = PHI2 + G(J-1)*PHIP(J-1)**2+G(J)*PHIP(J)**2
110  CONTINUE

```

 * Compute phip' and phim' *

```

B1P = T(JM+1)*PHIP(JM+1) - T(JM-1)*PHIP(JM-1)
B2P = T(JM+2)*PHIP(JM+2) - T(JM-2)*PHIP(JM-2)
B1M = T(JM+1)*PHIM(JM+1) - T(JM-1)*PHIM(JM-1)
B2M = T(JM+2)*PHIM(JM+2) - T(JM-2)*PHIM(JM-2)

```

```

1      PHIPPR = (16./(21.*DZ))* ( -A1P + (37./32.)*A2P - (37./5.)*B1P
      - (17./40.)*B2P )

```

```

1      PHIMPR = (16./ (21.*DZ)) * ( -A1M + (37./32.)*A2M - (37./5.)*B1M
      - (17./40.)*B2M )

```

```

DO 120 J = 1, JM
      W(J) = PHIM(J)
120  CONTINUE

```

```

DO 140 J = JM+1, N
    W(J) = PHIP(J)
140 CONTINUE

```

```
*****
* Get new trial value for k**2 *
*****
```

$$K2NEW = K2OLD - W(JM) * (PHIPPR - PHIMPR) / PHI2$$

RETURN

* Early return *

600 CONTINUE

```

DO 620 J = 1, JM
      W(J) = PHIM(J)
620  CONTINUE

```

```

DO 640 J = JM+1, N
      W(J) = PHIP(J)
640  CONTINUE

```

```
700      K2NEW = 1.E-20
```

RETURN
END

SUBROUTINE INTERP (N, Z, X, NI, ZTOTAL, ZI, XI)

```

C*****
C
C Subroutine Interp
C
C Interpolate function x(z), from depth z=0 to z=ztotal.
C
C Input parameters:
C
C N      Length of arrays X and Z
C Z      Real*4 array of length N
C X      Real*4 array of length N
C NI     Length of desired output arrays ZI and XI
C ZTOTAL Total depth to which interpolated output is desired
C
C Output parameters:
C
C ZI     Regular (Real*4) interval array, ranging from 0 to ZTOTAL,
C        of length NI
C XI     Interpolated values (Real*4 array of length NI
C
C*****

```

```

      REAL Z(1), X(1), ZI(1), XI(1)

      DZ = ZTOTAL/(NI-1)
      J = 1
      DO 50 I = 1, NI
        ZI(I) = (I-1)*DZ
40      CONTINUE
        IF ( ZI(I) .GE. Z(J) .AND. ZI(I) .LE. Z(J+1)) THEN
          XI(I) = X(J) + (X(J+1)-X(J))*(ZI(I)-Z(J))
1          /(Z(J+1)-Z(J))
          ELSE
            J = J + 1
            IF ( I.EQ.NI .AND. ABS(ZI(I)-Z(J)).LE.0.01 ) THEN
              ZI(I) = Z(J)
              XI(I) = X(J)
              RETURN
            ENDIF
            IF ( J .GT. N ) STOP 'J > N : ERR IN INTERP'
            GO TO 40
          ENDIF
50      CONTINUE

      RETURN
      END

```

SUBROUTINE PROFILE_CALC

```

C*****
C
C PROGRAM          PROFILE_CALC
C
C PURPOSE          Profiles of temperature, salinity and Brunt-Väisälä
C                  frequency are computed by "advecting" the temperature
C                  and salinity base profiles by the caculated displacement*
C                  field.
C
C AUTHOR           K.D. Saunders
C
C HISTORY          10/27/88          - Begun coding and testing
C
C INTERFACING      All program I/O is performed via named common
C
C OUTPUT
C -----
C UNIT  FILE          FORMAT          DATA
C -----
C  11  DIAGNOSTICS.LIS ASCII          Diagnostic information
C  12  MODEL1.DAT      DIRECT         OUTPUT profiles
C -----
C
C Notes:           1. The "advection" is done by creating a starting
C                  of depths defined by the base depth + displacement.
C                  The base temperatures and salinities are associated
C                  with this depth vector and are then sorted in order
C                  of increasing depth and interpolated back onto the
C                  base depth vector. The BV frequencies are then
C                  computed from the new T and S profiles.
C
C*****

```

IMPLICIT NONE

INCLUDE 'MODEL1.INC'

LOGICAL*1 SORTED

```

REAL    Z0(MAX),TTO(MAX),S0(MAX),
1       ZINT(MAX),TINT(MAX),SINT(MAX),
2       BVINT(MAX),P(2),T_TEMP(MAX),S_TEMP(MAX),
3       PAV,E,BVFRQ,DUM

```

INTEGER LOC_REC

```

DO 510 I = 1,NZ
  Z0(I)   = ZBV(I) + ZD(I)
  ZINT(I) = ZBV(I)
  TTO(I)  = TEMP(I,IX)
  S0(I)   = SAL(I,IX)
510 CONTINUE

```

```

ZINT(NZ) = Z0(NZ)
ZINT(1)  = Z0(1)

```

```

!*****
! Make sure input z's *
! are sorted in      *
! ascending order    *
!*****

```

```

DO 520 I = 1,NZ
    SORTED = .TRUE.
    DO 530 J = 1,NZ-1
        IF( Z0(J).EQ. Z0(J+1)) THEN
            Z0(J+1) = Z0(J)+.01
        END IF

```

```

        IF( Z0(J).GT. Z0(J+1)) THEN
            DUM = Z0(J)
            Z0(J) = Z0(J+1)
            Z0(J+1) = DUM
            DUM = TT0(J)
            TT0(J) = TT0(J+1)
            TT0(J+1) = DUM
            DUM = S0(J)
            S0(J) = S0(J+1)
            S0(J+1) = DUM
            SORTED = .FALSE.
        END IF

```

```

530 CONTINUE

```

```

    IF( SORTED ) GOTO 1000

```

```

520 CONTINUE

```

```

C
C
C
1000 CONTINUE

```

```

*****
* DATA IN ASCENDING ORDER *
*****

```

```

CALL INTRPL(6,NZ,Z0,TT0,NZ,ZINT,TINT)
CALL INTRPL(6,NZ,Z0,S0,NZ,ZINT,SINT)

```

```

C
C
C
*****
* Compute BV Freqs *
*****

```

```

DO 540 I = 1,NZ-1
    T_TEMP(1) = TINT(I)
    T_TEMP(2) = TINT(I+1)
    S_TEMP(1) = SINT(I)
    S_TEMP(2) = SINT(I+1)
    P(1) = ZINT(I)
    P(2) = ZINT(I+1)
    BVINT(I) = BVFRQ(S_TEMP,T_TEMP,P,2,PAV,E)

```

```

540 CONTINUE

```

```

    BVINT(NZ) = BVINT(NZ-1)

```

```

    LOC_REC = NX*(IT-1) + IX

```

```

1 WRITE(12,rec=LOC_REC)
    (ZD(I),TINT(I),SINT(I),BVINT(I),I=1,NZ)

```

```

RETURN

```

```

100 FORMAT( ' MODIFIED PROFILE ',I4,' X = ',F10.3//

```

```

1 ' Z ZD T S

```

```

110 FORMAT( 5F12.3)

```

```

120 FORMAT( '///' TIME = ', F18.4)

```

```

BV '///)

```

130

```
FORMAT(24X,'  (' ,F8.3,')' , '  (' ,F8.3,')' /)  
END
```

```

      FUNCTION  DIST(ELAT,ELONG,SLAT,SLONG)
C*****
C*****
C
C PROGRAM      DIST
C
C PURPOSE      DISTANCE IN KM BETWEEN TWO POSITIONS ON THE EARTH
C
C HISTORY      8/5/88          1. Program written
C
C AUTHOR(S)    K.D. Saunders (NOARL)
C
C*****
C*****
C
C INPUT
C              SLAT - REAL*4          - STARTING LATITUDE IN DEC. °
C              SLON - REAL*4          - STARTING LONGITUDE IN DEC. °
C              ELAT - REAL*4          - ENDING  LATITUDE IN DEC. °
C              ELON - REAL*4          - ENDING  LONGITUDE IN DEC. °
C
C*****
C*****
C
C OUTPUT
C              DIST - REAL*4          - DISTANCE IN KM BETWEEN THE
C                                     STARTING AND ENDING POSITIONS
C
C*****
C*****
C
C Notes
C              The program assumes a spherical earth and uses basic
C              spherical trigonometry.
C
C              One degree of latitude is assumed to be 111.195 km.
C
C*****
C*****

```

IMPLICIT NONE

```

REAL ELAT,ELONG,SLAT,SLONG,SL,EL,DL,X,DIST
REAL CONV,KMPERDEG /111.195/,TWOPI

```

```

TWOPI  =  2*3.14159265
CONV   =  3.14159265/180.

```

```

1  IF( ABS(ELAT) .GT. 90. .OR.
2     ABS(SLAT) .GT. 90. .OR.
3     ABS(ELONG).GT.180. .OR.
   ABS(SLONG).GT.180. ) THEN

```

```

      DIST = 99999.0
      RETURN

```

END IF

```

EL = ELAT*CONV
SL = SLAT*CONV

```

```

DL = ABS(ELONG-SLONG)*CONV

```

```

IF (DL .GE. TWOPI) DL = DL - TWOPI
X = SIN(EL)*SIN(SL) + COS(EL)*COS(SL)*COS(DL)
IF(ABS(ABS(X) - 1.0) .LT. 0.00001) THEN
    DIST = 0.0
    RETURN
END IF

IF(ABS (X) .GT. 1 ) THEN
    DIST = 9999.
    WRITE(*,*) ' ARGUMENT TO ACOS .GT. 1, =',X
    RETURN
END IF

DIST = KMPERDEG*ACOS(X)/CONV

RETURN
END

```

SUBROUTINE LINT(X1,X2,N,X)

```

C*****
C*****
C
C PROGRAM          LINT
C
C PURPOSE          LINEAR INTERPOLATOR
C
C HISTORY          8/5/88          1. Program written
C
C AUTHOR(S)        K.D. Saunders (NOARL)
C
C
C*****
C*****
C
C INPUT
C
C          X1 - REAL*4
C          X2 - REAL*4
C          N  - INTEGER*4
C
C
C*****
C*****
C
C OUTPUT
C
C          X - real array
C
C
C*****
C*****
C
C Notes
C
C          Given Values X1 and X2, computes N (inclusive) points
C          between them. I.e. X(1) = X1, X(N) = X2 with the rest
C          evenly spaced.
C
C*****
C*****

```

IMPLICIT NONE

C ***** PASSED VARIABLES *****

```

REAL    X1,X2,X(*)
INTEGER N

```

C ***** LOCAL VARIABLES *****

```

REAL    DX
INTEGER I

```

DX = (X2-X1)/(N-1)

DO 550 I = 1,N

X(I) = (I-1)*DX + X1

550 CONTINUE

RETURN

END

SUBROUTINE INTRPL(IU,L,X,Y,N,U,V)

```

C*****
C
C PROGRAM          INTRPL
C
C PURPOSE          INTERPOLATION OF A SINGLE VALUED FUNCTION.
C                  THIS SUBROUTINE INTERPOLATES, FROM VALUES OF THE
C                  FUNCTION GIVEN A ORDINATES OF INPUT DATA POINTS IN
C                  THE X-Y PLANE AND FOR A GIVEN SET OF X-VALUES(ABCISSAS),
C                  THE VALUES OF A SINGLE VALUED FUNCTION Y=Y(X).
C
C AUTHOR           HIROSHI AKIMA,U.S.DEPT OF COMMERCE,OFFICE OF
C                  TELECOMMUNICATIONS, INSTITUTE OF TELECOMMUNICATIONS
C                  SCIENCES, BOULDER COLO
C                  THIS ALGORITHM WAS PUBLISHED IN COMM. ACM. 15(10)
C                  OCT 1972
C
C*****
C
C INPUT PARAMETERS ARE
C   IU = LOGICAL UNIT NUMBER OF STANDARD OUTPUT UNIT
C   L  = NUMBER OF INPUT DATA POINTS
C   X  = ARRAY OF DIMENSION L STORING THE X VALUES
C        (ABCISSAS) OF THE DATA POINTS IN ASCENDING ORDER
C   Y  = ARRAY OF DIMENSION L STORING THE Y VALUES
C        (ORDINATES) OF THE INPUT DATA POINTS
C   N  = NUMBER OF POINTS AT WHICH INTERPOLATION OF THE
C        Y VALUES (ORDINATE) IS DESIRED
C   U  = ARRAY OF DIMENSION N STORING THE X VALUES OF THE
C        DESIRED POINTS.
C
C OUTPUT PARAMETERS
C   V  = ARRAY OF DIMENSION N WHERE THE INTERPOLATED Y
C        VALUES ARE STORED
C*****

```

DIMENSION X(1),Y(1),U(1),V(1)

EQUIVALENCE (P0,X3),(Q0,Y3),(Q1,T3)

REAL M1,M2,M3,M4,M5

EQUIVALENCE (UK,DX),(IMN,X2,A1,M1),(IMX,X5,A5,M5),
1 (J,SW,SA),(Y2,W2,W4,Q2),(Y5,W3,Q3)

```

C*****
C                  PRELIMINARY PROCESSING
C*****

```

```

10      L0=L

        LM1=L0-1
        LM2=LM1-1
        LP1=L0+1
        N0=N
        IF( LM2 .LT. 0 )      GO TO 90
        IF( N0 .LE. 0 )      GO TO 91
        DO 11 I=2,L0
            IF(X(I-1)-X(I))  11,95,96
11      CONTINUE

```

IPV = 0

C
C
C

* MAIN DO LOOP *

DO 80 K = 1,N0
 UK = U(K)

C
C
C

* ROUTINE TO LOCATE DESIRED POINT *

20 IF(LM2 .EQ. 0) GO TO 27
 IF(UK .GE. X(L0))GO TO 26
 IF(UK .LT. X(1)) GO TO 25

 IMN=2
 IMX = L0

21 I = (IMN+IMX)/2
 IF(UK .GT. X(I)) GO TO 23

22 IMX = I
 GO TO 24

23 IMN = I + 1

24 IF(IMX .GT. IMN) GO TO 21
 I = IMX
 GO TO 30

25 I=1
 GO TO 30

26 I = LP1
 GO TO 30
27 I=2

C
C
C

* CHECK IF I = IPV *

30 IF(I .EQ. IPV) GO TO 70
 IPV = I

C
C
C
C
C

* ROUTINES TO PICK UP NECESSARY X *
* AND Y VALUES AND TO ESTIMATE *
* THEM IF NECESSARY *

40 J = I
 IF(J.EQ.1) J=2
 IF(J.EQ.LP1) J=L0

 X3 = X(J-1)
 Y3 = Y(J-1)
 X4 = X(J)
 Y4 = Y(J)
 A3 = X4-X3

 M3 = (Y4-Y3)/A3
 IF(LM2 .EQ. 0) GO TO 43
 IF(J .EQ. 2) GO TO 41

 X2 = X(J-2)
 Y2 = Y(J-2)

```

A2 = X3-X2
M2 = (Y3-Y2)/A2

41 IF(J .EQ. L0) GO TO 42
X5 = X(J+1)
Y5 = Y(J+1)

A4 = X5-X4
M4 = (Y5-Y4)/A4
IF(J .EQ. 2) M2 = M3 + M3 - M4
GO TO 45

42 M4 = M3+M3-M2
GO TO 45

43 M2 = M3

45 IF(J .LE. 3) GO TO 46
A1 = X2-X(J-3)
M1 = (Y2-Y(J-3))/A1

GO TO 47

46 M1 = M2+M2-M3

47 IF(J .GE. LM1) GO TO 48
A5 = X(J+2) - X5
M5 = (Y(J+2) - Y5)/A5
GO TO 50

48 M5=M4+M4-M3

C
C
C

50 IF( I .EQ. LP1) GO TO 52

W2 = ABS(M4-M3)
W3 = ABS(M2-M1)
SW = W2+W3
IF(SW .NE. 0.0) GO TO 51
W2 = 0.5
W3 = 0.5
SW = 1.0

51 T3 = (W2*M2+W3*M3)/SW

IF(I .EQ. 1) GO TO 54

52 W3 = ABS(M5-M4)
W4 = ABS(M3-M2)
SW = W3+W4

IF(SW .NE. 0.0) GO TO 53

W3 = 0.5
W4 = 0.5
SW = 1.0

53 T4=(W3*M3+W4*M4)/SW

IF(I .NE. LP1) GO TO 60
T3 = T4
SA = A2 + A3
T4 = 0.5*(M4+M5-A2*(A2-A3)*(M2-M3)/(SA*SA))
X3 = X4

```

```

*****
* NUMERICAL DIFFERENTIATION *
*****

```

```

Y3 = Y4
A3 = A2
M3 = M4
GO TO 60

```

```

54      T4 = T3

```

```

      SA = A3+A4
      T3 = 0.5*(M1+M2-A4*(A3-A4)*(M3-M4)/(SA*SA))
      X3 = X3 - A4
      Y3 = Y3 - M2*A4
      A3 = A4
      M3 = M2

```

```

C      *****
C      * DETERMINATION OF THE COEFFICIENTS *
C      *****

```

```

60      Q2 = (2.0*(M3-T3)+M3 - T4)/A3
      Q3 = (-M3-M3+T3+T4)/(A3*A3)

```

```

C      *****
C      * COMPUTATION OF THE POLYNOMIAL *
C      *****

```

```

70      DX = UK-P0

```

```

80      V(K) = Q0+DX*(Q1+DX*(Q2+DX*Q3))

```

```

      RETURN

```

```

C      *****
C      * ERROR EXITS *
C      *****

```

```

90      WRITE(IU,2090)
      GO TO 99

```

```

91      WRITE(IU,2091)
      GO TO 99

```

```

95      WRITE(IU,2095)
      GO TO 97

```

```

96      WRITE(IU,2096)

```

```

97      WRITE(IU,2097) I,X(I)

```

```

99      WRITE(IU,2099) L0,N0
      RETURN

```

```

2090     FORMAT(1X,' *** L = 1 OR LESS'//)
2091     FORMAT(1X,' *** N = 0 OR LESS'//)
2095     FORMAT(1X,' *** IDENTICAL X VALUES'//)
2096     FORMAT(1X,' *** X VALUES OUT OF SEQUENCE'//)
2097     FORMAT(1X,' I=',I7,10X,'X(I) =',E12.3//)
2099     FORMAT(1X,' I=',I7,10X,'N =',I7//)
*1X,' *****ERROR DETECTED IN ROUTINE INTRPL*****'//)
      END

```

FUNCTION BVFRQ(S,T,P,NOBS,PAV,E)

```

C*****
C
C PROGRAM          BVFRQ
C
C PURPOSE          COMPUTES Brunt-Väisälä frequency in CPH *
C
C AUTHOR           R. MILLARD, WOODS HOLE OCEANOGRAPHIC INSTITUTION
C
C NOTES:
C
C   USES 1980 EQUATION OF STATE
C
C   UNITS:
C     PRESSURE      P0      DECIBARS
C     TEMPERATURE   T       DEG CELSIUS (IPTS-68)
C     SALINITY      S       (IPSS-78)
C     BOUYANCY FREQ BVFRQ   CPH
C     N**2          E       RADIANS/SECOND
C
C CHECKVALUE: BVFRQ=14.57836 CPH E=6.4739928E-4 RAD/SEC.
C             S(1)=35.0, T(1)=5.0, P(1)=1000.0
C             S(2)=35.0, T(2)=4.0, P(2)=1002.0
C *****NOTE RESULT CENTERED AT PAV=1001.0 DBARS *****
C JULY 12 1982
C COMPUTES N IN CYCLES PER HOUR, AND E=N**2 IN RAD/SEC**2
C AFTER FORMULATION OF BRECK OWEN'S & N.P. FOFONOFF
C
C*****

```

IMPLICIT NONE

REAL P(1),T(1),S(1)

```

REAL      E,BVFRQ,CXX,CX,CXY,CY,PAV,DATA,V350P,VBAR,
1         SIG,DVDP,A0
REAL      SVAN,THETA
INTEGER   NOBS,K

```

EXTERNAL SVAN,THETA

```

E = 0.0
BVFRQ = 0.0

```

IF(NOBS.LT.2) RETURN

```

CXX = 0.0
CX  = 0.0
CXY = 0.0
CY  = 0.0

```

```

*****
* COMPUTE LEAST SQUARES ESTIMATE OF *
* SPECIFIC VOLUME ANAMOLY GRADIENT *
*****

```

```

DO 20 K=1,NOBS
  CX = CX+P(K)

```

20 CONTINUE

PAV=CX/NOBS

```

DO 35 K=1,NOBS
  DATA = SVAN(S(K),THETA(S(K),T(K),P(K),PAV),PAV,SIG)*1.0E-8

```

```

      CXY = CXY+DATA*(P(K)-PAV)
      CY  = CY+DATA
      CXX = CXX+(P(K)-PAV)**2
35  CONTINUE

      IF(CXX.EQ.0.0) RETURN

      A0   = CXY/CXX
      V350P = (1./((SIG+1000.))-DATA
      VBAR  = V350P+CY/NOBS
      DVDP  = A0

      IF(VBAR.EQ.0.0) RETURN

      E     = -.96168423E-2*DVDP/(VBAR)**2
      BVFRQ = 572.9578*SIGN(SQRT(ABS(E)),E)

      RETURN
      END

```

FUNCTION GRADY(Y,P,NOBS,PAV,YBAR)

```

C*****
C  FUNCTION COMPUTE LEAST SQUARES SLOPE 'GRADY' OF Y VERSUS P      *
C  THE GRADIENT IS REPRESENTATIVE OF THE INTERVAL CENTERED AT PAV *
C                                                                    *
C  COMPUTE GRADIENT OF Y VERSUS P                                  *
C  JULY 15 1982                                                    *
C*****

```

```

      REAL P(1),Y(1)

      GRADY = 0.0
      A0    = 0.0
      CXX   = 0.0
      CX    = 0.0
      CXY   = 0.0
      CY    = 0.0

      IF(NOBS.LE.1) GO TO 30

      DO 20 K=1,NOBS
20      CX = CX+P(K)

      PAV = CX/NOBS

      DO 35 K=1,NOBS
      CXY=CXY+Y(K)*(P(K)-PAV)
      CY =CY+Y(K)
      CXX=CXX+(P(K)-PAV)**2
35  CONTINUE

      IF(CXX.EQ.0.0) RETURN

      A0   = CXY/CXX
      YBAR = CY/NOBS

30  CONTINUE

      GRADY = A0

```

RETURN
END

FUNCTION B (BV,J,F,FI)

```

C*****
C
C FUNCTION      B
C
C PURPOSE      Weighting function in eigenvalue/eigenfunction equation
C              W'' + k^2 B(z) W = 0.
C
C Author       K.D. Saunders
C
C History      11/18/88          Begun Coding
C
C Parameters
C              BV(*)   Real*4   Array of Brunt-Väisälä frequencies.
C              J       Int*4    Index to BV ( z = (j-1)*dz )
C              F       Real*4   Frequency
C              FI      Real*4   Inertial Frequency
C
C Notes        In this implementation,
C              B = (BV(j)**2-f**2)/(F**2-fi**2).
C
C              This is not especially useful now, but will be when it
C              becomes necessary to include current shear.
C*****

```

IMPLICIT NONE

REAL F,FI,BV(1),B

INTEGER J

B = (BV(J)**2 - F**2)/(F**2 - FI**2)

RETURN
END

REAL FUNCTION SVAN(S,T,P0,SIGMA)

```

C *****
C SPECIFIC VOLUME ANOMALY (STERIC ANOMALY) BASED ON 1980 EQUATION
C OF STATE FOR SEAWATER AND 1978 PRACTICAL SALINITY SCALE.
C
C REFERENCES:
C   MILLERO, ET AL (1980) DEEP-SEA RES.,27A,255-264
C   MILLERO AND POISSON 1981,DEEP-SEA RES.,28A PP 625-629.
C
C BOTH ABOVE REFERENCES ARE ALSO FOUND IN UNESCO REPORT 38 (1981)
C MODIFIED RCM
C UNITS:
C   PRESSURE      P0      DECIBARS
C   TEMPERATURE   T       DEG CELSIUS (IPSS-68)
C   SALINITY      S       (IPSS-78)
C   SPEC. VOL. ANA. SVAN   M**3/KG *1.0E-8
C   DENSITY ANA.  SIGMA   KG/M**3
C *****
C

```

```

C CHECK VALUE: SVAN=981.3021 E-8 M**3/KG.  FOR S = 40 (IPSS-78) ,
C T = 40 DEG C, P0= 10000 DECIBARS.
C
C CHECK VALUE: SIGMA = 59.82037 KG/M**3 FOR S = 40 (IPSS-78) ,
C T = 40 DEG C, P0= 10000 DECIBARS.
C
C *****

```

```

REAL P,T,S,SIG,SR,R1,R2,R3,R4
REAL A,B,C,D,E,A1,B1,AW,BW,K,K0,KW,K35

```

```

*****
* EQUIVALENCE STMTS *
*****

```

```

EQUIVALENCE (E,D,B1),(BW,B,R3),(C,A1,R2)
EQUIVALENCE (AW,A,R1),(KW,K0,K)

```

```

*****
* DATA *
*****

```

```

DATA R3500,R4/1028.1063,4.8314E-4/
DATA DR350/28.106331/

```

```

C *****
C R4 IS REFERED TO AS C IN MILLERO AND POISSON 1981
C CONVERT PRESSURE TO BARS AND TAKE SQUARE ROOT SALINITY.
C *****

```

```

P=P0/10.
SR = SQRT(ABS(S))

```

```

C *****
C PURE WATER DENSITY AT ATMOSPHERIC PRESSURE
C BIGG P.H.,(1967) BR. J. APPLIED PHYSICS 8 PP 521-537.
C *****

```

```

1 R1 = (((6.536332E-9*T-1.120083E-6)*T+1.001685E-4)*T
      -9.095290E-3)*T+6.793952E-2)*T-28.263737

```

```

C *****
C SEAWATER DENSITY ATM PRESS.
C COEFFICIENTS INVOLVING SALINITY
C R2 = A IN NOTATION OF MILLERO AND POISSON 1981
C *****

```

```

1 R2 = (((5.3875E-9*T-8.2467E-7)*T+7.6438E-5)*T-4.0899E-3)*T
      +8.24493E-1

```

```

C *****
C R3 = B IN NOTATION OF MILLERO AND POISSON 1981
C *****

```

```

R3 = (-1.6546E-6*T+1.0227E-4)*T-5.72466E-3

```

```

C *****
C INTERNATIONAL ONE-ATMOSPHERE EQUATION OF STATE OF SEAWATER
C *****

```

```

SIG = (R4*S + R3*SR + R2)*S + R1

```

```

C *****
C SPECIFIC VOLUME AT ATMOSPHERIC PRESSURE
C *****

```

V350P = 1.0/R3500
 SVA = -SIG*V350P/(R3500+SIG)
 SIGMA = SIG+DR350

C *****
 C SCALE SPECIFIC VOL. ANAMOLY TO NORMALLY REPORTED UNITS *
 C *****

SVAN=SVA*1.0E+8
 IF(P.EQ.0.0) RETURN

C *****
 C ***** NEW HIGH PRESSURE EQUATION OF STATE FOR SEAWATER *****
 C *****
 C MILLERO, ET AL , 1980 DSR 27A, PP 255-264 *
 C CONSTANT NOTATION FOLLOWS ARTICLE *
 C *****
 C *
 C COMPUTE COMPRESSION TERMS *
 C *****

E = (9.1697E-10*T+2.0816E-8)*T-9.9348E-7
 BW = (5.2787E-8*T-6.12293E-6)*T+3.47718E-5
 B = BW + E*S

D = 1.91075E-4
 C = (-1.6078E-6*T-1.0981E-5)*T+2.2838E-3
 AW = ((-5.77905E-7*T+1.16092E-4)*T+1.43713E-3)*T
 1 -0.1194975
 A = (D*SR + C)*S + AW

 B1 = (-5.3009E-4*T+1.6483E-2)*T+7.944E-2
 A1 = ((-6.1670E-5*T+1.09987E-2)*T-0.603459)*T+54.6746
 KW = (((-5.155288E-5*T+1.360477E-2)*T-2.327105)*T
 1 +148.4206)*T-1930.06
 K0 = (B1*SR + A1)*S + KW

C *****
 C EVALUATE PRESSURE POLYNOMIAL *
 C *
 C K EQUALS THE SECANT BULK MODULUS OF SEAWATER *
 C DK = K(S,T,P)-K(35,0,P) *
 C K35 = K(35,0,P) *
 C *****

DK = (B*P + A)*P + K0
 K35 = (5.03217E-5*P+3.359406)*P+21582.27
 GAM=P/K35
 PK = 1.0 - GAM
 SVA = SVA*PK + (V350P+SVA)*P*DK/(K35*(K35+DK))

C *****
 C SCALE SPECIFIC VOL. ANAMOLY TO NORMALLY REPORTED UNITS *
 C *****

SVAN=SVA*1.0E+8
 V350P = V350P*PK

C *****
 C COMPUTE DENSITY ANAMOLY WITH RESPECT TO 1000.0 KG/M**3 *
 C 1) DR350: DENSITY ANAMOLY AT 35 (IPSS-78), 0 DEG. C AND 0 DECIBARS *
 C 2) DR35P: DENSITY ANAMOLY 35 (IPSS-78), 0 DEG. C , PRES. VARIATION*

```

C 3) DVAN : DENSITY ANAMOLY VARIATIONS INVOLVING SPECIFIC VOL. ANAMOLY*
C *****
C
C CHECK VALUE: SIGMA = 59.82037 KG/M**3 FOR S = 40 (IPSS-78),
C T = 40 DEG C, P0= 10000 DECIBARS.
C *****

```

```

DR35P = GAM/V350P
DVAN = SVA/(V350P*(V350P+SVA))
SIGMA = DR350+DR35P-DVAN

```

```

RETURN
END

```

```

REAL FUNCTION THETA(S,T0,P0,PR)

```

```

C *****
C TO COMPUTE LOCAL POTENTIAL TEMPERATURE AT PR
C USING BRYDEN 1973 POLYNOMIAL FOR ADIABATIC LAPSE RATE
C AND RUNGE-KUTTA 4-TH ORDER INTEGRATION ALGORITHM.
C
C REFERENCES:
C   BRYDEN,H.,1973,DEEP-SEA RES.,20,401-408
C   FOFONOFF,N.,1977,DEEP-SEA RES.,24,489-491
C
C UNITS:
C   PRESSURE          P0          DECIBARS
C   TEMPERATURE       T0          DEG CELSIUS (IPSS-68)
C   SALINITY           S          (IPSS-78)
C   REFERENCE PRS     PR          DECIBARS
C   POTENTIAL TMP.    THETA       DEG CELSIUS
C
C CHECKVALUE: THETA= 36.89073 C,S=40 (IPSS-78),T0=40 DEG C,
C P0=10000 DECIBARS,PR=0 DECIBARS
C *****

```

```

C *****
C   SET-UP INTERMEDIATE TEMPERATURE AND PRESSURE VARIABLES
C *****
P = P0
T = T0

```

```

H = PR - P
XK = H*ATG(S,T,P)
T = T + 0.5*XK
Q = XK
P = P + 0.5*H
XK = H*ATG(S,T,P)
T = T + 0.29289322*(XK-Q)
Q = 0.58578644*XK + 0.121320344*Q
XK = H*ATG(S,T,P)
T = T + 1.707106781*(XK-Q)
Q = 3.414213562*XK - 4.121320344*Q
P = P + 0.5*H
XK = H*ATG(S,T,P)
THETA = T + (XK-2.0*Q)/6.0

```

```

RETURN
END

```

```

REAL FUNCTION ATG(S,T,P)

```

```

C *****
C ADIABATIC TEMPERATURE GRADIENT DEG C PER DECIBAR
C
C REFERENCES:
C   BRYDEN,H.,1973,DEEP-SEA RES.,20,401-408
C UNITS:
C   PRESSURE           P           DECIBARS
C   TEMPERATURE        T           DEG CELSIUS (IPSS-68)
C   SALINITY           S           (IPSS-78)
C   ADIABATIC          ATG          DEG. C/DECIBAR
C
C CHECKVALUE:
C   ATG=3.255976E-4 C/DBAR FOR S=40 (IPSS-78),
C   T=40 DEG C,P0=10000 DECIBARS
C *****

```

DS = S - 35.0

```

ATG = (((-2.1687E-16*T+1.8676E-14)*T-4.6206E-13)*P
1      + ((2.7759E-12*T-1.1351E-10)*DS+((-5.4481E-14*T
2      + 8.733E-12)*T-6.7795E-10)*T+1.8741E-8))*P
3      + (-4.2393E-8*T+1.8932E-6)*DS
4      + ((6.6228E-10*T-6.836E-8)*T+8.5258E-6)*T+3.5803E-5

```

RETURN
END

PROGRAM MODEL1_CTL_LIST

```

C*****
C
C PROGRAM            MODEL1_CTL_LIST
C
C PURPOSE            LISTS MODEL1 CONTROL INFORMATION
C
C AUTHOR             K.D. Saunders (NOARL, Code 331)
C
C INPUT
C-----
C Unit    Filename            Type            Contents
C-----
C 16      MODEL1.CTL          * direct *      MODEL1 Control Data
C-----
C
C OUTPUT
C-----
C Unit    Filename            Type            Contents
C-----
C 6       SYS$OUTPUT          *cntl window*   Program/control information
C-----
C*****

```

INCLUDE 'MODEL1.INC'

```

1            OPEN( UNIT=16,FILE='MODEL1.CTL',STATUS='OLD',
              DISP='KEEP',ACCESS='DIRECT',RECL=5)

```

```

              READ(16'1) NX,DX,XMAX
              READ(16'2) NZ,DZ,ZMAX
              READ(16'3) NT,DT,TMAX
              READ(16'4) LAT,LON,LAT1,LON1,AZIMUTH
              READ(16'5) T,IT_T

```

```

1            WRITE(6,1000) NX,DX,XMAX,NZ,DZ,ZMAX,NT,DT,TMAX,
              LAT,LON,LAT1,LON1,AZIMUTH,T,IT_T

```

```

              CLOSE( UNIT=16)
              STOP

```

```

1000            FORMAT( '    NX,DX,XMAX    ', I5,2G20.5 /
                      '    NZ,DZ,ZMAX    ', I5,2G20.5 /
                      '    NT,DT,TMAX    ', I5,2G20.5 /
                      '    LAT,LON       ', 5X,2G20.5/
                      '    LAT1,LON1     ', 5X,2G20.5/
                      '    AZIMUTH       ', 5X, G20.5/
                      '    T             ', 5X, G20.5/
                      '    T iteration ', I5//)
              END

```

```

C*****
C
C   PROGRAM:  LEVASC
C
C   PURPOSE:  THIS PROGRAM READS A DIRECT ACCESS FILE CREATED BY LEVRD
C              AND WRITES THE DATA IN ASCII FORMAT.  THE OUTPUT GROUP
C              CONSISTS OF 30 DEPTH LEVELS WITH DEPENDENT VARIABLES OF
C              NO. OF TEMP OBSERVATIONS, MEAN TEMP, STANDARD DEVIATION OF
C              TEMP, NO. OF SAL OBSERVATIONS, MEAN SAL, AND STANDARD
C              DEVIATION OF SAL.  THE MEAN SEASON DAY, LAT, AND LONG ARE
C              ALSO OUTPUT.
C
C   AUTHOR    S.A. Briggs (NOARL, Code 331)
C
C   INPUT
C-----
C Unit      Filename      .Type          Contents
C-----
C 10      LEVITUS.DAT      * direct *    LEVITUS data
C-----
C
C   OUTPUT
C-----
C Unit      Filename      Type          Contents
C-----
C 11      LEVITUS.ASC      * Ascii *    LEVITUS data
C-----
C*****
C
      DIMENSION D(180)

      WRITE(6,*)'CLIMATOLOGICAL ATLAS OF THE WORLD OCEAN '
      WRITE(6,*)'(NOAA PROFESSIONAL PAPER NO. 13, DEC 1982)'
      WRITE(6,*)
      WRITE(6,*) '>>> WRITING THIS FILE INTO ASCII FORMAT'

C   OPEN INPUT & OUTPUT FILES

      OPEN(UNIT=10,FILE='DRB0:[CLIMATE]LEVITUS.DAT',
& ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='OLD',
& ERR=1,RECL=180,READONLY)

      IOU = 11
      OPEN(UNIT=IOU,FILE='LEVITUS.ASC',STATUS='NEW')

      DO IREC = 1,999999
        READ(10,IREC,Iostat=IFLAG)D
        IF (IFLAG.EQ. 36) GO TO 2          ! END OF FILE CONDITION
        DO I = 1,30
          IFACT=I*6
          WRITE(IOU,100,ERR=3)(D(J),J=IFACT-5,IFACT)
100      FORMAT (6G13.3)
        END DO
      END DO

C PROGRAM TERMINATION POINTS

      1  WRITE(6,*)'ERROR IN OPENING LEVITUS FILE'
         STOP

      2  WRITE(6,*)' END OF PROGRAM'
         STOP

      3  WRITE(6,*)'ERROR IN WRITING OUTPUT FILE, UNIT ',IOU

```

STOP

END

```

C*****
C
C PROGRAM:  ASCLEV
C
C PURPOSE:  THIS PROGRAM READS AN ASCII FORMAT CREATED BY LEVASC AND
C           WRITES THE DATA INTO A DIRECT ACCESS FILE. THE OUTPUT GROUP
C           CONSISTS OF 30 DEPTH LEVELS WITH DEPENDENT VARIABLES OF
C           NO. OF TEMP OBSERVATIONS, MEAN TEMP, STANDARD DEVIATION OF
C           TEMP, NO. OF SAL OBSERVATIONS, MEAN SAL, AND STANDARD DEVIATION
C           OF SAL. THE MEAN SEASON DAY, LAT, AND LONG ARE ALSO OUTPUT.
C
C
C AUTHOR:   S.A. BRIGGS (NOARL, Code 331)
C
C*****

```

```

      DIMENSION D(180)

```

```

      WRITE(6,100)'CLIMATOLOGICAL ATLAS OF THE WORLD OCEAN '
      WRITE(6,100) '(NOAA PROFESSIONAL PAPER NO. 13, DEC 1982)'
      WRITE(6,100)
      WRITE(6,100) '>>> WRITING THIS FILE INTO DIRECT ACCESS FORMAT'
100    FORMAT (A)

```

```

C*****
C INPUT FILE *
C*****
      IOU = 11
      OPEN(UNIT=IOU,FILE='MODEL$:LEVITUS.ASC',STATUS='OLD',
&          ERR=1,READONLY)

```

```

C*****
C OUTPUT FILE *
C*****
      OPEN(UNIT=10,FILE='levitus.dat',
&          ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='NEW',
&          ERR=2,RECL=180)

```

```

      DO IREC = 1,999999
        DO I = 1,30
          IFACT=I*6
          READ(IOU,200,END=5,ERR=3)(D(J),J=IFACT-5,IFACT)
200      FORMAT (6G13.3)
          END DO
          WRITE(10,REC=IREC,ERR=4)D
        END DO

```

```

C*****
C ERROR MSGS *
C*****

```

```

      1  WRITE(6,100)' ERROR IN OPENING INPUT LEVITUS ASCII FILE'
        STOP

      2  WRITE(6,100)' ERROR IN OPENING OUTPUT LEVITUS BINARY FILE'
        STOP

      3  WRITE(6,*)' ERROR IN READING INTPUT FILE, UNIT ',IOU
        STOP

```

```
4      WRITE(6,100)' ERROR IN WRITING OUTPUT LEVITUS FILE'  
      STOP
```

```
C*****  
C  NORMAL TERMINATION *  
C*****
```

```
5      WRITE(6,100)' END OF PROGRAM'  
      STOP
```

```
END
```

PROGRAM MODEL1_UVW_READ

```

C*****
C
C PROGRAM            MODEL1_UVW_READ
C
C PURPOSE            READ U,V,W    FIELDS
C
C AUTHOR             K.D. Saunders (NORDA, Code 331)
C
C INPUT
C-----
C Unit    Filename            Type            Contents
C-----
C    5    SYS$INPUT           KEYBOARD        Control Information
C 14    MODEL1.UV            * direct *       UVW data to plot
C 13    MODEL1.AUX           * ascii *        Descriptor for MODEL1.DAT
C-----
C
C OUTPUT
C-----
C Unit    Filename            Type            Contents
C-----
C    6    SYS$OUTPUT          *cntl window*   Program/control information
C-----
C*****

```

IMPLICIT NONE

```

INTEGER*4        MAX
PARAMETER        (MAX=500)

```

```

CHARACTER*80     LINE,CBUFF(20)

```

```

INTEGER*4        I,NT,IDX,IDZ
INTEGER*4        NXX,NZZ,IX,IZ,NVARS

```

```

REAL*4
1                DT,DXX,DZZ,XMAX,
2                TD(MAX,MAX),SD(MAX,MAX),
3                U(MAX,MAX),V(MAX,MAX),W(MAX,MAX)

```

```

EQUIVALENCE      (SD(1,1),U(1,1)),(TD(1,1),V(1,1))

```

```

C*****
C        GET INPUT DATA GENERATED BY MODEL1
C*****

```

```

1        FORMAT(A)

```

```

OPEN (FILE='MODEL1.AUX',UNIT=13,STATUS='OLD',DISP='KEEP')

```

```

READ(13,1) LINE
READ(13,1) LINE
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*)    NT

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*)    NXX

```

```
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NZZ
```

```
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DT
```

```
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DXX
```

```
XMAX = DXX*(NXX-1)
```

```
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DZZ
```

```
TYPE *, ' NT,NX,NZ,DT,DX,DZ '
TYPE *, NT,NXX,NZZ
TYPE *, DT,DXX,DZZ
```

```
TYPE * , ' ENTER DIX,DIZ '
ACCEPT *, IDX,IDZ
TYPE * , ' ***** PLEASE WAIT - READING IN DATA *****'
```

```
1      OPEN (FILE='MODEL1.UV',UNIT=14,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=3*NZZ)
```

```
      DO IX= 1,NXX,IDX
        READ(14,IX)(U(I,IX),V(I,IX),W(I,IX),I=1,NZZ)
```

```
100          DO IZ = 1,NZZ,IDZ
              WRITE(6,100) IX,IZ,U(IZ,IX),V(IZ,IX),W(IZ,IX)
              FORMAT( 2I5,3G20.5)
```

```
          END DO
```

```
      END DO
```

```
STOP
END
```

PROGRAM MODEL1_LOOK

```

C*****
C
C PROGRAM          MODEL1_LOOK
C
C PURPOSE          LISTS MODEL1 DISPLACEMENT FIELDS
C
C AUTHOR           K.D. Saunders (NOARL, Code 331)
C
C INPUT
C-----
C Unit  Filename          Type          Contents
C-----
C  5     SYS$INPUT         KEYBOARD      Control Information
C 12     MODEL1.DAT        « direct »    Data to plot
C 13     MODEL1.AUX        « ascii »     Descriptor for MODEL1.DAT
C-----
C
C OUTPUT
C-----
C Unit  Filename          Type          Contents
C-----
C  6     SYS$OUTPUT        «cntl window» Program/control information
C-----
C*****

```

IMPLICIT NONE

INTEGER*4 MAX
PARAMETER (MAX=500)

CHARACTER*2 PLOT_TYPE
CHARACTER*80 LINE,CBUFF(20)

REAL*4 X,Y,XMAX,YMAX

REAL*4 J,K,R,Z

1 REAL*4 T(MAX), S(MAX), BV(MAX),
2 ZD(MAX,MAX),DT,DXX,DZZ,WM,WP,WT,HM,HP,HT,
 T_AV(MAX),S_AV(MAX),TD(MAX,MAX),SD(MAX,MAX)

INTEGER*4 N,I

1 INTEGER*4 NXX,NZZ,NT,IX,IZ,NVARS,IZMIN,IZMAX,IZDELTA,
 IXMIN,IXMAX,IXDELTA

DATA J/17.0/
DATA K/16/

```

C*****
C        GET INPUT DATA GENERATED BY MODEL1
C*****

```

1 FORMAT(A)

OPEN (FILE='MODEL1.AUX',UNIT=13,STATUS='OLD',DISP='KEEP')

```

READ(13,1) LINE
READ(13,1) LINE
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NT

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NXX

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NZZ

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DT

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DXX

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DZZ

```

```

TYPE *, ' NT,NX,NZ,DT,DX,DZ '
TYPE *, NT,NXX,NZZ
TYPE *, DT,DXX,DZZ

```

```

1 OPEN (FILE='MODEL1.DAT',UNIT=12,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=4*NZZ)

```

```

DO IX = 1,NXX
  READ(12,IX)(ZD(I,IX),T(I),S(I),BV(I),I=1,NZZ)
  DO IZ = 1,NZZ
    TD(IZ,IX) = T(IZ)
    SD(IZ,IX) = S(IZ)
    T_AV(IZ) = T_AV(IZ) + T(IZ)
    S_AV(IZ) = S_AV(IZ) + S(IZ)
  END DO
END DO

```

```

DO IZ = 1,NZZ
  T_AV(IZ) = T_AV(IZ)/NXX
  S_AV(IZ) = S_AV(IZ)/NXX
END DO

```

```

DO IX = 1,NXX
  DO IZ = 1, NZZ
    TD(IZ,IX) = TD(IZ,IX) - T_AV(IZ)
    SD(IZ,IX) = SD(IZ,IX) - S_AV(IZ)
  END DO
END DO

```

```

TYPE *, ' ENTER PLOT TYPE : ZD,TD,SD '
ACCEPT 1, PLOT_TYPE

```

```

      IF( PLOT_TYPE .EQ. 'TD' .OR. PLOT_TYPE .EQ. 'SD' ) THEN
        DO IX = 1,NXX
          DO IZ = 1,NZZ
            IF( PLOT_TYPE .EQ. 'TD' )
              ZD(IZ,IX) = TD(IZ,IX)
            IF( PLOT_TYPE .EQ. 'SD' )
              ZD(IZ,IX) = SD(IZ,IX)
          END DO
        END DO
      END IF

```

```

END IF

```

```

C*****
C      GET INPUT DATA FOR SCREEN CONTROL      *
C*****

```

```

      TYPE *, ' ENTER IZMIN,IZMAX,IZDELTA'
      ACCEPT *, IZMIN,IZMAX,IZDELTA

```

```

      TYPE *, ' ENTER IXMIN,IXMAX,IXDELTA'
      ACCEPT *, IXMIN,IXMAX,IXDELTA

```

```

      DO IZ = IZMIN,IZMAX,IZDELTA
      DO IX = IXMIN,IXMAX,IXDELTA
        TYPE *, ' IZ,IX, VALUE',IZ,IX,ZD(IZ,IX)
      END DO
      END DO

```

```

      STOP ' END MODEL1_LOOK'
      END

```

PROGRAM MODEL1_LOOK

```

C*****
C
C PROGRAM          MODEL1_LOOK
C
C PURPOSE          LISTS MODEL1 DISPLACEMENT FIELDS
C
C AUTHOR           K.D. Saunders (NOARL, Code 331)
C
C INPUT
C-----
C Unit  Filename          Type          Contents
C-----
C  5     SYS$INPUT         KEYBOARD      Control Information
C 12     MODEL1.DAT        « direct »    Data to plot
C 13     MODEL1.AUX        « ascii »     Descriptor for MODEL1.DAT
C-----
C
C OUTPUT
C-----
C Unit  Filename          Type          Contents
C-----
C  6     SYS$OUTPUT        «cntl window» Program/control information
C-----
C*****

```

IMPLICIT NONE

INTEGER*4 MAX
PARAMETER (MAX=500)

CHARACTER*2 PLOT_TYPE
CHARACTER*80 LINE,CBUFF(20)

REAL*4 X,Y,XMAX,YMAX

REAL*4 J,K,R,Z

1 REAL*4 T(MAX), S(MAX), BV(MAX),
2 ZD(MAX,MAX),DT,DX,DZZ,WM,WP,WT,HM,HP,HT,
 T_AV(MAX),S_AV(MAX),TD(MAX,MAX),SD(MAX,MAX)

INTEGER*4 N,I

1 INTEGER*4 NXX,NZZ,NT,IX,IZ,NVARS,IZMIN,IZMAX,IZDELTA,
 IXMIN,IXMAX,IXDELTA

DATA J/17.0/
DATA K/16/

```

C*****
C          GET INPUT DATA GENERATED BY MODEL1
C*****

```

1 FORMAT(A)

OPEN (FILE='MODEL1.AUX',UNIT=13,STATUS='OLD',DISP='KEEP')

```

READ(13,1) LINE
READ(13,1) LINE
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NT

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NXX

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NZZ

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DT

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DXX

```

```

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DZZ

```

```

TYPE *, ' NT,NX,NZ,DT,DX,DZ '
TYPE *, NT,NXX,NZZ
TYPE *, DT,DXX,DZZ

```

```

1 OPEN (FILE='MODEL1.DAT',UNIT=12,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=4*NZZ)

```

```

DO IX = 1,NXX
  READ(12,IX)(ZD(I,IX),T(I),S(I),BV(I),I=1,NZZ)
  DO IZ = 1,NZZ
    TD(IZ,IX) = T(IZ)
    SD(IZ,IX) = S(IZ)
    T_AV(IZ) = T_AV(IZ) + T(IZ)
    S_AV(IZ) = S_AV(IZ) + S(IZ)
  END DO
END DO

```

```

DO IZ = 1,NZZ
  T_AV(IZ) = T_AV(IZ)/NXX
  S_AV(IZ) = S_AV(IZ)/NXX
END DO

```

```

DO IX = 1,NXX
  DO IZ = 1, NZZ
    TD(IZ,IX) = TD(IZ,IX) - T_AV(IZ)
    SD(IZ,IX) = SD(IZ,IX) - S_AV(IZ)
  END DO
END DO

```

```

TYPE * , ' ENTER PLOT TYPE : ZD,TD,SD '
ACCEPT 1, PLOT_TYPE

```

```

      IF( PLOT_TYPE .EQ. 'TD' .OR. PLOT_TYPE .EQ. 'SD' ) THEN
        DO IX = 1,NXX
          DO IZ = 1,NZZ
            IF( PLOT_TYPE .EQ. 'TD' )
              ZD(IZ,IX) = TD(IZ,IX)
            IF( PLOT_TYPE .EQ. 'SD' )
              ZD(IZ,IX) = SD(IZ,IX)
          END DO
        END DO
      END IF

```

```

END IF

```

```

C*****
C      GET INPUT DATA FOR SCREEN CONTROL
C*****

```

```

TYPE *, ' ENTER IZMIN,IZMAX,IZDELTA'
ACCEPT *, IZMIN,IZMAX,IZDELTA

```

```

TYPE *, ' ENTER IXMIN,IXMAX,IXDELTA'
ACCEPT *, IXMIN,IXMAX,IXDELTA

```

```

DO IZ = IZMIN,IZMAX,IZDELTA
DO IX = IXMIN,IXMAX,IXDELTA
  TYPE *, ' IZ,IX, VALUE',IZ,IX,ZD(IZ,IX)
END DO
END DO

```

```

STOP ' END MODEL1_LOOK'
END

```

PROGRAM MODEL1_EIGLOOK

```

C*****
C
C PROGRAM            MODEL1_EIGLOOK
C
C PURPOSE            LIST & PLOT EIGENMODES AND EIGENVALUES
C
C AUTHOR             K.D. Saunders (NOARL, Code 331)
C
C INPUT
C-----
C Unit    Filename            Type            Contents
C-----
C    5    SYSSINPUT            KEYBOARD        Control Information
C 15    MODEL1.EIG            « direct »     Eigenvalues and modes
C 13    MODEL1.AUX            « ascii »     Descriptor for MODEL1.DAT
C-----
C
C OUTPUT
C-----
C Unit    Filename            Type            Contents
C-----
C    6    SYSSOUTPUT          « cntl window » Program/control information
C    NA   POPFIL.DAT          « DISSPLA METAFILE » Color plots
C-----
C*****

```

IMPLICIT NONE

INTEGER*4 MAX
PARAMETER (MAX=1000)

CHARACTER*80 LINE,CBUFF(20)

REAL*4 K(20),WMINT(MAX),Z(MAX)
REAL*4 DT,DXX,DZZ
REAL*4 WMAX,WT,ZMAX

1 INTEGER*4 NXX,NZZ,NT,IX,IZ,NVARS,IFREQ,NFREQ,NMODES,M,
 IT,LOC,NDX,NDF,NDM,KK

```

C*****
C        GET INPUT DATA GENERATED BY MODEL1
C*****

```

1 FORMAT(A)

OPEN (FILE='MODEL1.AUX',UNIT=13,STATUS='OLD',DISP='KEEP')

READ(13,1) LINE
READ(13,1) LINE
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NT

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NXX

READ(13,1) LINE

```
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NZZ
```

```
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DT
```

```
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DXX
```

```
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DZZ
```

```
TYPE *, ' NT,NX,NZ,DT,DX,DZ '
TYPE *, NT,NXX,NZZ
TYPE *, DT,DXX,DZZ
```

```
WRITE(6,*) ' ENTER ITERATION NUMBER '
READ(5,*) IT
```

```
1 OPEN (FILE='MODEL1.EIG',UNIT=15,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=NZZ+1)
```

```
NMODES = 6
NFREQ = 8
WRITE(*,*) ' ENTER NDX,NDF,NDM '
READ(*,*) NDX,NDF,NDM
DO IZ = 1,NZZ
      Z(IZ) = (IZ-1)*DZZ
END DO
ZMAX = Z(NZZ)
```

```
C *****
C  DISSPLA SUBPROGRAMS *
C  *****
```

```
CALL COMPRS
```

```
DO IX = 1,NXX,NDX
  DO IFREQ = 1,NFREQ,NDF
```

```
      CALL SETCLR(%REF('GREEN'))
      CALL PAGE(8.5,11.0)
      CALL AREA2D(5.0,6.0)
      CALL FASHON
      CALL HEIGHT(.20)
      CALL HEADIN(%REF('Vertical Eigenmodes$'),100,1.5,1)
      CALL XNAME(%REF('W$'),100)
      CALL YNAME(%REF('DEPTH$ '),100)
      CALL SETCLR(%REF('MAGENTA'))
      CALL GRAF(-1.25,%REF('SCALE'),1.25,ZMAX,%REF('SCALE'),0)
      CALL HEIGHT(0.15)
      CALL MESSAG(%REF('IFREQ = $'),100,0.5,-0.5)
      CALL INTNO(IFREQ,%REF('ABUT'),%REF('ABUT'))
      CALL MESSAG(%REF('IX = $'),100,0.5,-0.8)
      CALL INTNO(IX,%REF('ABUT'),%REF('ABUT'))
```

```
DO M = 1,NMODES,NDM
```

```

      LOC = M + (IFREQ-1)*(NMODES) + (IX-1)*(NMODES)*(NFREQ)

      READ(15'LOC) K(M),(WMINT(KK),KK=1,NZZ)

      WRITE(20,101) IX,IFREQ,M,K(M)
      WMAX = 1.0E-12
      DO IZ = 1,NZZ
        IF( ABS(WMINT(IZ)).GT.WMAX) WMAX=ABS(WMINT(IZ))
        WRITE(20,*) IZ,WMINT(IZ)
      END DO
      WT = 9.0/(9.0+(M-1)**2)
      WRITE(*,*) 'WMAX,WT = ',WMAX,WT
      DO IZ = 1,NZZ
        WMINT(IZ) = WT*WMINT(IZ)/WMAX
      END DO

      CALL SETCLR(%REF('YELLOW'))
      CALL CURVE( WMINT,Z,NZZ,0)

      END DO
      CALL ENDPL(0)
    END DO
  END DO

  CALL DONEPL

101  FORMAT( // ' IX,IFREQ,MODE ', 3I5/' K(M) = ',G16.5//
1    2X,'IZ',',', W(IX,IFREQ,M) '///)
100  FORMAT(1X,I5,G16.5)

  STOP ' END MODEL1_LOOK'
  END

```

PROGRAM MODEL1_CONTOUR

```

*****
C
C PROGRAM          MODEL1_CONTOUR
C
C PURPOSE          COMPUTE AND PLOT MODEL FIELDS, CONTOURING VIA DISPLA
C
C AUTHOR           K.D. Saunders (NORDA, Code 331)
C
C
C HISTORY          10/28/88      Program written based on FRACTAL code
C
C                               Only the displacement field is plotted
C                               at present. Later modifications will
C                               allow plotting temperature, salinity
C                               and their deviation fields.
C
C                               10/31/88      Modified to used direct writes to pixel
C                               locations on screen (to avoid full
C                               memory crashes.)
C
C                               11/5/88      GPX PROGRAM (PLOT_MODEL_FIELDS) was
C                               modified to interpolate the data and
C                               produce contours using the DISSPLA
C                               plotting package.
C
C INPUT
C -----
C Unit  Filename      Type           Contents
C -----
C  5     SYS$INPUT      KEYBOARD      Control Information
C 12     MODEL1.DAT     « direct »    Data to plot
C 13     MODEL1.AUX     « ascii »     Descriptor for MODEL1.DAT
C -----
C
C OUTPUT
C -----
C Unit  Filename      Type           Contents
C -----
C  6     SYS$OUTPUT     «cntl window» Program/control information
C NA     SYS$WORKSTATION «plot window»  Color plots
C -----
C
C Notes:
C           This program is designed to be used on the DEC GKS
C           Workstation 2000. It will not work on any other system
C
*****

```

IMPLICIT NONE

INTEGER*4 MAX
PARAMETER (MAX=500)

CHARACTER*2 PLOT TYPE
CHARACTER*80 INFILE,AUXFILE,DATFILE

REAL*4 X,Y,X_SIDE,Y_SIDE,
 X0,Y0,YMAX,

```

1      XFACT,YFACT,ZP,ZM,XAR(MAX*MAX),YAR(MAX*MAX),
1      ZAR(MAX*MAX),ZMAT(200,200)

```

```

      CHARACTER*80  XTITLE  //'X$'//,
1      YTITLE  //'Y$'//,
2      ZTITLE,
3      LINE,
4      CBUFF(20)

```

```

1      REAL*4      J,K,R,RED,GREEN,BLUE,DEG,SX,
      XLAST,XB,XE,YLAST,YB,YE,Z

```

```

1      INTEGER*4   I_INDEX(10,10,10),NPTX,NPTY,
2      VCM_SIZE,VD_ID,WD_ID,WD_ID2,N,I,VCM_ID,
      ILAST,NITER,IXB,IXE,IYB,IYE,IM,L

```

```

      DATA J/17.0/
      DATA K/16/
      DATA VCM_SIZE/130/

```

```

1      REAL*4      T(MAX),    S(MAX),    ZZ(MAX),    BV(MAX),
2      ZD(MAX,MAX),DT,DX,DZZ,WM,WP,WT,HM,HP,HT,
3      T_AV(MAX),S_AV(MAX),TD(MAX,MAX),SD(MAX,MAX),
      SVEL,XMAX,ZMAX,TZ,ZINCR,ZDMAX,ZDMIN,ZDZ,DX,DY

```

```

      INTEGER*4    NXX,NZZ,NT,IX,IZ,NVARS,NCONT,IXX,IYY

```

```

      CHARACTER*60  TITLE1,TITLE2

```

```

      REAL*4      DUMMY(100000)
      COMMON      DUMMY

```

```

      EQUIVALENCE  (TD(1,1),XAR(1)),(SD(1,1),YAR(1))

```

```

C*****
C      GET INPUT DATA GENERATED BY MODEL1
C*****

```

```

1      FORMAT(A)

      TYPE *, ' ENTER INPUT FILE NAME '
      INFILE = ' '
      ACCEPT 1, INFILE
      DO N = 80,1,-1
          L = N
          IF(INFILE(N:N) .NE. ' ') GOTO 1000
      END DO
      CONTINUE

```

```

1000    AUXFILE = INFILE(1:L)//'.AUX'
      DATFILE = INFILE(1:L)//'.DAT'

```

```

      TYPE *,AUXFILE,DATFILE

```

```

      OPEN (FILE=AUXFILE,UNIT=13,STATUS='OLD',DISP='KEEP')

```

```

      READ(13,1) LINE
      READ(13,1) LINE
      READ(13,1) LINE
      CALL PARSE(LINE,CBUFF,NVARS)

```

```

READ(CBUFF(2),*) NT

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NXX

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NZZ

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DT

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DXX

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DZZ

TYPE *, ' NT,NX,NZ,DT,DX,DZ '
TYPE *, NT,NXX,NZZ
TYPE *, DT,DXX,DZZ

XMAX = (NXX-1)*DXX
ZMAX = (NZZ-1)*DZZ

1 OPEN (FILE=DATFILE,UNIT=12,STATUS='OLD',DISP='KEEP',
      ACCESS='DIRECT',FORM='UNFORMATTED',RECL=4*NZZ)

DO IX = 1,NXX
  READ(12,IX)(ZD(I,IX),T(I),S(I),BV(I),I=1,NZZ)
  DO IZ = 1,NZZ
    TD(IZ,IX) = T(IZ)
    SD(IZ,IX) = S(IZ)
    T_AV(IZ) = T_AV(IZ) + T(IZ)
    S_AV(IZ) = S_AV(IZ) + S(IZ)
  END DO
END DO

DO IZ = 1,NZZ
  T_AV(IZ) = T_AV(IZ)/NXX
  S_AV(IZ) = S_AV(IZ)/NXX
END DO

TYPE *, ' ENTER PLOT TYPE : ZD,TD,SD,SV '
ACCEPT 1, PLOT_TYPE

DO IX = 1,NXX
  DO IZ = 1, NZZ
    IF( PLOT_TYPE .NE. 'SV') THEN
      TD(IZ,IX) = TD(IZ,IX) - T_AV(IZ)
      SD(IZ,IX) = SD(IZ,IX) - S_AV(IZ)
    END IF
  END DO
END DO

TITLE1 = 'Vertical Displacement$'
ZINCR = 2.5

```

```

1 IF( PLOT_TYPE .EQ. 'TD' .OR. PLOT_TYPE .EQ. 'SD' .OR.
    PLOT_TYPE .EQ. 'SV' ) THEN
    DO IX = 1,NXX
        DO IZ = 1,NZZ

            IF( PLOT_TYPE .EQ. 'TD' ) THEN
                TITLE1 = 'Temperature Anomaly$'
                ZD(IZ,IX) = TD(IZ,IX)
                ZINCR = 0.05
            END IF

            IF( PLOT_TYPE .EQ. 'SD' ) THEN
                TITLE1 = 'Salinity Anomaly$'
                ZD(IZ,IX) = SD(IZ,IX)
                ZINCR = 0.005
            END IF

            IF( PLOT_TYPE .EQ. 'SV' ) THEN
                TITLE1 = 'Sound Velocity Anomaly$'
                ZD(IZ,IX) = SVEL(SD(IZ,IX),TD(IZ,IX),
                                (IZ-1)*DZZ )
                IF(IX .EQ. 1) S_AV(IZ) = 0
                S_AV(IZ) = S_AV(IZ) + ZD(IZ,IX)
                ZINCR = 0.2
            END IF

        END DO
    END DO

    IF( PLOT_TYPE .EQ. 'SV' ) THEN
        DO IZ = 1,NZZ
            S_AV(IZ) = S_AV(IZ)/NXX
        END DO
        DO IX = 1,NXX
            DO IZ = 1,NZZ
                ZD(IZ,IX) = ZD(IZ,IX) - S_AV(IZ)
            END DO
        END DO
    END IF

    END IF

    ZDMAX = -1.0E10
    ZDMIN = 1.0E10

    DO IX = 1,NXX
        DO IZ = 1,NZZ/2

            TZ = ZD(IZ,IX)
            ZD(IZ,IX) = ZD(NZZ-IZ+1,IX)
            ZD(NZZ-IZ+1,IX) = TZ

            IF(ZDMAX .LT. TZ ) ZDMAX = TZ
            IF(ZDMAX .LT. ZD(IZ,IX) ) ZDMAX = ZD(IZ,IX)
            IF(ZDMIN .GT. TZ ) ZDMIN = TZ
            IF(ZDMIN .GT. ZD(IZ,IX) ) ZDMIN = ZD(IZ,IX)

        END DO
    END DO

```

```

- NCONT = (ZDMAX-ZDMIN)/ZINCR
TYPE *, ' ZDMAX,ZDMIN,ZINCR,NCONT', ZDMAX,ZDMIN,ZINCR,NCONT
IF (NCONT .GT. 50) THEN
    ZINCR = (ZDMAX - ZDMIN)/49
    NCONT = 25
END IF

```

C
C
C
C

PLOTTING SECTION

```

NPTX = 200
NPTY = 200

```

```

X_SIDE = (NXX-1)*DXX
Y_SIDE = (NZZ-1)*DZZ

```

```

XMAX = X_SIDE
YMAX = Y_SIDE

```

```

DX      = X_SIDE/NPTX
DY      = Y_SIDE/NPTY

```

```

ZDMAX = -1.0E10
ZDMIN = 1.0E10

```

```

DO IXX = 1,NPTX
    X = (IXX-1)*DX
    IX = X/DXX + 1
    IF(IX .LT. 1) IX = 1
    IF(IX .GT. NXX) IX = NXX

    WM = ABS((IX-1)*DXX - X)/DXX
    WP = ABS(IX*DXX - X)/DXX
    WT = WM+WP
    WM = 1.0 - WM/WT
    WP = 1.0 - WP/WT

```

```

DO IYY = 1,NPTY
    Y = (NPTY-(IYY-1))*DY

    IZ = (YMAX-Y)/DZZ + 1
    IF(IZ .LT. 1) IZ = 1
    IF(IZ .GT. NZZ) IZ = NZZ
    HM = ABS((IZ-1)*DZZ - (YMAX-Y))/DZZ
    HP = ABS( IZ*DZZ      - (YMAX-Y))/DZZ
    HT = HM + HP
    HM = 1.0 - HM/HT
    HP = 1.0 - HP/HT

```

```

IM = IZ-1
IF(IM.LE.1)IM = 1

IF(IX .GT. 1) THEN

```

```

                ZP = WP*ZD(IZ,IX) + WM*ZD(IZ,IX-1)
                ZM = WP*ZD(IM,IX) + WM*ZD(IM,IX-1)
            ELSE
                ZP = ZD(IZ,IX)
                ZM = ZD(IM,IX)
            END IF

            ZMAT(IXX,IYY) = (HP*ZP + HM*ZM)/(HM+HP)
            IF( ZDMAX .LT. ZMAT(IXX,IYY)) ZDMAX = ZMAT(IXX,IYY)
            IF( ZDMIN .GT. ZMAT(IXX,IYY)) ZDMIN = ZMAT(IXX,IYY)

END DO
END DO
! Y
! X

NCONT = (ZDMAX-ZDMIN)/ZINCR

TYPE *, ' ZDMAX,ZDMIN,ZINCR,NCONT', ZDMAX,ZDMIN,ZINCR,NCONT

IF (NCONT .GT. 50) THEN
    ZINCR = (ZDMAX - ZDMIN)/49
    NCONT = 50
END IF

CALL COMPRS
CALL BCOMON(100000)
CALL PAGE( 11.0,8.5)
CALL SCMPX
CALL AREA2D(8.0,4.0)
CALL HEADIN(%REF('MODEL OCEAN SIMULATION$'),100,1.2,2)
CALL HEADIN(%REF(TITLE1),100,1.0,2)
CALL MESSAG(%REF('Contour interval = $'),100,0.0,-1.0)
CALL REALNO(ZINCR,3,%REF('ABUT'),%REF('ABUT'))
CALL YNAME(%REF('DEPTH: m$'),100)
CALL XNAME(%REF('X-DISTANCE: km $'),100)
CALL GRAF(0.0,10.0,XMAX,ZMAX,-1000.0,0.0)
CALL FRAME
CALL CONMAK(ZMAT,200,200,ZINCR)
CALL CONMIN(1.0)
CALL CONDIG(1)
CALL CONLIN(0,%REF('MYCON'),%REF('LABELS'),2,10)
CALL CONLIN(1,%REF('MYCON'),%REF('LABELS'),1,8)
CALL CONTUR(2,%REF('LABELS'),%REF('DRAW'))
CALL ENDPL(1)
CALL DONEPL

STOP
END

SUBROUTINE MYCON(RARRAY,IARRAY)
REAL*4      RARRAY(1)
INTEGER*4    IARRAY(1)
IF( RARRAY(1) .LT. 0) CALL DOT
IF( RARRAY(1) .LT. 0) IARRAY(1) = 1
IF( RARRAY(1) .GE. 0) CALL RESET(%REF('DOT'))
IF( RARRAY(1) .GE. 0) IARRAY(1) = 3
RETURN
END

```

```

      REAL FUNCTION SVEL(S,T,P0)
C *****
C SOUND SPEED SEAWATER CHEN AND MILLERO 1977,JASA,62,1129-1135
C UNITS:
C      PRESSURE           P0           DECIBARS
C      TEMPERATURE        T           DEG CELSIUS (IPSS-68)
C      SALINITY            S           (IPSS-78)
C      SOUND SPEED        SVEL        METERS/SECOND
C CHECKVALUE: SVEL=1731.995 M/S, S=40 (IPSS-78),T=40 DEG C,P=10000 DBAR
C
      EQUIVALENCE (A0,B0,C0),(A1,B1,C1),(A2,C2),(A3,C3)
C
C SCALE PRESSURE TO BARS
      P=P0/10.
C*****
      SR = SQRT(ABS(S)).
C S**2 TERM
      D = 1.727E-3 - 7.9836E-6*P
C S**3/2 TERM
      B1 = 7.3637E-5 +1.7945E-7*T
      B0 = -1.922E-2 -4.42E-5*T
      B = B0 + B1*P
C S**1 TERM
      A3 = (-3.389E-13*T+6.649E-12)*T+1.100E-10
      A2 = ((7.988E-12*T-1.6002E-10)*T+9.1041E-9)*T-3.9064E-7
      A1 = (((-2.0122E-10*T+1.0507E-8)*T-6.4885E-8)*T-1.2580E-5)*T
      X   +9.4742E-5
      A0 = (((-3.21E-8*T+2.006E-6)*T+7.164E-5)*T-1.262E-2)*T
      X   +1.389
      A = ((A3*P+A2)*P+A1)*P+A0
C S**0 TERM
      C3 = (-2.3643E-12*T+3.8504E-10)*T-9.7729E-9
      C2 = (((1.0405E-12*T-2.5335E-10)*T+2.5974E-8)*T-1.7107E-6)*T
      X   +3.1260E-5
      C1 = (((-6.1185E-10*T+1.3621E-7)*T-8.1788E-6)*T+6.8982E-4)*T
      X   +0.153563
      C0 = (((3.1464E-9*T-1.47800E-6)*T+3.3420E-4)*T-5.80852E-2)*T
      X   +5.03711)*T+1402.388
      C = ((C3*P+C2)*P+C1)*P+C0
C SOUND SPEED RETURN
      SVEL = C + (A+B*SR+D*S)*S
      RETURN
      END

```

PROGRAM MODEL1 EXPORT

```

C*****
C
C PROGRAM          MODEL1_EXPORT
C
C PURPOSE          COMPUTE AND REFORMAT MODEL FIELD DATA FOR EXPORTING
C                  TO PC
C
C AUTHOR           K.D. Saunders (NOARL, Code 331)
C
C-----
C-----
C
C INPUT
C-----
C Unit  Filename          Type          Contents
C-----
C   5    SYS$INPUT         KEYBOARD      Control Information
C  12    MODEL1.DAT        « direct »   Data to plot
C  12    MODEL1.UV         « direct »   UV data to plot
C  13    MODEL1.AUX        « ascii  »   Descriptor for MODEL1.DAT
C-----
C
C OUTPUT
C-----
C Unit  Filename          Type          Contents
C-----
C   6    SYS$OUTPUT        «cntl window» Program/control information
C   8    MODEL1.PC         «ASCII»      DATA FOR PC PLOT - internally
C                               documented
C*****

```

IMPLICIT NONE

```

INTEGER*4      MAX, IT
PARAMETER      (MAX=500)

```

```
CHARACTER*2      PLOT_TYPE
CHARACTER*8      PBOT,PTOP
CHARACTER*80     LABEL,LINE,CBUFF(20)
```

```

      REAL*4      X,Y,X SIDE,Y SIDE,
1      XMAX,YMAX,X0,Y0,DX,DY,ZP,ZM

```

```
REAL*4      J,K,R,XLAST,XB,XE,YLAST,YB,YE,Z
```

```

1      REAL*4      T(MAX),      S(MAX),      ZZ(MAX),      BV(MAX),
2                  ZD(MAX,MAX),DT,DXX,DZZ,WM,WP,WT,HM,HP,HT,
3                  T_AV(MAX),S_AV(MAX),TD(MAX,MAX),SD(MAX,MAX),
4                  SVEL,U(MAX,MAX),V(MAX,MAX),W(MAX,MAX),
5                  UT(MAX),VT(MAX),WTT(MAX)

```

```
INTEGER*4      NXX,NZZ,NT,IX,IZ,NVARS,LOC,ILOC,IBUFF(80),ITMP
```

INTEGER*4 NPTX,NPTY,N,I,IM,INX,INY

DATA J/17.0/
DATA K/16/

EQUIVALENCE $(ZD(1,1), U(1,1)), (TD(1,1), V(1,1)), (SD(1,1), W(1,1))$

EQUIVALENCE (UT(1),T(1)),(VT(1),S(1)),(WTF(1),ZZ(1))

C*****
C GET INPUT DATA GENERATED BY MODEL1 *
C*****

1 FORMAT(A)

1 OPEN (FILE='MODEL1.AUX',UNIT=13,
STATUS='OLD',DISP='KEEP')

READ(13,1) LINE
READ(13,1) LINE
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NT

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NXX

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NZZ

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DT

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DXX

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DZZ

TYPE *, ' NT,NX,NZ,DT,DX,DZ '
TYPE *, NT,NXX,NZZ
TYPE *, DT,DXX,DZZ
TYPE *, ' ENTER PLOT TYPE : ZD,TD,SD,SV '

ACCEPT 1, PLOT_TYPE

TYPE *, ' ENTER THE TIME ITERATION NUMBER '
ACCEPT *, IT

1 IF(PLOT_TYPE .NE. 'U' .AND. PLOT_TYPE .NE. 'V' .AND.
PLOT_TYPE .NE. 'W') THEN

1 OPEN (FILE='MODEL1.DAT',UNIT=12,
2 STATUS='OLD',DISP='KEEP',ACCESS='DIRECT',
FORM='UNFORMATTED',RECL=4*NZZ)

DO IX = 1,NXX
LOC = NXX*(IT-1)+IX
READ(12'LOC)(ZD(I,IX),T(I),S(I),BV(I),I=1,NZZ)
DO IZ = 1,NZZ

```

        TD(IZ,IX) = T(IZ)
        SD(IZ,IX) = S(IZ)
        T_AV(IZ) = T_AV(IZ) + T(IZ)
        S_AV(IZ) = S_AV(IZ) + S(IZ)
    END DO
END DO

DO IZ = 1,NZZ
    T_AV(IZ) = T_AV(IZ)/NXX
    S_AV(IZ) = S_AV(IZ)/NXX
END DO

DO IX = 1,NXX
    DO IZ = 1, NZZ
        IF( PLOT_TYPE .NE. 'SV') THEN
            TD(IZ,IX) = TD(IZ,IX) - T_AV(IZ)
            SD(IZ,IX) = SD(IZ,IX) - S_AV(IZ)
        END IF
    END DO
END DO

```

ELSE

```

1  OPEN (FILE='MODEL1.UV',UNIT=12,
2      STATUS='OLD',DISP='KEEP',ACCESS='DIRECT',
      FORM='UNFORMATTED',RECL=3*NZZ)

```

```

    DO IX = 1,NXX
        LOC = NXX*(IT-1) + IX
        READ(12,LOC)(UT(I),VT(I),WTT(I),I=1,NZZ)
        DO IZ = 1,NZZ
            U(IZ,IX)=UT(IZ)
            V(IZ,IX)=VT(IZ)
            W(IZ,IX)=WTT(IZ)
        END DO
    END DO

```

```

    IF( PLOT_TYPE .EQ. 'V') THEN

```

```

        DO IX = 1,NXX
            DO IZ = 1,NZZ
                ZD(IZ,IX) = V(IZ,IX)
            END DO
        END DO

```

```

    END IF

```

```

    IF( PLOT_TYPE .EQ. 'W') THEN

```

```

        DO IX = 1,NXX
            DO IZ = 1,NZZ
                ZD(IZ,IX) = W(IZ,IX)
            END DO
        END DO

```

```

    END IF

```

```

END IF

```

```

OPEN (FILE='MODEL1.PC',UNIT=8,DISP='KEEP',STATUS='NEW')

```

```

IF( PLOT_TYPE .EQ. 'TD' .OR. PLOT_TYPE .EQ. 'SD' .OR.

```

```

1      PLOT_TYPE .EQ. 'SV' ) THEN
      DO IX = 1,NXX
      DO IZ = 1,NZZ
      IF( PLOT_TYPE .EQ. 'TD')
1      ZD(IZ,IX) = TD(IZ,IX)
      IF( PLOT_TYPE .EQ. 'SD')
1      ZD(IZ,IX) = SD(IZ,IX)
      IF( PLOT_TYPE .EQ. 'SV' ) THEN
1      ZD(IZ,IX) = SVEL(SD(IZ,IX),TD(IZ,IX),
      (IZ-1)*DZZ )
      IF(IX .EQ. 1) S_AV(IZ) = 0
      S_AV(IZ) = S_AV(IZ) + ZD(IZ,IX)
      END IF

```

```

      END DO
END DO

```

```

IF( PLOT_TYPE .EQ. 'SV' ) THEN
DO IZ = 1,NZZ
  S_AV(IZ) = S_AV(IZ)/NXX
END DO

```

```

DO IX = 1,NXX
  DO IZ = 1,NZZ
    ZD(IZ,IX) = ZD(IZ,IX) - S_AV(IZ)
  END DO
END DO
END IF

```

```

END IF

```

```

C*****
C      GET INPUT DATA FOR SCREEN CONTROL      *
C*****

```

```

NPTX= 640
NPTY= 480

```

```

X_SIDE = NXX*DXX
Y_SIDE = NZZ*DZZ

```

```

DX      = X_SIDE/NPTX
DY      = Y_SIDE/NPTY

```

```

X0 = 0
Y0 = 0

```

```

XMAX = X_SIDE - DX
YMAX = Y_SIDE - DY

```

```

IF (PLOT_TYPE .EQ. 'ZD') THEN
  LABEL = 'VERTICAL DISPLACEMENT (m)'
  PBOT = '-10.0'
  PTOP = '10.0'
END IF

```

```

IF( PLOT_TYPE .EQ. 'TD') THEN
  LABEL = 'TEMPERATURE ANOMALY (°C)'
  PBOT = '-1.0'
  PTOP = '1.0'
END IF
IF( PLOT_TYPE .EQ. 'SD') THEN
  LABEL = 'SALINITY ANOMALY (psu)'
END IF
IF( PLOT_TYPE .EQ. 'SV') THEN
  LABEL = 'SOUND VELOCITY ANOMALY (m/s)'
END IF
IF( PLOT_TYPE .EQ. 'U') THEN
  LABEL = 'HORIZONTAL -U- VELOCITY (m/s)'
END IF
IF( PLOT_TYPE .EQ. 'V' ) THEN
  LABEL = 'HORIZONTAL -V- VELOCITY (m/s)'
END IF
IF( PLOT_TYPE .EQ. 'W' ) THEN
  LABEL = 'VERTICAL VELOCITY (m/s)'
END IF

```

```

2000 WRITE(8,2000) LABEL,NPTX,NPTY,PTOP,PBOT
      FORMAT( ' MODEL1 - SIMULATION PLOT DATA '/
1       A80/
1       ' NPTX,',I10/
2       ' NPTY,',I10/
3       A10/
4       A10)

```

```
DO INX = 1,NPTX
```

```
  X = DX*(INX-1)
```

```
  IX = X/DXX + 1
```

```
  IF(IX .LT. 1) IX = 1
```

```
  IF(IX .GT. NXX) IX = NXX
```

```
  WM = ABS((IX-1)*DXX - X)/DXX
```

```
  WP = ABS(IX*DXX - X)/DXX
```

```
  WT = WM+WP
```

```
  WM = 1.0 - WM/WT
```

```
  WP = 1.0 - WP/WT
```

```
  XLAST = X
```

```
  YLAST = Y0
```

```
  XB = X
```

```
  XE = X
```

```
  YB = Y0
```

```
  YE = Y0
```

```
DO INY = 1,NPTY
```

```
  Y = DY*(INY-1)
```

```
  IZ = (YMAX-Y)/DZZ + 1
```

```
  IF(IZ .LT. 1) IZ = 1
```

```
  IF(IZ .GT. NZZ) IZ = NZZ
```

```
  HM = ABS((IZ-1)*DZZ - (YMAX-Y))/DZZ
```

```
  HP = ABS( IZ*DZZ - (YMAX-Y))/DZZ
```

```
  HT = HM + HP
```

```

HM = 1.0 - HM/HT
HP = 1.0 - HP/HT

IM = IZ-1
IF(IM.LE.1)IM = 1

IF(IX .GT. 1) THEN
    ZP = (WP*ZD(IZ,IX) + WM*ZD(IZ,IX-1))/(WM+WP)
    ZM = (WP*ZD(IM,IX) + WM*ZD(IM,IX-1))/(WM+WP)
ELSE
    ZP = ZD(IZ,IX)
    ZM = ZD(IM,IX)
END IF

Z = (HP*ZP + HM*ZM)/(HM+HP)

```

```

IF (PLOT TYPE .EQ. 'ZD') THEN
    I = 5.0*Z
END IF
IF( PLOT TYPE .EQ. 'TD') THEN
    I = 50.*Z
END IF
IF( PLOT TYPE .EQ. 'SD') THEN
    I = 500.*Z
END IF
IF( PLOT TYPE .EQ. 'SV') THEN
    I = 50.0*Z
END IF
IF( PLOT TYPE .EQ. 'U') THEN
    I = 500.*Z
END IF
IF( PLOT TYPE .EQ. 'V' ) THEN
    I = 500.*Z
END IF
IF( PLOT TYPE .EQ. 'W' ) THEN
    I = 5000.*Z
END IF

```

```

IF(I.LE.0) I = 100 + I
I = ABS(I)
I = I - (I/100)*100
IF(I .LT. 1 ) I = 0
IF(I .GT. 99 ) I = 99

```

```

ILOC = ILOC + 1
IBUFF(ILOC) = I
IF( ILOC .GE. 39) THEN
    WRITE(8,2010) (IBUFF(ITMP),ITMP=1,39)
    ILOC = 0
END IF

```

```

END DO                                ! Y
END DO                                ! X

```

2010 FORMAT(1X,39I2)

```

PAUSE
STOP
END

```

REAL FUNCTION SVEL(S,T,P0)

C *****

C SOUND SPEED SEAWATER CHEN AND MILLERO 1977,JASA,62,1129-1135

C UNITS:

```

C      PRESSURE      P0      DECIBARS
C      TEMPERATURE   T      DEG CELSIUS (IPSS-68)
C      SALINITY      S      (IPSS-78)
C      SOUND SPEED   SVEL    METERS/SECND
C CHECKVALUE: SVEL=1731.995 M/S, S=40 (IPSS-78), T=40 DEG C, P=10000 DBAR
C
C      EQUIVALENCE (A0,B0,C0),(A1,B1,C1),(A2,C2),(A3,C3)
C
C      SCALE PRESSURE TO BARS
C      P=P0/10.
C*****
C      SR = SQRT(ABS(S))
C S**2 TERM
C      D = 1.727E-3 - 7.9836E-6*P
C S**3/2 TERM
C      B1 = 7.3637E-5 +1.7945E-7*T
C      B0 = -1.922E-2 -4.42E-5*T
C      B = B0 + B1*P
C S**1 TERM
C      A3 = (-3.389E-13*T+6.649E-12)*T+1.100E-10
C      A2 = ((7.988E-12*T-1.6002E-10)*T+9.1041E-9)*T-3.9064E-7
C      A1 = (((-2.0122E-10*T+1.0507E-8)*T-6.4885E-8)*T-1.2580E-5)*T
C      X   +9.4742E-5
C      A0 = (((-3.21E-8*T+2.006E-6)*T+7.164E-5)*T-1.262E-2)*T
C      X   +1.389
C      A = ((A3*P+A2)*P+A1)*P+A0
C S**0 TERM
C      C3 = (-2.3643E-12*T+3.8504E-10)*T-9.7729E-9
C      C2 = (((1.0405E-12*T-2.5335E-10)*T+2.5974E-8)*T-1.7107E-6)*T
C      X   +3.1260E-5
C      C1 = (((-6.1185E-10*T+1.3621E-7)*T-8.1788E-6)*T+6.8982E-4)*T
C      X   +0.153563
C      C0 = (((3.1464E-9*T-1.47800E-6)*T+3.3420E-4)*T-5.80852E-2)*T
C      X   +5.03711)*T+1402.388
C      C = ((C3*P+C2)*P+C1)*P+C0
C SOUND SPEED RETURN
C      SVEL = C + (A+B*SR+D*S)*S
C      RETURN
C      END

```

PROGRAM MODEL1_ENERGY

```

C*****
C
C PROGRAM      MODEL1_ENERGY
C
C PURPOSE      COMPUTE ENERGY DENSITIES AND PLOT MODEL FIELDS
C
C AUTHOR       K.D. Saunders (NOARL, Code 331)
C
C INPUT
C-----
C Unit  Filename      Type      Contents
C-----
C  5     SYS$INPUT     KEYBOARD  Control Information
C 12     MODEL1.DAT    * direct *  Data to plot
C 14     MODEL1.UV     * direct *  UVW data to plot
C 13     MODEL1.AUX    * ascii  *  Descriptor for MODEL1.DAT
C-----
C
C OUTPUT
C-----
C Unit  Filename      Type      Contents
C-----
C  6     SYS$OUTPUT    * cntl window *  Program/control information
C NA     POPFIL.DAT    * DISSPLA META FILE *  Plot information
C  8     ENERGY.LIS   * ASCII *  Summary information
C-----
C*****

```

INTEGER*4 MAX
PARAMETER (MAX=500)

CHARACTER*80 LINE,CBUFF(20),ANS

INTEGER*4 N,I,NXX,NZZ,NT,IX,IZ,NVARS,IPAK(2000)

REAL*4	T(MAX), S(MAX), BV(MAX),
1	ZD(MAX,MAX),DT,DXX,DZZ,
2	TD(MAX,MAX),SD(MAX,MAX),
3	U(MAX,MAX),V(MAX,MAX),W(MAX,MAX),PI,
4	Z2(MAX),U2(MAX),V2(MAX),EK_HOR(MAX),EK_W(MAX),
5	E_POT(MAX),X(MAX),EMAX,XMAX,W2(MAX),EK_AV,EP_AV,
6	UV_NORM,W_NORM,E_POT_Z(MAX),EK_HOR_Z(MAX),
7	EK_W_Z(MAX),ETOT(MAX),
8	J,K,R,RED,GREEN,
9	Z(MAX),EWEK_RATIO(MAX),
A	EPEK_RATIO(MAX)

EQUIVALENCE (SD(1,1),U(1,1)),(TD(1,1),V(1,1))

DATA J/17.0/
DATA K/16/

```

C*****
C      GET INPUT DATA GENERATED BY MODEL1
C*****

```

1 FORMAT(A)

PI =4.0*ATAN(1.0)
CR = (2*PI/3600.0)**2
RHO = 1025.0

! NOMINAL DENSITY SEAWATER

UV_NORM = 0
W_NORM = 0
N_NORM = 0

OPEN (FILE='MODEL1.AUX',UNIT=13,STATUS='OLD',DISP='KEEP')
OPEN (FILE='ENERGY.LIS',UNIT=8,STATUS='NEW',DISP='KEEP')

TYPE *, ' IS THIS A GENERIC GM BV PROFILE ?'
ACCEPT 1, ANS

READ(13,1) LINE
READ(13,1) LINE
READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NT

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NXX

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) NZZ

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DT

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DXX

XMAX = DXX*(NXX-1)

READ(13,1) LINE
CALL PARSE(LINE,CBUFF,NVARS)
READ(CBUFF(2),*) DZZ

TYPE *, ' NT,NX,NZ,DT,DX,DZ '

TYPE *, NT,NXX,NZZ

TYPE *, DT,DXX,DZZ

TYPE *, ' ***** PLEASE WAIT - READING IN DATA *****'

1 OPEN (FILE='MODEL1.DAT',UNIT=12,STATUS='OLD',DISP='KEEP',
ACCESS='DIRECT',FORM='UNFORMATTED',RECL=4*NZZ)
1 OPEN (FILE='MODEL1.UV',UNIT=14,STATUS='OLD',DISP='KEEP',
ACCESS='DIRECT',FORM='UNFORMATTED',RECL=3*NZZ)

C *****
C INITIALIZE VARIABLES *
C *****

EMAX = 0
EKAV = 0

```

EKHAV= 0
EKWAV= 0
EPAV = 0
ZAV = 0
N NORM = 0
DO IZ= 1,MAX
    E_POT_Z(IZ) = 0
    EK_HOR_Z(IZ) = 0
    EK_W_Z(IZ) = 0
END DO

```

```

C *****
C READ & COMPUTE VARIABLES *
C *****

```

```

DO IX = 1,NXX
    X(IX) = (IX-1)*DXX
    Z2(IX) = 0
    U2(IX) = 0
    V2(IX) = 0
    W2(IX) = 0
    E_POT(IX) = 0
    EK_HOR(IX) = 0
    EK_W(IX) = 0
    READ(12,IX)(ZD(I,IX),T(I),S(I),BV(I),I=1,NZZ)
    READ(14,IX)(U(I,IX),V(I,IX),W(I,IX),I=1,NZZ)

```

```

    IF( ANS .EQ. 'YES') THEN
        DO I = 1,NZZ
            BV(I) = 3.0*EXP( -DZZ*(I-1)/1300.0)
            IF( I.LT.4) BV(I) = 2.99
        END DO
    END IF

```

```

NZTOT = 0

```

```

DO IZ = 2,NZZ
    UV_NORM = UV_NORM + 0.5*(ABS(U(IZ,IX))+ABS(V(IZ,IX)))
    W_NORM = W_NORM + ABS(W(IZ,IX))
    Z_NORM = Z_NORM + ABS(ZD(IZ,IX))
    ZAV = ZAV + ZD(IZ,IX)
    N_NORM = N_NORM + 1

```

```

    Z2(IX) = Z2(IX) + ZD(IZ,IX)**2
    U2(IX) = U2(IX) + U(IZ,IX)**2
    V2(IX) = V2(IX) + V(IZ,IX)**2
    W2(IX) = W2(IX) + W(IZ,IX)**2

```

```

1      E_POT(IX)=E_POT(IX) +
        0.5*CR*Bv(IZ)*Bv(IZ)*ZD(IZ,IX)**2

```

```

1      EK_HOR_Z(IZ) = 0.5*RHO*
1      (U(IZ,IX)**2+V(IZ,IX)**2)+
        EK_HOR_Z(IZ)

```

```

1      EK_W_Z(IZ) = 0.5*RHO*
        W(IZ,IX)**2 + EK_W_Z(IZ)

```

```

1      E_POT_Z(IZ) = E_POT_Z(IZ) +
1      0.5*RHO*
        CR*Bv(IZ)*Bv(IZ)*ZD(IZ,IX)**2

```

```

NZTOT= NZTOT + 1

```

```

END DO

```

```

EK_HOR(IX) = 0.5*(U2(IX) + V2(IX))
EK_W(IX)   = 0.5*W2(IX)

```

```

EK_HOR(IX) = EK_HOR(IX)*DZZ*RHO/NZZ
EK_W(IX)   = EK_W(IX)*DZZ*RHO/NZZ
E_POT(IX)  = E_POT(IX)*DZZ*RHO/NZZ

```

```

U2(IX) = U2(IX)/NZTOT
V2(IX) = V2(IX)/NZTOT
W2(IX) = W2(IX)/NZTOT
Z2(IX) = Z2(IX)/NZTOT

```

```

IF( EK_HOR(IX) .GT. EMAX) EMAX = EK_HOR(IX)
IF( EK_W(IX)   .GT. EMAX) EMAX = EK_W(IX)
IF( E_POT(IX)  .GT. EMAX) EMAX = E_POT(IX)
EKHAV = EKHAV + EK_HOR(IX)
EKWAV = EKWAV + EK_W(IX)
EKAV = EKAV + EK_HOR(IX) + EK_W(IX)
EPAV = EPAV + E_POT(IX)

```

```

END DO

```

```

ZAV = ZAV/(NXX*NZZ)
U2AV = 0
V2AV = 0
W2AV = 0
Z2AV = 0
DO IX = 1, NXX
    U2AV = U2AV + U2(IX)
    V2AV = V2AV + V2(IX)
    W2AV = W2AV + W2(IX)
    Z2AV = Z2AV + Z2(IX)

```

```

END DO
U2AV = U2AV/NXX
V2AV = V2AV/NXX
W2AV = W2AV/NXX
Z2AV = Z2AV/NXX

```

```

UVRMS = SQRT(U2AV+V2AV)
WRMS = SQRT(W2AV)
ZRMS = SQRT(ABS(Z2AV - ZAV**2))

```

```

UV_NORM = UV_NORM/N_NORM
W_NORM = W_NORM/N_NORM
Z_NORM = Z_NORM/N_NORM

```

```

EKAV = EKAV/NXX
EPAV = EPAV/NXX
EKHAV = EKHAV/NXX
EKWAV = EKWAV/NXX

```

```

ETOTT = EKAV + EPAV

```

```

WRITE(8,1000) EKAV,EKHAV,EKWAV,EPAV,ETOTT,
               UV_NORM,W_NORM,Z_NORM,ZAV,
               UVRMS,WRMS,ZRMS
FORMAT(// ' EKAV = ',G0.5/

```

```

1      ' EKHAV= ',G20.5/
1      ' EKWAV= ',G20.5/
1      ' EPAV = ',G20.5/
1      ' ETOT = ',G20.5///
2      ' UV NORM = ',G20.5/
3      ' W NORM = ',G20.5/
3      ' Z NORM = ',G20.5///
4      ' ZAV = ',G20.5/
4      ' UVRMS = ',G20.5/
5      ' WRMS = ',G20.5/
6      ' ZRMS = ',G20.5//)

```

```

EMAXZ = 0
ETOTMAX = 0
EKH_ZINT = 0
EP_ZINT = 0

```

```

      EK_HOR_Z(1) = EK_HOR_Z(1)/NXX
      E_POT_Z(1) = E_POT_Z(1)/NXX
      EK_W_Z(1) = EK_W_Z(1)/NXX

```

```

DO IZ = 2,NZZ

```

```

      EK_HOR_Z(IZ) = EK_HOR_Z(IZ)/NXX
      E_POT_Z(IZ) = E_POT_Z(IZ)/NXX
      EK_W_Z(IZ) = EK_W_Z(IZ)/NXX

```

```

      IF( E_POT_Z(IZ) .GT. EMAXZ) EMAXZ = E_POT_Z(IZ)
      IF( EK_HOR_Z(IZ) .GT. EMAXZ) EMAXZ = EK_HOR_Z(IZ)
      IF( EK_W_Z(IZ) .GT. EMAXZ) EMAXZ = EK_W_Z(IZ)
      Z(IZ) = (IZ-1)*DZZ
      IF( EK_HOR_Z(IZ) .NE. 0) THEN
        EWEK_RATIO(IZ) = EK_W_Z(IZ)/EK_HOR_Z(IZ)
        EPEK_RATIO(IZ) = E_POT_Z(IZ)/EK_HOR_Z(IZ)
      ELSE
        EWEK_RATIO(IZ) = 0
        EPEK_RATIO(IZ) = 0
      END IF
      ETOT(IZ) = E_POT_Z(IZ)+EK_HOR_Z(IZ)+EK_W_Z(IZ)
      IF( ETOTMAX .LT. ETOT(IZ)) ETOTMAX = ETOT(IZ)
      EKH_ZINT = EKH_ZINT + EK_HOR_Z(IZ)*DZZ
      EP_ZINT = EP_ZINT + E_POT_Z(IZ)*DZZ

```

```

END DO

```

```

WRITE(*,2000) EKH_ZINT,EP_ZINT

```

```

2000 1  FORMAT( ' Vertically Integrated HKE = ',g16.4/
          ' Vertically integrated PE = ',g16.4//)

```

```

DO IZ = 2,NZZ

```

```

      ETOT(IZ) = 0.8* EMAXZ* ETOT(IZ)/ETOTMAX

```

```

END DO

```

```

ZMAX = DZZ*(NZZ-1)

```

```

C *****
C  DISPLA SUBPROGRAMS      *
C *****

```

```

CALL COMPRS
CALL SETCLR(%REF('GREEN'))

```

```

CALL PAGE(11.0,8.5)
CALL AREA2D(8.0,4.0)
CALL TRIPLX
CALL HEIGHT(.20)
CALL HEADIN(%REF('Internal Wave Energy$'),100,1.5,2)
CALL HEADIN(%REF('Horizontal Distribution$'),100,1.2,2)
CALL XNAME(%REF('Horizontal Distance - km$ '),100)

```

```

CALL YNAME(%REF('Energy Level - J/m**2$ '),100)
CALL GRAF(0.0,%REF('SCALE'),XMAX,0.0,%REF('SCALE'),EMAX)
CALL HEIGHT(0.1)
CALL LINES(%REF('Potential Energy$'),IPAK,1)
CALL LINES(%REF('Horizontal Kinetic Energy$'),IPAK,2)
CALL LINES(%REF('Vertical Kinetic Energy$'),IPAK,3)
CALL LEGLIN
CALL SCLPIC(0.5)
CALL SETCLR(%REF('RED'))
CALL CURVE( X,E_POT,NXX,20)
CALL SETCLR(%REF('YELLOW'))
CALL DASH
CALL CURVE( X,EK_HOR,NXX,20)
CALL SETCLR(%REF('GREEN'))
CALL DOT
CALL CURVE( X,EK_W,NXX,20)
CALL LEGEND(IPAK,3,6.0,3.0)
CALL ENDPL(0)

CALL RESET(%REF('ALL'))

CALL SETCLR(%REF('GREEN'))

CALL PAGE(8.5,11.0)
CALL AREA2D(4.0,8.0)
CALL TRIPLX
CALL HEIGHT(.20)
CALL HEADIN(%REF('Internal Wave Energy$'),100,1.5,2)
CALL HEADIN(%REF('Vertical Distribution$'),100,1.2,2)
CALL YNAME(%REF('Depth - m$ '),100)
CALL XNAME(%REF('Energy Level - J/m**2$ '),100)
CALL GRAF(0.0,%REF('SCALE'),EMAXZ,Z(NZTOT),%REF('SCALE'),0.0)
CALL HEIGHT(0.1)
CALL LINES(%REF('Potential Energy$'),IPAK,1)
CALL LINES(%REF('Horizontal Kinetic Energy$'),IPAK,2)
CALL LINES(%REF('Vertical Kinetic Energy$'),IPAK,3)
CALL LINES(%REF('Total Energy$'),IPAK,4)
CALL LEGLIN
CALL SCLPIC(0.5)
CALL SETCLR(%REF('RED'))
CALL CURVE( E_POT_Z(2),Z(2),NZTOT,20)
CALL SETCLR(%REF('YELLOW'))
CALL DASH
CALL CURVE( EK_HOR_Z(2),Z(2),NZTOT,20)
CALL SETCLR(%REF('GREEN'))
CALL DOT
CALL CURVE( EK_W_Z(2),Z(2),NZTOT,20)
CALL CHNDOT
CALL CURVE( ETOT(2),Z(2),NZTOT,20)
CALL LEGEND(IPAK,4,2.0,1.5)

CALL ENDPL(0)

CALL RESET(%REF('ALL'))

CALL SETCLR(%REF('GREEN'))

CALL PAGE(8.5,11.0)
CALL AREA2D(4.0,8.0)
CALL TRIPLX
CALL HEIGHT(.20)
CALL HEADIN(%REF('ENERGY RATIOS $'),100,1.5,1)
CALL YNAME(%REF('Depth - m$ '),100)
CALL XNAME(%REF('Energy Ratio$ '),100)
CALL GRAF(0.0,%REF('SCALE'),1.0,Z(NZTOT),%REF('SCALE'),0.0)
CALL HEIGHT(0.1)

```

```
CALL LINES(%REF('WKE/HKE Ratio$'),IPAK,1)
CALL LINES(%REF('Pot./HKE Ratio$'),IPAK,2)
CALL LEGLIN
CALL SCLPIC(0.5)
CALL SETCLR(%REF('RED'))
CALL CURVE( EWEK_RATIO(2),Z(2),NZTOT,20)
CALL SETCLR(%REF('YELLOW'))
CALL DASH
CALL CURVE( EPEK_RATIO(2),Z(2),NZTOT,20)
CALL LEGEND(IPAK,2,2.0,1.5)

CALL ENDPL(0)

CALL DONEPL
STOP
END
```

PROGRAM MODEL1_PLOT

```

C*****
C
C PROGRAM            MODEL1_PLOT
C
C PURPOSE            COMPUTE AND PLOT MODEL FIELDS
C
C AUTHOR             K.D. Saunders (NOARL, Code 331)
C
C INPUT
C-----
C Unit    Filename            Type            Contents
C-----
C    5    SYS$INPUT            KEYBOARD        Control Information
C  12    MODEL1.DAT            « direct »      Data to plot
C  12    MODEL1.UV            « direct »      UV data to plot
C  13    MODEL1.AUX            « ascii »      Descriptor for MODEL1.DAT
C-----
C
C OUTPUT
C-----
C Unit    Filename            Type            Contents
C-----
C    6    SYS$OUTPUT           «cntl window»   Program/control information
C  NA    SYS$WORKSTATION      «plot window»   Color plots
C-----
C
C Notes:
C                    This program is designed to be used on the DEC GKS
C                    Workstation 2000. It will not work on any other system
C
C*****

```

IMPLICIT NONE

```

INCLUDE            'SYS$LIBRARY:UISENTRY'
INCLUDE            'SYS$LIBRARY:UISUSRDEF'

```

```

INTEGER*4          MAX,IT
PARAMETER          (MAX=500)

```

```

LOGICAL*1          DRAW,INSIDE,NEW,INSIDELAST /.FALSE./

```

```

CHARACTER*2        PLOT_TYPE

```

```

CHARACTER*8        PBOT,PTOP

```

```

CHARACTER*80       MY_FONT1,LABEL,NUMBER

```

```

1 CHARACTER*80      XTITLE  /'X$'/,
2                   YTITLE  /'Y$'/,
3                   ZTITLE,
4                   LINE,
                    CBUFF(20)

```

```

1 REAL*4            X,Y,X_SIDE,Y_SIDE,
1                   XMAX,YMAX,
                    X0,Y0,DX,DY,

```

```

1          XFACT,YFACT,ZP,ZM,WWW,X_OFF,Y_OFF

      REAL*4          J,K,R,RED,GREEN,BLUE,DEG,SX,SY,
1          XLAST,XB,XE,YLAST,YB,YE,Z

      REAL*4          T(MAX), S(MAX), ZZ(MAX), BV(MAX),
1          ZD(MAX,MAX),DT,DXX,DZZ,WM,WP,WT,HM,HP,HT,
2          T_AV(MAX),S_AV(MAX),TD(MAX,MAX),SD(MAX,MAX),
3          SVEL,U(MAX,MAX),V(MAX,MAX),W(MAX,MAX),
4          UT(MAX),VT(MAX),WTT(MAX)

      REAL*4          DZ,FACTOR,OFFSET,SXB,SYB,SYE,Q,XT,ZMIN,ZMAX

      REAL*4          TMPX,TMPY,PIX_SIZE

      INTEGER*4        NXX,NZZ,NT,IX,IZ,NVARS,LOC

      INTEGER*4        I_INDEX(10,10,10),NPTX,NPTY,
1          VCM_SIZE,VD_ID,WD_ID,WD_ID2,N,I,VCM_ID,
2          ILAST,NITER,IXB,IXE,IYB,IYE,IM

      DATA J/17.0/
      DATA K/16/
      DATA VCM_SIZE/130/

      EQUIVALENCE      (ZD(1,1),U(1,1)),(TD(1,1),V(1,1)),(SD(1,1),W(1,1))
      EQUIVALENCE      (UT(1),T(1)),(VT(1),S(1)),(WTT(1),ZZ(1))

      MY_FONT1='DVWSVT0G03CK00GG0001QZZZZ02A000'

```

```

C*****
C      GET INPUT DATA GENERATED BY MODEL1      *
C*****

```

```

1      FORMAT(A)

      OPEN (FILE='MODEL1.AUX',UNIT=13,
1          STATUS='OLD',DISP='KEEP')

```

```

      READ(13,1) LINE
      READ(13,1) LINE
      READ(13,1) LINE
      CALL PARSE(LINE,CBUFF,NVARS)
      READ(CBUFF(2),*) NT

```

```

      READ(13,1) LINE
      CALL PARSE(LINE,CBUFF,NVARS)
      READ(CBUFF(2),*) NXX

```

```

      READ(13,1) LINE
      CALL PARSE(LINE,CBUFF,NVARS)
      READ(CBUFF(2),*) NZZ

```

```

      READ(13,1) LINE
      CALL PARSE(LINE,CBUFF,NVARS)
      READ(CBUFF(2),*) DT

```

```

      READ(13,1) LINE
      CALL PARSE(LINE,CBUFF,NVARS)

```

```
READ(CBUFF(2),*) DXX
```

```
READ(13,1) LINE  
CALL PARSE(LINE,CBUFF,NVARS)  
READ(CBUFF(2),*) DZZ
```

```
TYPE *, ' NT,NX,NZ,DT,DX,DZ '  
TYPE *, NT,NXX,NZZ  
TYPE *, DT,DXX,DZZ  
TYPE *, ' ENTER PLOT TYPE : ZD,TD,SD,SV '
```

```
ACCEPT 1, PLOT_TYPE
```

```
c degugging "goto" 2/29/89  
C go to 1001
```

```
TYPE *, ' ENTER THE TIME ITERATION NUMBER '  
ACCEPT *, IT
```

```
1 IF( PLOT_TYPE .NE. 'U' .AND. PLOT_TYPE .NE. 'V' .AND.  
PLOT_TYPE .NE. 'W') THEN
```

```
1 OPEN (FILE='MODEL1.DAT',UNIT=12,  
2 STATUS='OLD',DISP='KEEP',ACCESS='DIRECT',  
FORM='UNFORMATTED',RECL=4*NZZ)
```

```
DO IX = 1,NXX  
LOC = NXX*(IT-1)+IX  
READ(12'LOC')(ZD(IX),T(IX),S(IX),BV(IX),I=1,NZZ)  
DO IZ = 1,NZZ  
TD(IZ,IX) = T(IZ)  
SD(IZ,IX) = S(IZ)  
T_AV(IZ) = T_AV(IZ) + T(IZ)  
S_AV(IZ) = S_AV(IZ) + S(IZ)
```

```
END DO  
END DO
```

```
DO IZ = 1,NZZ  
T_AV(IZ) = T_AV(IZ)/NXX  
S_AV(IZ) = S_AV(IZ)/NXX  
END DO
```

```
DO IX = 1,NXX  
DO IZ = 1, NZZ  
IF( PLOT_TYPE .NE. 'SV') THEN  
TD(IZ,IX) = TD(IZ,IX) - T_AV(IZ)  
SD(IZ,IX) = SD(IZ,IX) - S_AV(IZ)  
END IF  
END DO  
END DO
```

```
ELSE
```

```
1 OPEN (FILE='MODEL1.UV',UNIT=12,  
2 STATUS='OLD',DISP='KEEP',ACCESS='DIRECT',  
FORM='UNFORMATTED',RECL=3*NZZ)
```

```

DO IX = 1,NXX
  LOC = NXX*(IT-1) + IX
  READ(12,LOC)(UT(I),VT(I),WTT(I),I=1,NZZ)
  DO IZ = 1,NZZ
    U(IZ,IX)=UT(IZ)
    V(IZ,IX)=VT(IZ)
    W(IZ,IX)=WTT(IZ)
  END DO
END DO

IF( PLOT_TYPE .EQ. 'V') THEN
  DO IX = 1,NXX
    DO IZ = 1,NZZ
      ZD(IZ,IX) = V(IZ,IX)
    END DO
  END DO
END IF

IF( PLOT_TYPE .EQ. 'W') THEN
  DO IX = 1,NXX
    DO IZ = 1,NZZ
      ZD(IZ,IX) = W(IZ,IX)
    END DO
  END DO
END IF

END IF

1 IF( PLOT_TYPE .EQ. 'TD' .OR. PLOT_TYPE .EQ. 'SD' .OR.
  PLOT_TYPE .EQ. 'SV' ) THEN
  DO IX = 1,NXX
    DO IZ = 1,NZZ
      IF( PLOT_TYPE .EQ. 'TD')
        ZD(IZ,IX) = TD(IZ,IX)
      IF( PLOT_TYPE .EQ. 'SD')
        ZD(IZ,IX) = SD(IZ,IX)
      IF( PLOT_TYPE .EQ. 'SV' ) THEN
        ZD(IZ,IX) = SVEL(SD(IZ,IX),TD(IZ,IX),
          (IZ-1)*DZZ )
        IF(IX .EQ. 1) S_AV(IZ) = 0
        S_AV(IZ) = S_AV(IZ) + ZD(IZ,IX)
      END IF
    END DO
  END DO

  IF( PLOT_TYPE .EQ. 'SV' ) THEN
    DO IZ = 1,NZZ
      S_AV(IZ) = S_AV(IZ)/NXX
    END DO

    DO IX = 1,NXX
      DO IZ = 1,NZZ
        ZD(IZ,IX) = ZD(IZ,IX) - S_AV(IZ)
      END DO
    END DO
  END IF

```

END IF

c continue here for debugging goto 2/29/89
1001 continue

C*****
C GET INPUT DATA FOR SCREEN CONTROL *
C*****

C TYPE *, ' ENTER SCREEN SIZE IN CM '
C ACCEPT *,SX
C TYPE *, ' ENTER NO POINTS ON SCREEN IN X AND Y DIRECTIONS '
C ACCEPT *, NPTX,NPTY

SX = 34.0
SY = 28.5
NPTX= 1000
NPTY= 1000

X_SIDE = NXX*DXX
Y_SIDE = NZZ*DZZ

PIX_SIZE = 29.38
XFACT = PIX_SIZE*(SX/X_SIDE)*0.9
YFACT = PIX_SIZE*(SY/Y_SIDE)*0.9
X_OFF = PIX_SIZE*(0.05)*SX
Y_OFF = PIX_SIZE*(0.05)*SY

DX = X_SIDE/NPTX
DY = Y_SIDE/NPTY

X0 = 0
Y0 = 0

XMAX = X_SIDE - DX
YMAX = Y_SIDE - DY

C*****
C SET UP PLOTTING FEATURES *
C*****

VCM_ID = UIS\$CREATE_COLOR_MAP(VCM_SIZE)
C GET BETTER SCALING
TMPX = -0.05*X_SIDE
TMPY = -0.05*Y_SIDE
C VD_ID = UIS\$CREATE_DISPLAY(0,0,XMAX,YMAX,SX,SX,VCM_ID)
1 VD_ID = UIS\$CREATE_DISPLAY(TMPX,TMPY,XMAX-TMPX,YMAX-TMPY,
SX,SY,VCM_ID)
WD_ID = UIS\$CREATE_WINDOW(VD_ID,'SYS\$WORKSTATION','WINDOW #1')

DO N = 1,124
DEG = N*360.0/124.0 + 60.0

```

        IF( DEG .GT. 360.0) DEG = DEG - 360.0
        CALL UIS$HSV_TO_RGB(DEG,1.0,1.0,RED,GREEN,BLUE)
        CALL UIS$SET_COLOR(VD_ID,N,RED,GREEN,BLUE)
    END DO

C CREATE BLACK
    CALL UIS$SET_COLOR(VD_ID,101,0.0,0.0,0.0)

C CREATE WHITE
    CALL UIS$SET_COLOR(VD_ID,102,1.0,1.0,1.0)

    DO I = 1,124
        CALL UIS$SET_WRITING_INDEX(VD_ID,0,I,I)
    END DO

C "GO TO 1000" FOR DEBUGGING PURPOSES 2/28/89 SAB
C     GO TO 1000

C*****
C DRAW A BOX AROUND THE AREA      *
C*****
C     CALL UIS$SET_LINE_WIDTH(VD_ID,102,102,4.0,UIS$C_WIDTH_PIXELS)
C     CALL UIS$PLOT(VD_ID,102,0.0,0.0, XMAX,0.0,
C     1           XMAX,YMAX, 0.0,YMAX, 0.0,0.0)

    DO X = 0,XMAX,DX
        IX = X/DXX + 1
        IF(IX .LT. 1) IX = 1
        IF(IX .GT. NXX) IX = NXX

        WM = ABS((IX-1)*DXX - X)/DXX
        WP = ABS(IX*DXX - X)/DXX
        WT = WM+WP
        WM = 1.0 - WM/WT
        WP = 1.0 - WP/WT

        XLAST = X
        YLAST = Y0
        XB = X
        XE = X
        YB = Y0
        YE = Y0
        NEW = .TRUE.
        DRAW = .FALSE.

    DO Y = 0,YMAX,DY

        IZ = (YMAX-Y)/DZZ + 1
        IF(IZ .LT. 1) IZ = 1
        IF(IZ .GT. NZZ) IZ = NZZ
        HM = ABS((IZ-1)*DZZ - (YMAX-Y))/DZZ
        HP = ABS( IZ*DZZ      - (YMAX-Y))/DZZ
        HT = HM + HP
        HM = 1.0 - HM/HT
        HP = 1.0 - HP/HT

C     IF( IX .EQ. 1 .AND. IZ.LT. 10) THEN
C         TYPE *, 'YMAX-Y,IZ,HM,HP',YMAX-Y,IZ,HM,HP
C     END IF

```

```

IM = IZ-1
IF(IM.LE.1)IM = 1

IF(IX .GT. 1) THEN
    ZP = (WP*ZD(IZ,IX) + WM*ZD(IZ,IX-1))/(WM+WP)
    ZM = (WP*ZD(IM,IX) + WM*ZD(IM,IX-1))/(WM+WP)
ELSE
    ZP = ZD(IZ,IX)
    ZM = ZD(IM,IX)
END IF

```

```

Z = (HP*ZP + HM*ZM)/(HM+HP)

```

C DEFAULT COLOR SCALE UNIT LABELS

```

PBOT = '-0.1'
PTOP = '0.1'

IF (PLOT TYPE .EQ. 'ZD') THEN
    I = 5.0*Z
    LABEL = 'VERTICAL DISPLACEMENT (m)'
    PBOT = '-10.0'
    PTOP = '10.0'
END IF
IF( PLOT TYPE .EQ. 'TD') THEN
    I = 50.*Z
    LABEL = 'TEMPERATURE ANOMALY (°C)'
    PBOT = '-1.0'
    PTOP = '1.0'
END IF
IF( PLOT TYPE .EQ. 'SD') THEN
    I = 500.*Z
    LABEL = 'SALINITY ANOMALY (psu)'
END IF
IF( PLOT TYPE .EQ. 'SV') THEN
    I = 50.0*Z
    LABEL = 'SOUND VELOCITY ANOMALY (m/s)'
END IF
IF( PLOT TYPE .EQ. 'U') THEN
    I = 500.*Z
    LABEL = 'HORIZONTAL -U- VELOCITY (m/s)'
END IF
IF( PLOT TYPE .EQ. 'V' ) THEN
    I = 500.*Z
    LABEL = 'HORIZONTAL -V- VELOCITY (m/s)'
END IF
IF( PLOT TYPE .EQ. 'W' ) THEN
    I = 5000.*Z
    LABEL = 'VERTICAL VELOCITY (m/s)'
END IF

```

```

IF(I.LE.0) I = 100 + I
I = ABS(I)
I = I - (I/100)*100
IF(I .LT. 1 ) I = 1
IF(I .GT. 100 ) I = 100

```

```

C
I = 2*(I/2)

```

```

IF( NEW ) THEN
    NEW = .FALSE.
    ILAST = I
    XLAST = X
    YLAST = Y

```

```

        END IF

        IF( I .NE. ILAST) THEN
            XE = XLAST
            YE = YLAST
            DRAW = .TRUE.
        END IF

        IF( Y .GT. YMAX-DY) THEN
            XE= X
            YE =Y
            DRAW = .TRUE.
        END IF

        IF( DRAW )THEN

C          CALL UIS$PLOT(VD_ID,ILAST,XB,YB,XE,YE)


            IXB = XFACT*XB + X_OFF
            IXE = XFACT*XE + X_OFF
            IYB = YFACT*YB + Y_OFF
            IYE = YFACT*YE + Y_OFF

            CALL UISDC$PLOT(WD_ID,ILAST,IXB,
1          IYB,IXE,IYE)

            XB = X
            YB = Y
            DRAW = .FALSE.
        END IF

        XLAST = X
        YLAST = Y
        ILAST = I

        END DO
        END DO
        ! Y
        ! X

C CONTINUE HERE FOR DEBUGGING 2/28/89
1000 CONTINUE

C CREATE BACKGROUND FOR COLOR SCALE
    DZ = .05
    FACTOR = XMAX/250.0
    OFFSET = (4.5/6.0)*XMAX
    SYB = TMPY
    SYE = 300.0

    DO XT = -50,50,DZ
        X = FACTOR*XT + OFFSET
        I = XT
        IF (I .LT. 0) I = I + 100
        CALL UIS$PLOT(VD_ID,102,X,SYB,X,SYE)
    END DO

C CREATE COLOR SCALE
    OFFSET = (4.5/6.0)*XMAX
    FACTOR = XMAX/300.0
    SYB = 0.0
    SYE = 100.0

    DO XT = -50,50,DZ
        X = FACTOR*XT + OFFSET
        I = XT
        IF (I .LT. 0) I = I + 100

```

```
      CALL UIS$PLOT(VD_ID,I,X,SYB,X,SYE)
END DO
```

C SETUP FONTS FOR WRITING

```
      CALL UIS$SET_FONT(VD_ID,0,1,MY_FONT1)
      CALL UIS$NEW_TEXT_LINE(VD_ID,4)
```

C FOR DEBUGGING

```
C      LABEL = 'TEMPERATURE TEST'
C      PBOT = '-10.0'
C      PTOP = '10.0'
```

C LABEL PLOT TITLE

```
      X = -50*FACTOR + OFFSET
      call uis$set_char_size(VD_ID,101,103,,,75.0)
      CALL UIS$TEXT(VD_ID,103,LABEL,X,SYE+200)
```

C LABEL COLOR SCALE UNITS

```
      call uis$set_char_size(VD_ID,101,103,,,70.0)
      X = -50*FACTOR + OFFSET - .9
      CALL UIS$TEXT(VD_ID,103,PBOT,X,SYE+100)
```

```
      X = OFFSET - .5
      CALL UIS$TEXT(VD_ID,103,'0',X,SYE+100)
```

```
      X = 50*FACTOR + OFFSET - .9
      CALL UIS$TEXT(VD_ID,103,PTOP,X,SYE+100)
```

C LABEL HORIZONTAL AXIS

```
      LABEL = 'HORIZONTAL DISTANCE (km)'
      X = XMAX/3.0
      call uis$set_char_size(VD_ID,102,103,,,75.0)
      CALL UIS$TEXT(VD_ID,103,LABEL,X,YMAX + 150.0)
```

```
      DO I = 0,40,10
          X = I
          PTOP = NUMBER(I)
          call uis$set_char_size(VD_ID,103,103,,,70.0)
          CALL UIS$TEXT(VD_ID,103,PTOP,X,YMAX + 80.0)
      END DO
```

C LABEL VERTICAL AXIS

```
      LABEL = 'DEPTH (m)'
      X = YMAX/2.5
```

```
      call uis$set_char_size(VD_ID,102,103,,,75.0)
      call uis$set_text_slope(VD_ID,103,103,90.0)
      CALL UIS$TEXT(VD_ID,103,LABEL,-2.0,X)
```

```
      DO I = 0,3000,1000
          X = I
          PTOP = NUMBER(3000-I)
          call uis$set_char_size(VD_ID,103,103,,,70.0)
          CALL UIS$TEXT(VD_ID,103,PTOP,-1.3,X)
      END DO
```

```
PAUSE
STOP
```

END

```
      REAL FUNCTION SVEL(S,T,P0)
C *****
C SOUND SPEED SEAWATER CHEN AND MILLERO 1977,JASA,62,1129-1135
C UNITS:
C     PRESSURE      P0      DECIBARS
C     TEMPERATURE   T       DEG CELSIUS (IPSS-68)
C     SALINITY      S       (IPSS-78)
C     SOUND SPEED   SVEL    METERS/SECOND
C CHECKVALUE: SVEL=1731.995 M/S, S=40 (IPSS-78),T=40 DEG C,P=10000 DBAR
C
      EQUIVALENCE (A0,B0,C0),(A1,B1,C1),(A2,C2),(A3,C3)
C
C  SCALE PRESSURE TO BARS
      P=P0/10.
C*****
      SR = SQRT(ABS(S)).
C S**2 TERM
      D = 1.727E-3 - 7.9836E-6*P
C S**3/2 TERM
      B1 = 7.3637E-5 +1.7945E-7*T
      B0 = -1.922E-2 -4.42E-5*T
      B = B0 + B1*P
C S**1 TERM
      A3 = (-3.389E-13*T+6.649E-12)*T+1.100E-10
      A2 = ((7.988E-12*T-1.6002E-10)*T+9.1041E-9)*T-3.9064E-7
      A1 = (((-2.0122E-10*T+1.0507E-8)*T-6.4885E-8)*T-1.2580E-5)*T
X      +9.4742E-5
      A0 = (((-3.21E-8*T+2.006E-6)*T+7.164E-5)*T-1.262E-2)*T
X      +1.389
      A = ((A3*P+A2)*P+A1)*P+A0
C S**0 TERM
      C3 = (-2.3643E-12*T+3.8504E-10)*T-9.7729E-9
      C2 = (((1.0405E-12*T-2.5335E-10)*T+2.5974E-8)*T-1.7107E-6)*T
X      +3.1260E-5
      C1 = (((-6.1185E-10*T+1.3621E-7)*T-8.1788E-6)*T+6.8982E-4)*T
X      +0.153563
      C0 = (((3.1464E-9*T-1.47800E-6)*T+3.3420E-4)*T-5.80852E-2)*T
X      +5.03711)*T+1402.388
      C = ((C3*P+C2)*P+C1)*P+C0
C SOUND SPEED RETURN
      SVEL = C + (A+B*SR+D*S)*S
      RETURN
      END
```

```

C*****
C
C PROGRAM          DISPERSION
C
C PURPOSE          Produces modal dispersion diagrams for a Brunt-
C                  Väisälä frequency profile at a given location and
C                  season.
C
C AUTHOR(S)        K.D. Saunders (NOARL)
C*****
C*****
C INPUT
C-----
C UNIT   FILE          DATA
C-----
C   5    SYS$INPUT      « Ephemeral input file - keyboard »
C
C                  1. Starting Latitude      (decimal °)
C                  2. Starting Longitude     (decimal °)
C                  3. Direction of section ( ° from north)
C                  4. Max Range (xmax)       ( km )
C                  5. Max Depth (zmax)       ( m )
C                  6. Delta x                ( km )
C                  7. Delta z                ( m )
C                  8. Max time               ( s )
C                  9. Delta time             ( s )
C
C
C  10    LEVITUS.DAT    « DIRECT ACCESS»
C
C                  Base Temperature and Salinity Profiles needed to
C                  define the field of Brunt-Väisälä frequencies.
C                  along the section.
C*****
C*****

```

```

C*****
C*****
C
C OUTPUT
C-----
C Unit   FILE           DATA
C-----
C  6     SYS$OUTPUT      « ephemeral file »
C
C                        1. Diagnostic information
C
C
C  11     TEST_DIAGNOSTICS.LIS « ASCII file »
C
C                        1. Diagnostic information
C
C  ??     POPFIL.DAT      1. DISSPLA meta file
C
C*****
C*****

```

```
C*****
C*****
C
C
C NOTES
C 1. The following assumptions are made for this first level
C model:
C
C   ▣ no mean currents are assumed.
C     (this restriction will be relaxed in later versions)
C
C   ▣ only the internal wave part of the spectrum affects the
C     fields of temperature and salinity.
C     (this restriction will be relaxed in later versions)
C
C   ▣ the temperature and salinity fields are initially defined
C     based on the Levitus 5° data base averages.
C     (this restriction will be relaxed in later versions)
C
C   ▣ if the BV frequency is imaginary, it is set to zero in the
C     mode calculations.
C
C   ▣ the internal wave field does not affect the modes for  $t > 0$ 
C     (this restriction will be relaxed in later versions)
C
C 2. The profile input section is derived from programs written
C     by William Teague, NOARL, Code 331 in conjunction with the
C     MOODS data base project.
C
C 3. The internal wave simulation section is derived from programs*
C     written by Dr. David Rubenstein , SAIC
C
C*****
C*****
```

IMPLICIT NONE

CHARACTER*8 TIMEBUFF

REAL*4 TTT0,DTTT

INCLUDE 'MODEL1.INC'

TTT0 = SECNDS(0.0)

CALL CONTROL_INPUT
CALL PROFILE_INPUT
CALL INT_WAVE_SIMULATION

CALL TIME(TIMEBUFF)
DTTT = SECNDS(TTT0)
WRITE(11,100) TIMEBUFF,DTTT/60.0

100 1 FORMAT('///// ' ENDING TIME = ',A8 /
 ' ELASPED TIME(MIN) = ',G20.5)

STOP
END

SUBROUTINE CONTROL_INPUT

```

C*****
C*****
C
C PROGRAM          CONTROL_INPUT
C
C PURPOSE          Reads in control data
C                  IO terminal files opened
C                  Auxilliary Latitude and Longitude computed
C
C HISTORY 10/25/88      1. Coding begun.
C
C AUTHOR(S)      K.D. Saunders (NOARL)
C
C*****
C*****
C
C INPUT
C-----
C      FILE          DATA
C-----
C      SYS$INPUT      1. Starting Latitude      (decimal °)
C                    2. Starting Longitude      (decimal °)
C                    3. Direction of section ( ° from north)
C                    4. Max Range (xmax)         ( km )
C                    5. Max Depth (zmax)         ( m )
C                    6. Delta x                   ( km )
C                    7. Delta z                   ( m )
C                    8. Max time                  ( s )
C                    9. Delta time                ( s )
C
C*****
C*****
C
C OUTPUT
C-----
C      FILE          DATA
C-----
C      SYS$OUTPUT      1. Diagnostic information
C
C      COMMONS        1. All output is passed through named common
C*****
C*****

```

IMPLICIT NONE

CHARACTER*8 TIMEBUFF

INTEGER*4 IANG

REAL*4 SINE, COSE

INCLUDE 'MODEL1.INC'

```

OPEN ( FILE=TERMINAL_INPUT,UNIT=5,STATUS='UNKNOWN',DISP='DELETE')
OPEN ( FILE=TERMINAL_OUT ,UNIT=6,STATUS='UNKNOWN',DISP='DELETE')
OPEN ( FILE='TEST_DIAGNOSTICS.LIS',UNIT=11,STATUS='NEW',DISP='KEEP')

```

```

CALL TIME(TIMEBUFF)
WRITE(11,90) TIMEBUFF
FORMAT(/// ' STARTING TIME = ',A8//)

```

C
C
C
C

* Read in control *
* data *

```
WRITE(6,100)
READ(5,*)      LAT,LON
WRITE(6,105)
READ(5,1)      SEASON
WRITE(6,110)
READ(5,*)      ZMAX,DZ
```

```
NX = 1
```

```
NZ = ZMAX/DZ + 1
```

```
IF( NZ .GT. MAX) THEN
```

```
    NZ = MAX
```

```
    DZ = ZMAX/(NZ-1)
```

```
END IF
```

```
RETURN
```

```
1      FORMAT(A)
```

```
100    FORMAT( //' *****OCEAN SIMULATION MODEL*****'//
              '                      VERSION 1.0                      '///
              ' Enter latitude, longitude '/')
```

```
105    FORMAT(/// ' Enter season (WINTER,SPRING,SUMMER OR FALL)')///)
```

```
110    FORMAT(/// ' Enter maximum and delta depths in m, '///)
```

```
END
```

SUBROUTINE PROFILE_INPUT

```

C*****
C*****
C
C PROGRAM          PROFILE_INPUT
C
C PURPOSE          LOCATES PROFILES AT LAT,LON,LAT1,LON1 AND READS IN THE
C                  TEMPERATURE AND SALINITY PROFILES AT BOTH LOCATIONS FROM*
C                  LEVITUS 5° DATABASE.
C
C HISTORY          10/25/88          1. Program begun.
C
C AUTHOR(S)        K.D. Saunders (NOARL)
C
C*****
C*****
C
C INPUT
C
C      All input is via named common
C
C*****
C*****
C
C OUTPUT
C      SYS$OUTPUT          Diagnostic information
C
C      COMMONS             All data are returned via named common
C
C*****
C*****
C
C NOTES
C
C      The following notes are from the comments in Wm. Teague's program
C-----
C      PROGRAM:  LEVFEB
C      PURPOSE:  THIS PROGRAM READS A DIRECT ACCESS FILE CREATED BY LEVRD AND
C                WRITES AND WRITES THE DATA IN VFEB FORMAT.  THE OUTPUT GROUP
C                CONSISTS OF 30 DEPTH LEVELS WITH DEPENDENT VARIABLES OF
C                NO. OF TEMP OBSERVATIONS, MEAN TEMP, STANDARD DEVIATION OF
C                TEMP, NO. OF SAL OBSERVATIONS, MEAN SAL, AND STANDARD DEVIATION
C                OF SAL.
C-----
C*****

```

IMPLICIT NONE

INCLUDE 'MODEL1.INC'

```

      INTEGER*4  ISHIF,
1              IPOSLOOP,
2              ISF,
3              IREC

```

```

      REAL*4     D(180),
1              ZLEV(30),
2              T_TEMP(300),
3              S_TEMP(300),
4              P(2),
5              PAV,

```

```

6      X_RATIO,
7      D_PROFILES,
8      E,
9      RLAT,
A      RLON,
B      DIST,
C      BVFRQ

```

```

1      DATA ZLEV/0,10,20,30,50,75,100,125,150,200,250,300,400,500,
2      600,700,800,900,1000,1100,1200,1300,1400,1500,
      1750,2000,2500,3000,3500,4000/

```

```

C*****
C  OPEN INPUT FILE  *
C*****

```

```

&      OPEN(UNIT=10,FILE='MODELBASE$:LEVITUS.DAT',
&      ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='OLD',
&      ERR=9091,RECL=180,READONLY)

```

```

C      *****
C      *   WINTER = FEB, MAR, APR   -   *
C      *   USE MID MARCH FOR TIME IN FDOC(1,1)   *
C      *****

```

```

      IF (SEASON(1:2).EQ.'WI') THEN
        ISHIF=0

```

```

C      *****
C      *   SPRING = MAY, JUN, JUL   *
C      *****

```

```

      ELSE IF (SEASON(1:2).EQ.'SP') THEN
        ISHIF=36

```

```

C      *****
C      *   SUMMER = AUG, SEP, OCT   *
C      *****

```

```

      ELSE IF (SEASON(1:2).EQ.'SU') THEN
        ISHIF=72

```

```

C      *****
C      *   FALL = NOV, DEC, JAN   *
C      *****

```

```

      ELSE IF (SEASON(1:2).EQ.'FA') THEN
        ISHIF=108

```

```

      ELSE

```

```

C      *****
C      *   USE SUMMER IF SEASON NOT CORRECTLY   *
C      *   SPECIFIED   *
C      *****

```

```

        ISHIF = 72
      END IF

```

```

      IPOSLOOP = 1

```

```

      RLAT = LAT
      RLON = LON

```

```

      IF(RLON.LT.0)RLON=RLON+360.
      RLAT=RLAT+90.

```

```

C      *****
C      *   CHECK LAT LON VALUESW   *
C      *****

```

```

IF (ABS(RLON).GE.360.) THEN
  WRITE(6,*)'LONGITUDE NOT BETWEEN -180 AND 180 ',RLON
  STOP
END IF

IF (ABS(RLAT).GT.180) THEN
  WRITE(6,*)'LATITUDE NOT BETWEEN -90 AND 90 ',RLAT
  STOP
ENDIF

```

C
C
C

```

*****
* COMPUTE DIRECT ACCESS RECORD NO.S *
*****

```

```

I=RLON/5.+1.
J=RLAT/5.+1.
IREC=(I-1)*144+J+ISHIF

```

C
C
C
C

```

*****
* READ DATA RECORD - NUMOBS, TEMP, *
* SIGMA, NUMOBS, SAL, SIGMA *
*****

```

```

READ(10'IREC,ERR=9092)D

```

```

K=0
ISF=0

```

```

WRITE(11,130)

```

```

DO 50 L=1,90,3

```

```

  K=K+1
  BUF(1)=ZLEV(K)
  BUF(2)=D(L)
  BUF(3)=D(L+1)
  BUF(4)=D(L+2)
  BUF(5)=D(L+90)
  BUF(6)=D(L+91)
  BUF(7)=D(L+92)

```

C
C
C
C
C

```

*****
* CHECK FOR 0 OBSERVATIONS *
* INSERT MISSING RECORD FLAG*
* THEN -999.0 *
*****

```

```

  IF (BUF(2).LE.0.1) THEN
    BUF(3)=-999.0
    BUF(4)=-999.0
  END IF

```

```

  IF (BUF(5).LE.0.1) THEN
    BUF(6)=-999.0
    BUF(7)=-999.0
  END IF

```

```

  Z_IN(IPOSLOOP,K) = BUF(1)
  TEMP_IN(IPOSLOOP,K) = BUF(3)
  SAL_IN(IPOSLOOP,K) = BUF(6)

```

```

  IF( K.GT.1 .AND. TEMP_IN(IPOSLOOP,K) .LE.-998.0) THEN
    TEMP_IN(IPOSLOOP,K) = TEMP_IN(IPOSLOOP,K-1)
  END IF

```

```

  IF( K.GT.1 .AND. SAL_IN(IPOSLOOP,K) .LE. -998.0) THEN
    SAL_IN(IPOSLOOP,K) = SAL_IN(IPOSLOOP,K-1)
  END IF

```

```

1      WRITE(11,140) IPOSLOOP,K,TEMP_IN(IPOSLOOP,K),
      SAL_IN(IPOSLOOP,K)

50     CONTINUE

C
C
C      *****
      * CLOSE THE LEVITUS FILE *
      *****

      CLOSE(UNIT=10)

C*****
C
C      INTERPOLATE TEMPERATURE AND SALINITY PROFILES FROM THE INPUT
C      PROFILES ONTO THE SECTION
C
C      1. First, compute the distance between the profiles and use as
C      input distance.
C      2. Second, fill T,S to desired depth if required
C      3. Interpolate to the z-grid
C      4. Interpolate to the x-grid
C      5. Compute Brunt-Väisälä frequencies
C
C*****
C
      DO I = 1,NZ
          Z_OUT(I) = (I-1)*DZ
          ZBV(I)   = Z_OUT(I)
      END DO

C
C
C      *****
      * Set up starting
      * temperature
      * and salinity
      * profiles
      *****

      DO I = 1,30
          T_TEMP(I) = TEMP_IN(1,I)
          S_TEMP(I) = SAL_IN(1,I)
      END DO

      CALL INTRPL(6,30,ZLEV,T_TEMP,NZ,Z_OUT,DUMMY)
      DO I = 1,NZ
          TEMP(I,1) = DUMMY(I)
      END DO

      CALL INTRPL(6,30,ZLEV,S_TEMP,NZ,Z_OUT,DUMMY)
      DO I = 1,NZ
          SAL(I,1) = DUMMY(I)
      END DO

C
C
C      *****
      * Compute BV Freqs
      *****

      K = 1
      DO I = 1,NZ-1
          T_TEMP(1) = TEMP(I,K)
          T_TEMP(2) = TEMP(I+1,K)
          S_TEMP(1) = SAL(I,K)
          S_TEMP(2) = SAL(I+1,K)
          P(1)      = Z_OUT(I)
          P(2)      = Z_OUT(I+1)
          BVF(I,K)  = BVFRQ(S_TEMP,T_TEMP,P,2,PAV,E)
      END DO
      BVF(NZ,K) = BVF(NZ-1,K)

```

```

999      RETURN
9091     STOP 'ERROR IN OPENING LEVITUS FILE'
9092     STOP 'ERROR IN READING LEVITUS FILE'

. 100    FORMAT( ' PROFILE ',I4,'      X = ',F10.3//
      1      '      Z      T      S      BV '//)

. 110    FORMAT( 4F12.3)
120     FORMAT( '***** INITIAL INTERPOLATE PROFILES TEMP,SAL,BVF',
      1      ' ARRAYS *****'//)
130     FORMAT('// INPUT TEMPERATURE AND SALINITY PROFILES '//)
140     FORMAT( 1X,2I4,2F12.3 )

      END

```

SUBROUTINE INT_WAVE_SIMULATION

```

C*****
C*****
C
C PROGRAM      INT_WAVE_SIMULATION
C
C PURPOSE      Does most of the calculations for MODEL1. It is based
C               on the Garrett-Munk internal wave model.
C
C HISTORY      10/26/88      Coding begun
C
C AUTHOR(S)    K.D. Saunders (NOARL)
C
C*****
C*****
C
C INPUT        All interprocess communication is via named common.
C
C*****
C*****
C
C OUTPUT       All output is done in subroutine calls.
C
C*****
C*****
C
C Notes
C   Subroutines called:
C
C               DISPLACEMENTS
C
C*****
C*****

```

IMPLICIT NONE

INCLUDE 'MODEL1.INC'

LOGICAL*1 BV_CHANGED / .TRUE. /

INTEGER*4 IDIR,NBV,NNZ,IIX

REAL*4 ZT(MAX),BVT(MAX)

NMODES = 5
 NF = 5
 NBV = NZ
 NNZ = NZ

IX = 1

DO K = 1,NBV
 BVT(K) = BVF(K,IX)
 ZT(K) = ZBV(K)
END DO

NMODES = 5
 NF = 1
 NBV = NZ
 NNZ = NZ
 NX = 1

```

T      = 0
XMAX   = 0
IDIR   = 0
TYPE *, ' LAT,AZIMUTH',LAT,AZIMUTH

```

```

*****
* This is the same call *
* as in Rubenstein's,   *
* except for the IX      *
* parameter. The         *
* displacements are      *
* computed at each range*
*****

```

C
C
C
C
C
C
C
C

```

1 CALL DISPLACE(ZD,NNZ,NX,XMAX,T,
2               AZIMUTH,IDIR,NF,NMODES,
3               LAT,NBV,ZT,BVT,IX,
               MAX,MAX)

```

RETURN

```

100 FORMAT( ' IX,NX,NZ,NBV,NF,NMODES =',7I10/
1        ' XMAX = ',G15.4,' T = ',G15.4,
2        ' LAT = ',G15.4//)

```

```

110 FORMAT( ' ***RETURN FROM DISPLACE***')

```

```

120 FORMAT( ' ***** OCEAN SIMULATION MODEL VERSION 1.0 '//
1        ' NT ',I5/
2        ' NX ',I5/
3        ' NZ ',I5/
4        ' DT ',G20.5/
5        ' DX ',G20.5/
6        ' DZ ',G20.5/
7        ' T0 ',G20.5/
8        ' LAT ',G20.5/
9        ' LON ',G20.5/
A        ' AZ ',G20.5//)

```

END

```

1      SUBROUTINE DISPLACE ( Z, NZ, NX, TOTX, T, ANGLE, IDIR,
2      NF, NMODES, LAT, NBV, ZT, BVT, IX,
      NBVMAX, NXMAX)

```

```

C*****
C
C PROGRAM      DISPLACE
C
C PURPOSE:
C
C Calculate random vertical displacements (correlated in time) due to
C internal waves. A Garrett-Munk type of spectrum is used to generate the
C proper energy levels. The displacements are packed into array Z(NZ,NX),
C which covers a vertical plane.
C
C Input Parameters
C
C NZ          Number of points in the vertical used in
C              calculating modes (Integer*4)
C NX          Number of points in the horizontal (Integer*4)
C TOTX        Total distance in x-direction, in meters (Real*4)
C T           Time in seconds (Real*4)
C ANGLE       Azimuth angle of the vertical plane, in degrees
C IDIR        Flag for directionality of internal waves
C              = 0  Isotropic
C              = 1  Along-range propagation (ky = 0)
C              = 2  Cross-range propagation (kx = 0)
C NF          Number of frequencies in expansion (Integer*4)
C NMODES      Number of modes in expansion (Integer*4)
C LAT         Latitude, in degrees (Real*4)
C NBV         Number of points in BV profile, and in output array Z(Integer*4)
C ZT          Depths of BV frequencies, and of output displacements Z,
C              in meters (Real*4 array of length NBV)
C BVT         Set of BV frequencies, in cph (Real*4 array of length NBV)
C
C Output parameter
C
C Z           Array of vertical displacements, in meters (Real*4 2-D array
C              of size NBVMAX x NXMAX)
C
C Note: The BV-frequency array BVT is of length NBV, which is interpolated
C onto a regularly spaced grid of length NZ. The output array WM from
C subroutine MODESUB is interpolated back into a grid of length NBV.
C
C MAX is the maximum number of depth points allowed in MODESUB calculations
C
C*****

```

```

PARAMETER ( MAX = 5000, MODEMAX = 20, NFMAX = 500)

```

```

REAL      Z(NBVMAX,NXMAX), BVT(NBVMAX), ZT(NBVMAX), LAT
REAL      FF(2),ZF(2)
REAL      F(NFMAX), FR(NFMAX), ZDEP(MAX), WMINT(MAX)
REAL      K(0:MODEMAX), WM(MAX,0:MODEMAX),w(max)
REAL      KX, KY
REAL      COST(0:MODEMAX,NFMAX), SINT(0:MODEMAX,NFMAX)
REAL*4    EPSILON, KK(NFMAX,0:MODEMAX),XK(NFMAX),XF(NFMAX)

```

```

COMPLEX A(0:MODEMAX,NFMAX), I1, FACTOR

```

```

DATA JSTAR / 1 /

```

```

INTEGER*4  ISEED0    /191531459/,
1          ISEED,

```



```

        DO MM = 0,MODEMAX
            WM(M,MM) = 0.0
        END DO
    END DO

    CALL MODESUB ( NMODES, F(IFREQ), NBV, ZT, BVT, NZ, LAT,
1          EPSILON,
2          IPRINT , k, zdep, wm )

    WRITE(*,*) 'ifreq=', ifreq, F(IFREQ), '      K = '
    WRITE(*,*) (k(m), m=0, nmodes)
    WRITE(11,100) F(1), (K(M), M=0, 3)

    DO M = 0, NMODES
        KK(IFREQ, M) = 1000 * K(M) / (2. * pi)
    END DO

    IF( IFREQ .EQ. 1 ) THEN
        DO I = 1, NZ
            WRITE(11,110) ZDEP(I), (wm(i, m), m=0, 3)
        END DO
    END IF

C                                     *****
C                                     * END IFREQ LOOP *
C                                     *****

    END DO

C*****
C                                     *
C    PLOT OUTPUT HERE                                     *
C                                     *
C*****

    CALL COMPRS
    CALL SETDEV(8,9)

C                                     *****
C                                     * SET UP PLOT AND PLOT THE RESULTS *
C                                     *****

    CALL PAGE(11.0, 8.5)
    CALL PHYSOR(1.0, 2.0)
    CALL AREA2D(4.0, 5.0)
    CALL SERIF
    CALL SHDCHR(0.0, 1, .01, 1)
    CALL HEIGHT(.175)
    CALL HEADIN(%REF('INTERNAL WAVE MODES$'), 100, 1.25, 1)
    CALL XNAME(%REF('Horizontal Wavenumber - cpkm$'), 100)
    CALL YNAME(%REF('Frequency - cph$'), 100)
    CALL GRAF(0.0, 1.0, 4.0, 0.0, 1.0, 4.0)
    CALL FRAME
    CALL GRACE(0.0)

    DO M = 0, NMODES
        NNF = 0
        DO I = 1, NF
            IF(KK(I, M) .GT. 1.0E-4 ) THEN
                NNF = NNF + 1
                XK(NNF) = KK(I, M)
                XF(NNF) = F(I)
            END IF

            IF ( NNF.GE. 2 ) THEN
                XKNEXT = XK(NNF-1)
            END IF
        END DO
    END DO

```

```

1          + (XF(NNF)-XF(NNF-1))*
2          (XK(NNF-1)-XK(NNF-2))/
3          (XF(NNF-1)-XF(NNF-2))
          END IF

          IF( NNF.GT. 2 .AND.
1          XK(NNF) .LE. XK(NNF-1) ) THEN
              NNF = NNF -1
          END IF

```

```

C          *****
C          * END NFREQ LOOP, NF *
C          *****

```

END DO

CALL CURVE(XK(1),XF(1),NNF,0)

```

C          *****
C          * END MODES LOOP      *
C          *****

```

END DO

CALL ENDGR(1)

```

FF(1) = F(1)
FF(2) = F(1)
ZF(1) = 0.0
ZF(2) = ZT(NBV)
CALL PHYSOR(6.0,2.0)
CALL AREA2D(4.0,5.0)
CALL SERIF
CALL SHDCHR(0.0,1,.01,1)
CALL HEIGHT(.175)
CALL HEADIN(%REF('Brunt-Vaisala Frequency Profile$'),100,1.25,1)
CALL XNAME(%REF('Frequency$'),100)
CALL YNAME(%REF('Depth - m$'),100)
CALL GRAF(0.0,5.0,15.0,ZT(NBV),-500.0,0.0)
CALL FRAME
CALL CURVE(BVT,ZT,NBV,0)
CALL DASH
CALL CURVE(FF,ZF,2,0)
CALL ENDGR(2)
CALL ENDPL(0)
CALL DONEPL

```

RETURN

```

100  FORMAT(///'  FIRST FOUR MODES  '//
1      '  FREQUENCY = ',F15.5//
2      12X,'K(1)',11X,'K(2)',11X,'K(3)',11X,'K(4)'/
3      6X,4F15.5///'  AMPLITUDES  '//
4      5X,'Z',7X,'1',14X,'2',14X,'3',14X,'4'/)

```

```

110  FORMAT( 1X,F6.1,4F15.5)
      END

```

```

1      SUBROUTINE MODESUB ( NMODES, F, NBV, ZT, BVT, NZ, LAT, EPSLON,
                           IPRINT, K, Z, WM )

```

```

C*****
C
C  This routine computes the internal wave vertical modes and horizontal
C  wavenumbers for a prescribed Vaisala frequency profile, at a given set
C  of frequencies.  The ODE which is solved is
C
C
C      
$$w'' + [k^{*2}] * \frac{(N(z)^{*2} - F^{*2})}{(F^{*2} - Fi^{*2})} w = 0 ,$$

C
C  where w is vertical velocity, N(z) is Brunt-Vaisala frequency, k is
C  wavenumber, F is wave frequency, Fi is inertial frequency, and
C  z is depth.
C
C  The vertical modes W(z) generated by this code are in units of m/sec, and
C  the wavenumbers k**2 are in units of (radians/m)**2.  The normalization
C  of W(z) is such that the integral from bottom to surface of Potential +
C  Kinetic Energy is given by
C
C      Int[PE+KE]dz = Int[W(z)*{N(z)**2 - Fi**2}/{F**2 - Fi**2}]dz
C                   = No**2 * b**3 ,
C
C  where b = 1300 meters, and No is the scale Vaisala frequency =
C  3 cph * (2*pi/3600) (rad/cycle)*(hr/sec) = 5.24*10-4 rad/sec.
C
C  To produce the nondimensional normal modes Z(z) found in Garrett and
C  Munk (1972), one must divide W(z) by b*F, where b = 1300 m, and
C  F = frequency in radians per second.
C*****
C  David Rubenstein, Science Applications International Corp., Nov. 1987
C  Version for Lahey 77 Fortran, IBM-PC
C*****
C  Input Parameters
C
C  NMODES  Number of modes desired (Integer*4)
C  F       Wave Frequency, in cph (Real*4)
C  NBV     Number of points in BV profile (Integer*4)
C  ZT      Depths of BV frequencies, in meters (Real*4 array of length NBV)
C  BVT     Set of BV frequencies, in cph (Real*4 array of length NBV)
C  NZ      Preliminary estimate for number of points required in
C          vertical modes (Integer*4)
C  LAT     Latitude, in degrees (Real*4)
C  EPSLON  Relative accuracy required for determination of K**2 (Real*4)
C          Recommended value: 0.001
C  IPRINT  Print parameter.  Set = 0 for no diagnostics.
C
C  Output parameters
C
C  NZ      Actual number of points in computed vertical modes (Integer*4)
C  K       Wavenumber, in Radians/meter (Real*4 array of length NMODES)
C  Z       Depths, in meters, corresponding to vertical velocity modes
C          (Real*4 array of size NZ)
C  WM      Vertical velocity modes, in m/sec (Real*4 2-D array of size
C          NZ x NMODES)
C
C  Restrictions: Maximum value for NZ is MAX, and maximum value for NMODES
C  is MODEMAX, both of which are set in the parameter statement.
C
C  Suggestion on usage: Call this subroutine (MODESUB) once for each frequency*
C  desired, but start with the highest frequency and work downward.  This

```

```

C  subroutine tests for sufficient resolution, and the constraint is greatest *
C  at high frequencies. *
C *
C*****

```

```

PARAMETER ( MAX = 5000, MODEMAX = 20 )

```

```

REAL  BV(MAX), W(MAX), F, K2(0:MODEMAX), LAT, G(MAX)
REAL  Z(MAX), BVT(NBV), ZT(NBV), K2OLD, K2NEW
REAL  K2MAX, K2MIN, WM(MAX,0:MODEMAX), K(0:MODEMAX)

```

```

DATA  BDEP/1300.0/, NRES/5/

```

```

K2OLD = 0
K2NEW = 0
K2MAX = 0
K2MIN = 0
BDEP  = 1300.0
NRES  = 5

```

```

DEPTH = ZT(NBV)
PI     = 4.*ATAN(1.)
DEGRAD = PI/180.

```

```

FI = SIN(DEGRAD*LAT)/12.0

```

```

IF ( NMODES*NRES .GT. NZ ) THEN

```

```

    NZ = NRES*NMODES

```

```

    WRITE(*,25) NZ

```

```

25    FORMAT(' NZ has been adjusted to = ',i5)

```

```

ENDIF

```

```

28    CONTINUE

```

```

IF ( NZ .GT. MAX ) THEN

```

```

    WRITE(*,30)nz,max

```

```

30    FORMAT(' NZ = ',i5,' is > MAX = ',i4,'.',/,

```

```

1      ' Decrease NMODES or increase MAX.  Program aborting.')

```

```

    STOP

```

```

ENDIF

```

```

CALL INTERP ( NBV, ZT, BVT, NZ, DEPTH, Z, BV )

```

```

DZ = DEPTH/(NZ-1)

```

```

C *****
C * Test for resolution between turning *
C * points. NRES is the minimum number of *
C * vertical sampling intervals per mode *
C *****

```

```

DO 60 J = 1, NZ

```

```

    G(J) = (BV(J)**2 - F**2) / (F**2 - FI**2)

```

```

60    CONTINUE

```

```

CALL TURN( G, NZ, JA, JB, JM )

```

```

IF ( NMODES*NRES .GT. JB-JA ) THEN

```

```

    TYPE *, 'JA,JB',JA,JB

```

```

    TYPE *, 'F = ',F

```

```

    TYPE *, 'FI = ',FI

```

```

    WRITE(*,70) NMODES, NRES, JB-JA

```

```

70    FORMAT(' NMODES = ',i3,' * ',i3,' > JB-JA = ',i3,/,

```

```

1      ' Region between turning points insufficiently resolved.')

```

```

      NZ = NZ * 1.25
      WRITE(*,25) NZ

      GO TO 28
ENDIF

DO 80 J = 1, NZ
      G(J) = (BV(J)**2 - F**2) / (F**2 - FI**2)
80  CONTINUE

      CALL TURN( G, NZ, JA, JB, JM )
      CALL AVGIN( G, NZ, JA, JB, DZ, GAVG )

      AFACTOR = (PI/GAVG)**2
      I1 = 1
      N = NZ

C      *****
C      *      Loop through modes      *
C      *****

      DO 400 M = 0, NMODES

C      *****
C      *      First guess for k**2      *
C      *****

      K2OLD = AFACTOR*(M+1.5)**2
      IF(IPRINT.GE.1)WRITE(*,*)'Start iteration: k2old = ',k2old
      K2MAX = K2OLD*40.0
      K2MIN = K2OLD/40.0

      IF ( M .GT. 0 ) K2MIN = K2(M-1)

      ITERATE = 0
      NUMBER = 0
      K2NEW = K2MIN

C      *****
C      *      Top of Iteration loop      *
C      *****

100  CONTINUE

      IF(IPRINT.GE.1)WRITE(*,*)

      IF ( (K2MAX-K2MIN)/K2MAX .LT. EPSLON ) THEN
        IF(IPRINT.GE.1) WRITE(*,*)'Converged: k2min,k2max=',
1          k2min,k2max
          GO TO 115
      ENDIF

      ITERATE = ITERATE + 1

      IF ( ITERATE .GT. 200 ) THEN
        K(M) = 0.0
        GOTO 400
      END IF

      CALL NUMEROV ( M, I1, N, JA, JB, JM, G, W, DZ,
1          IPRINT, ICOUNT, ICROSS, K2OLD, K2NEW )

      IF ( IPRINT.GE.1 .OR. MOD(ITERATE,20).EQ.0 ) THEN
        WRITE(*,*)'ITERATE,M,ICOUNT,K2OLD,K2NEW,K2MIN,K2MAX = '
        WRITE(*,110)ITERATE,M,ICOUNT,K2OLD,K2NEW,K2MIN,K2MAX
110      FORMAT(3I4,2D15.4,5X,2D14.4)
      ENDIF

```

```

IF ( ICOUNT .NE. M ) THEN
  IF ( ICOUNT .LT. M ) THEN
    IF ( ICROSS .EQ. 0 ) THEN
      K2MIN = AMAX1(K2MIN,K2OLD)
    ELSE
      K2MIN = AMAX1(K2MIN,0.5*(K2MIN+K2OLD))
    ENDIF
  ELSE
    K2MAX = AMIN1(K2MAX,K2OLD)
  ENDIF

  IF ( ICROSS .EQ. 1 ) THEN
    NUMBER = NUMBER + 1
    DELTA1 = 0.5*(K2MAX+K2MIN) - K2OLD
    DELTA2 = K2OLD*((2.0*M+1.)/(2.0*ICOUNT+1) - 1.0)
    IF ( ABS(DELTA1) .GT. 2.*ABS(DELTA2) ) THEN
      K2OLD = K2OLD + DELTA2
    ELSE
      K2OLD = K2OLD + DELTA1
    ENDIF
    GO TO 100
  ENDIF

  IF ( ICOUNT .LT. M ) THEN
    K2OLD = 0.5*(K2MAX+K2OLD)
  ELSE
    K2OLD = 0.5*(K2MIN+K2OLD)
  ENDIF

  GO TO 100

```

```

*****
* END IF, ICOUNT *
*****

```

ENDIF

```

IF ( ABS((K2OLD-K2NEW)/K2NEW) .GT. EPSLON ) THEN
  IF ( ICROSS .EQ. 1 ) GO TO 100

  IF ( K2NEW .GT. K2OLD ) THEN
    K2MIN = K2OLD
  ELSE
    K2MAX = K2OLD
  ENDIF
  K2OLD = K2NEW
  GO TO 100
ENDIF

```

```

IF ( IPRINT .GE. 1 ) THEN
  WRITE(*,*)
  WRITE(*,*)'Converged: k2old, k2new=',k2old,k2new
ENDIF

```

```

*****
* Pad with zeros if bottom or top were *
* brought in
*****

```

115 CONTINUE

```

IF ( I1 .GT. 1 ) THEN
  IF ( IPRINT .GE. 1 ) WRITE(*,*)' ZERO-Pad: i1 = ',i1
  DO 135 I = 1, I1-1
    W(I) = 0.
  CONTINUE
ENDIF

```

135

```

      IF ( NZ .GT. N ) THEN
        IF ( IPRINT .GE. 1 ) WRITE(*,*)' ZERO-Pad:  N, NZ = ',N,nz
        DO 140 I = N+1, NZ
          W(I) = 0.
140      CONTINUE
      ENDIF

```

```

C
C
C
          *****
          *      Normalize w      *
          *****

```

```

      SUM = 0.
      DO 160 J = 2, NZ
        WT1 = (BV(J)**2 - FI**2) / (F**2 - FI**2)
        WT2 = (BV(J-1)**2 - FI**2) / (F**2 - FI**2)
        SUM = SUM + 0.5*(WT1*W(J-1)**2 + WT2*W(J)**2)*DZ
160      CONTINUE

```

```

      ANORM = ((2.*PI*3./3600. )**2) * (BDEP**3)
      ANORM = SQRT(ABS(ANORM/SUM))

```

```

      WM(1,M) = 0.
      WM(NZ,M) = 0.

```

```

      DO 180 J = 2, NZ-1
        WM(J,M) = ANORM*W(J)
180      CONTINUE

```

```

      K2(M) = K2NEW
      K(M) = SQRT(ABS(K2(M)))

```

```

      IF ( IPRINT .GE. 1 ) THEN
        WRITE(*,*)'Found mode #',m
        write(*,*)'Frequency=',f,'      K**2 = ',k2new
        WRITE(*,200)(WM(I,M), I = 1, NZ)
200      FORMAT(1X,6F12.7)
      ENDIF

```

```

C
C
C
          *****
          * Adjust afactor for next higher mode*
          *****

```

```

      AFACTOR = K2OLD/(M+1.5)**2

```

```

400      CONTINUE

```

```

      RETURN
      END

```

SUBROUTINE TURN (G, NZ, JA, JB, JM)

```

C*****
C
C      SUBROUTINE TURN
C
C*****

```

REAL G(NZ)

```

*****
* Find maximum *
*****

```

JM = 1
GMAX = G(1)

DO 10 I = 2, NZ
IF (G(I) .GT. GMAX) THEN
JM = I
GMAX = G(I)

10 CONTINUE

IF (JM .LE. 2) THEN
IF (G(I) .GT. 0.25*GMAX) THEN
JM = 3

ELSE
WRITE(*,*)'*** Peak too close to surface. Increase NZ.***'
WRITE(*,*)'jm = ',jm,' gmax=',gmax,' g:'
WRITE(*,*)(G(I),I=1,NZ)
STOP 10

ENDIF
ENDIF

```

*****
* Find upper turning point *
*****

```

DO 20 I = JM, 1, -1
IF (G(I) .GT. 0.) JA = I
20 CONTINUE

```

*****
* Find lower turning point *
*****

```

DO 30 I = JM, NZ
IF (G(I) .GT. 0.) JB = I
30 CONTINUE

RETURN
END

```
subroutine avgint ( g, nz, ja, jb, dz, gavg )
```

```
C*****  
C  
C      subroutine avgint  
C  
C*****
```

```
C  
C  
C      *****  
C      * Integrate g(z) from index j = ja *  
C      * to jb, and get average *  
C      *****
```

```
REAL G(NZ) .
```

```
GAVG = 0.
```

```
DO 20 J = JA, JB
```

```
    GAVG = GAVG + SQRT(ABS(G(J)))  
20 CONTINUE
```

```
GAVG = GAVG*DZ
```

```
RETURN  
END
```



```

        IF ( ARG .GT. 1.E6 ) THEN
            I1 = I
            GO TO 10
        ENDIF
30      CONTINUE
    ENDIF

    IF ( ICOUNT .GT. M ) THEN
        IF ( IPRINT .GE. 1 ) WRITE(*,*) 'ICOUNT > M IN PHIM'
        GO TO 600
    ENDIF

```

C
C
C

```

*****
* END LOOP, JBOT *
*****

```

40 CONTINUE

JTOP = N - 2

```

DO 60 J = JTOP, JM-2, -1
    PHIP(J) = ( (2. + 10.*T(J+1))*PHIP(J+1) +
1          (T(J+2) - 1.)*PHIP(J+2) ) / (1.-T(J))
    IF( J.GE.JM .AND. PHIP(J+1)*PHIP(J).LE.0. ) ICOUNT = ICOUNT+1

```

C
C
C
C

```

*****
* Bring in bottom if exponential growth is *
* sufficiently strong *
*****

```

```

1      ARG = (ABS(PHIP(J))+ABS(PHIP(J+1))) /
            (ABS(PHIP(N-1))+ABS(PHIP(N-2)))
    IF ( ARG .GT. 1.E12 ) THEN
        ITOP = N - 1
        DO 50 I = J, ITOP
            ARG = (ABS(PHIP(J))+ABS(PHIP(J+1))) /
1          (ABS(PHIP(I))+ABS(PHIP(I+1)))
            IF ( ARG .GT. 1.E6 ) THEN
                N = I
                GO TO 10
            ENDIF
50      CONTINUE
    ENDIF

```

50

ENDIF

```

    IF ( ICOUNT .GT. M ) THEN
        IF ( IPRINT .GE. 1 ) WRITE(*,*) 'ICOUNT > M IN PHIP'
        GO TO 600
    ENDIF

```

C
C
C

```

*****
* END LOOP, JTOP *
*****

```

60 CONTINUE

C
C
C

```

*****
* Does zero-crossing occur at match-point? *
*****

```

```

    IF ( PHIP(JM)*PHIM(JM) .LE. 0.0 ) THEN
        ICROSS = 1
        ICOUNT = ICOUNT + 1
        IF ( IPRINT .GE. 1 ) WRITE(*,*) 'zero-crossing at j=jm=',
1          jm, ' icount=', icount
        IF ( ICOUNT .NE. M ) THEN
            IF ( IPRINT .GE. 1 ) WRITE(*,*) 'icount <> m at match point'
            GO TO 600
        ENDIF
    ENDIF

```

ENDIF

```
C
C
C
*****
*   Look for sign-match   *
*****

1  IF ( PHIP(JM-1)*PHIM(JM).GT.0.0 .OR.
2    PHIP(JM-2)*PHIM(JM).GT.0.0 .OR.
3    PHIP(JM)*PHIM(JM+1).GT.0.0 .OR.
    PHIP(JM)*PHIM(JM+2).GT.0.0 ) THEN
    JM1 = JM + 0.5*(JB-JA)/(M+2.)
    JM2 = JM - 0.5*(JB-JA)/(M+2.)
    IF ( JM1 .GT. JA .AND. JM1 .LT. JB ) THEN
      JM = JM1
    ELSE
      IF ( JM2.GT.JA .AND. JM2.LT.JB ) THEN
        JM = JM2
      ELSE
        JM = 0.5*(JM + JB)
      ENDIF
    ENDIF
    IF ( IPRINT .GE. 1 ) WRITE(*,*)
1    'Sign match found.  New jm=',jm
    GO TO 700
  ELSE
    ICOUNT = M - 1
    GO TO 600
C
C
C
*****
* END IF, SIGN MATCH *
*****

    ENDIF

C
C
C
*****
* END IF, ZERO-CROSSING *
*****

    ENDIF

    IF ( ICOUNT .NE. M ) GO TO 600

C
C
C
*****
* Adjust phi by a factor *
*****

    FACT = PHIP(JM)/PHIM(JM)

    IF ( FACT .GT. 1. ) THEN
      DO 70 J = JM-2, N
        PHIP(J) = PHIP(J)/FACT
70      CONTINUE
      ELSE
        DO 80 J = 2, JM+2
          PHIM(J) = FACT*PHIM(J)
80      CONTINUE
    ENDIF

C
C
C
*****
* Integrate phi**2 *
*****

    PHI2 = 0.

    DO 100 J = 2, JM
      PHI2 = PHI2 + G(J-1)*PHIM(J-1)**2+G(J)*PHIM(J)**2
100    CONTINUE
```

```

DO 110 J = JM+1, N
  PHI2 = PHI2 + G(J-1)*PHIP(J-1)**2+G(J)*PHIP(J)**2
110 CONTINUE

```

```

PHI2 = 0.5*DZ*PHI2

```

```

C *****
C * Compute phip' and phim' *
C *****

```

```

A1P = 0.5*(PHIP(JM+1)-PHIP(JM-1))
A2P = 0.5*(PHIP(JM+2)-PHIP(JM-2))
A1M = 0.5*(PHIM(JM+1)-PHIM(JM-1))
A2M = 0.5*(PHIM(JM+2)-PHIM(JM-2))
B1P = T(JM+1)*PHIP(JM+1) - T(JM-1)*PHIP(JM-1)
B2P = T(JM+2)*PHIP(JM+2) - T(JM-2)*PHIP(JM-2)
B1M = T(JM+1)*PHIM(JM+1) - T(JM-1)*PHIM(JM-1)
B2M = T(JM+2)*PHIM(JM+2) - T(JM-2)*PHIM(JM-2)

```

```

1 PHIPPR = (16./(21.*DZ))* ( -A1P + (37./32.)*A2P - (37./5.)*B1P
  - (17./40.)*B2P )
1 PHIMPR = (16./(21.*DZ))* ( -A1M + (37./32.)*A2M - (37./5.)*B1M
  - (17./40.)*B2M )

```

```

DO 120 J = 1, JM
  W(J) = PHIM(J)
120 CONTINUE

```

```

DO 140 J = JM+1, N
  W(J) = PHIP(J)
140 CONTINUE

```

```

C *****
C * Get new trial value for k**2 *
C *****

```

```

K2NEW = K2OLD - W(JM)*(PHIPPR - PHIMPR) / PHI2

```

```

RETURN

```

```

C *****
C * Early return *
C *****

```

```

600 CONTINUE

```

```

DO 620 J = 1, JM
  W(J) = PHIM(J)
620 CONTINUE

```

```

DO 640 J = JM+1, N
  W(J) = PHIP(J)
640 CONTINUE

```

```

700 K2NEW = 1.E-20

```

```

RETURN
END

```

```
subroutine interp ( n, z, x, ni, ztotal, zi, xi )
```

```

C*****
C
C Title:      Interp
C
C Purpose:   Interpolate function x(z), from depth z=0 to z=ztot+al
C
C*****
C
C
C Input parameters:
C
C N          Length of arrays X and Z
C Z          Real*4 array of length N
C X          Real*4 array of length N
C NI         Length of desired output arrays ZI and XI
C ZTOTAL     Total depth to which interpolated output is desired
C
C Output parameters:
C
C ZI         Regular (Real*4) interval array, ranging from 0 to ZTOTAL,
C            of length NI
C XI         Interpolated values (Real*4 array of length NI
C
C*****

```

```
REAL Z(1), X(1), ZI(1), XI(1)
```

```
DZ = ZTOTAL/(NI-1)
```

```
J = 1
```

```
DO 50 I = 1, NI
```

```
    ZI(I) = (I-1)*DZ
```

```
40    CONTINUE
```

```
    IF ( ZI(I) .GE. Z(J) .AND. ZI(I) .LE. Z(J+1)) THEN
```

```
        XI(I) = X(J) + (X(J+1)-X(J))*(ZI(I)-Z(J))
        / (Z(J+1)-Z(J))
```

```
1
```

```
    ELSE
```

```
        J = J + 1
```

```
        IF ( I.EQ.NI .AND. ABS(ZI(I)-Z(J)).LE.0.01 ) THEN
```

```
            ZI(I) = Z(J)
```

```
            XI(I) = X(J)
```

```
            RETURN
```

```
        ENDIF
```

```
        IF ( J .GT. N ) STOP 40
```

```
        GO TO 40
```

```
    ENDIF
```

```
50    CONTINUE
```

```
RETURN
```

```
END
```

```

SUBROUTINE INTRPL(IU,L,X,Y,N,U,V)
C*****
C
C PROGRAM          INTRPL
C
C PURPOSE          INTERPOLATION OF A SINGLE VALUED FUNCTION.
C                  THIS SUBROUTINE INTERPOLATES, FROM VALUES OF THE
C                  FUNCTION GIVEN A ORDINATES OF INPUT DATA POINTS IN
C                  THE X-Y PLANE AND FOR A GIVEN SET OF X-VALUES(ABCISSAS),
C                  THE VALUES OF A SINGLE VALUED FUNCTION Y=Y(X).
C
C AUTHOR           HIROSHI AKIMA,U.S.DEPT OF COMMERCE,OFFICE OF
C                  TELECOMMUNICATIONS, INSTITUTE OF TELECOMMUNICATIONS
C                  SCIENCES, BOULDER COLO
C                  THIS ALGORITHM WAS PUBLISHED IN COMM. ACM. 15(10)
C                  OCT 1972
C*****
C
C INPUT PARAMETERS ARE
C   IU = LOGICAL UNIT NUMBER OF STANDARD OUTPUT UNIT
C   L  = NUMBER OF INPUT DATA POINTS
C   X  = ARRAY OF DIMENSION L STORING THE X VALUES
C        (ABCISSAS) OF THE DATA POINTS IN ASCENDING ORDER
C   Y  = ARRAY OF DIMENSION L STORING THE Y VALUES
C        (ORDINATES) OF THE INPUT DATA POINTS
C   N  = NUMBER OF POINTS AT WHICH INTERPOLATION OF THE
C        Y VALUES (ORDINATE) IS DESIRED
C   U  = ARRAY OF DIMENSION N STORING THE X VALUES OF THE
C        DESIRED POINTS.
C
C OUTPUT PARAMETERS
C   V  = ARRAY OF DIMENSION N WHERE THE INTERPOLATED Y
C        VALUES ARE STORED
C*****

      DIMENSION X(1),Y(1),U(1),V(1)

      EQUIVALENCE (P0,X3),(Q0,Y3),(Q1,T3)

      REAL M1,M2,M3,M4,M5

      EQUIVALENCE (UK,DX),(IMN,X2,A1,M1),(IMX,X5,A5,M5),
1          (J,SW,SA),(Y2,W2,W4,Q2),(Y5,W3,Q3)

C*****
C                  PRELIMINARY PROCESSING
C*****

10      L0 = L
        LM1 = L0-1
        LM2 = LM1-1
        LP1 = L0+1
        N0 = N

        IF( LM2 .LT. 0 )      GO TO 90
        IF( N0 .LE. 0 )      GO TO 91

        DO 11 I=2,L0

```

11 IF(X(I-1)-X(I)) 11, '5,96
CONTINUE

IPV = 0

C *****
C * MAIN DO LOOP *
C *****

DO 80 K = 1,N0
UK=U(K)

C *****
C * ROUTINE TO LOCATE DESIRED POINT *
C *****

20 IF(LM2 .EQ. 0) GO TO 27
IF(UK .GE. X(L0))GO TO 26
IF(UK .LT. X(1)) GO TO 25
IMN = 2
IMX = L0

21 I = (IMN+IMX)/2
IF(UK .GT. X(I)) GO TO 23

22 IMX = I
GO TO 24

23 IMN = I + 1

24 IF(IMX .GT. IMN) GO TO 21
I = IMX
GO TO 30

25 I=1
GO TO 30

26 I = LP1
GO TO 30

27 I=2

C *****
C * CHECK IF I = IPV *
C *****

30 IF(I .EQ. IPV) GO TO 70
IPV = I

C *****
C * ROUTINES TO PICK UP NECESSARY X *
C * AND Y VALUES AND TO ESTIMATE THEM *
C * IF NECESSARY *
C *****

40 J =I
IF(J.EQ.1) J=2
IF(J.EQ.LP1) J=L0

X3 = X(J-1)
Y3 = Y(J-1)
X4 = X(J)
Y4 = Y(J)
A3 = X4-X3
M3 = (Y4-Y3)/A3

IF(LM2 .EQ. 0) GO TO 43
IF(J .EQ. 2) GO TO 41

```

X2 = X(J-2)
Y2 = Y(J-2)
A2 = X3-X2
M2 = (Y3-Y2)/A2

41 IF(J .EQ. L0) GO TO 42
X5 = X(J+1)
Y5 = Y(J+1)

A4 = X5-X4
M4 = (Y5-Y4)/A4
IF(J .EQ. 2) M2=M3 + M3 - M4
GO TO 45

42 M4 = M3+M3-M2
GO TO 45

43 M2 = M3

45 IF(J .LE. 3) GO TO 46

A1 = X2-X(J-3)
M1 = (Y2-Y(J-3))/A1
GO TO 47

46 M1 = M2+M2-M3

47 IF(J .GE. LM1) GO TO 48
A5 = X(J+2) - X5
M5 = (Y(J+2) - Y5)/A5
GO TO 50

48 M5=M4+M4-M3

C *****
C * NUMERICAL DIFFERENTIATION *
C *****

50 IF( I .EQ. LP1) GO TO 52
W2 = ABS(M4-M3)
W3 = ABS(M2-M1)
SW = W2+W3
IF(SW .NE. 0.0) GO TO 51

W2 = 0.5
W3 = 0.5
SW = 1.0

51 T3 = (W2*M2+W3*M3)/SW
IF(I .EQ. 1) GO TO 54

52 W3 = ABS(M5-M4)
W4 = ABS(M3-M2)
SW = W3+W4
IF(SW .NE. 0.0) GO TO 53

W3 = 0.5
W4 = 0.5
SW = 1.0

53 T4=(W3*M3+W4*M4)/SW

IF(I .NE. LP1) GO TO 60
T3 = T4
SA = A2 + A3

```

```

T4 = 0.5*(M4+M5-A2*(A2-A3)*(M2-M3)/(SA*SA))
X3 = X4
Y3 = Y4
A3 = A2
M3 = M4
GO TO 60
54 T4=T3

SA = A3+A4
T3 = 0.5*(M1+M2-A4*(A3-A4)*(M3-M4)/(SA*SA))
X3 = X3 - A4
Y3 = Y3 - M2*A4
A3 = A4
M3 = M2

C *****
C * DETERMINATION OF THE COEFFICIENTS *
C *****

60 Q2 = (2.0*(M3-T3)+M3 - T4)/A3
   Q3 = (-M3-M3+T3+T4)/(A3*A3)

C *****
C * COMPUTATION OF THE POLYNOMIAL *
C *****

70 DX = UK-P0

80 V(K) = Q0+DX*(Q1+DX*(Q2+DX*Q3))

RETURN

C *****
C * ERROR EXITS *
C *****

90 WRITE(IU,2090)
   GO TO 99

91 WRITE(IU,2091)
   GO TO 99

95 WRITE(IU,2095)
   GO TO 97

96 WRITE(IU,2096)

97 WRITE(IU,2097) I,X(I)

99 WRITE(IU,2099) L0,N0
   RETURN

2090 FORMAT(1X,' *** L = 1 OR LESS'/)
2091 FORMAT(1X,' *** N = 0 OR LESS'/)
2095 FORMAT(1X,' *** IDENTICAL X VALUES'/)
2096 FORMAT(1X,' *** X VALUES OUT OF SEQUENCE'/)
2097 FORMAT(1X,' I=',I7,10X,'X(I) =',E12.3/)
2099 FORMAT(1X,' I=',I7,10X,'N =',I7/
1 1X,'*****ERROR DETECTED IN ROUTINE INTRPL*****'//)

END

```

FUNCTION BVFRQ(S,T,P,NOBS,PAV,E)

```

C*****
C
C PROGRAM          BVFRQ
C
C PURPOSE          COMPUTES Brunt-Väisälä frequency in CPH
C
C AUTHOR           R. MILLARD, WOODS HOLE OCEANOGRAPHIC INSTITUTION
C
C NOTES:
C
C   USES 1980 EQUATION OF STATE
C
C   UNITS:
C     PRESSURE      P0      DECIBARS
C     TEMPERATURE   T       DEG CELSIUS (IPTS-68)
C     SALINITY      S       (IPSS-78)
C     BOUYANCY FREQ BVFRQ   CPH
C     N**2          E       RADIANS/SECOND
C
C CHECKVALUE: BVFRQ=14.57836 CPH E=6.4739928E-4 RAD/SEC.
C             S(1)=35.0, T(1)=5.0, P(1)=1000.0
C             S(2)=35.0, T(2)=4.0, P(2)=1002.0
C *****NOTE RESULT CENTERED AT PAV=1001.0 DBARS *****
C JULY 12 1982
C COMPUTES N IN CYCLES PER HOUR, AND E=N**2 IN RAD/SEC**2
C AFTER FORMULATION OF BRECK OWEN'S & N.P. FOFONOFF
C
C*****
C   IMPLICIT NONE
C
C   REAL*4      P(1),T(1),S(1)
C
C   REAL*4      E,BVFRQ,CXX,CX,CXY,CY,PAV,DATA,V350P,VBAR,
1  SIG,DVDP,A0
C
C   REAL*4      SVAN,THETA
C
C   INTEGER*4   NOBS,K
C
C   EXTERNAL    SVAN,THETA
C
C   E = 0.0
C   BVFRQ = 0.0
C
C   IF (NOBS.LT.2) RETURN
C
C   CXX = 0.0
C   CX  = 0.0
C   CXY = 0.0
C   CY  = 0.0
C
C
C   *****
C   * COMPUTE LEAST SQUARES ESTIMATE OF SPECIFIC *
C   * VOLUME ANAMOLY GRADIENT *
C   *****
C
C   DO 20 K=1,NOBS
C     CX =CX+P(K)
20  CONTINUE
C
C   PAV=CX/NOBS
C
C   DO 35 K=1,NOBS
C     DATA = SVAN(S(K),THETA(S(K),T(K),P(K),PAV),PAV,SIG)*1.0E-8

```

```

      CXY = CXY+DATA*(P(K)-PAV)
      CY  = CY+DATA
      CXX = CXX+(P(K)-PAV)**2
35  CONTINUE

      IF(CXX.EQ.0.0) RETURN

      A0   = CXY/CXX
      V350P = (1./(SIG+1000.))-DATA
      VBAR  = V350P+CY/NOBS
      DVDP  = A0

      IF(VBAR.EQ.0.0) RETURN

      E      = -.96168423E-2*DVDP/(VBAR)**2
      BVFRQ  = 572.9578*SIGN(SQRT(ABS(E)),E)

      RETURN
      END

```

```

      FUNCTION GRADY(Y,P,NOBS,PAV,YBAR)

```

```

C*****
C
C  TITLE:  GRADY
C
C  PURPOSE: FUNCTION COMPUTE LEAST SQUARES SLOPE 'GRADY' OF Y VERSUS P
C            THE GRADIENT IS REPRESENTATIVE OF THE INTERVAL CENTERED AT
C            PAV
C            COMPUTE GRADIENT OF Y VERSUS P
C
C  DATE:    JULY 15 1982
C
C*****

```

```

      REAL*4 P(1),Y(1)

      GRADY = 0.0
      A0    = 0.0
      CXX   = 0.0
      CX    = 0.0
      CXY   = 0.0
      CY    = 0.0

      IF(NOBS.LE.1) GO TO 30

      DO 20 K=1,NOBS
20      CX =CX+P(K)

      PAV = CX/NOBS

      DO 35 K=1,NOBS
      CXY = CXY+Y(K)*(P(K)-PAV)
      CY  = CY+Y(K)
      CXX = CXX+(P(K)-PAV)**2
35  CONTINUE

      IF(CXX.EQ.0.0) RETURN

      A0   = CXY/CXX
      YBAR = CY/NOBS

```

30 CONTINUE

GRADY=A0

RETURN
END

OCEAN SIMULATION MODELING

DISTRIBUTION:

Naval Oceanographic and Atmospheric Research Laboratory
Stennis Space Center, MS 39529-5004

ATTN: Code 110
111
113
115 (Meert, Lundberg, Stanley, Goggins, Winchester)
200
220
221 (Norton)
224 (Campbell)
240 (Farwell)
242
244 (Slater)
245 (Wagstaff)
250
300
310
311 (Ferer, Selsor)
320
321 (Pressman, Holyer, Mitchell, Smith, Lybanon)
322 (Harding, Preller, Martin, Carnes)
323 (Heburn, Blake, Hurlburt, Thompson, Kindle)
330
331 (Hollman, Teague, Hallock, Boyd, Pickett, Burns)
332
333
350
351 (Walker)
352 (Mozley, Arnone)
360
362
125L (10)
125P

NOARL
Code 400 (Hovermale, Tag, Brand, Haggerty)
Monterey, CA 93943-5006

INO
Attn: (B. Willems, J. Leese, L. Kantha, E. Johnson,)

Naval Ocean Systems Center
Attn: (J. Richter Code 17)
San Diego, CA 92152

Office of Naval Technology

Attn: (CDR L. Bounds)
(Dr. M. Brisco)
(Dr. C. Votaw)
(Dr. P. Selwyn)

800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research

Attn: (Dr. A. Brand)
(Dr. A. Weinstein)

800 N. Quincy Street
Arlington, VA 22217-5000

Space and Naval Warfare Systems Command

Attn: Code 315, 312, PD 80
Washington, DC 20363-5000

Naval Underwater Systems Center

Attn: J. Keil, Code 60
Newport, RI 02841-5047

Naval Coastal Systems Center

Attn: D. Sheppard, Code 10
Panama City, FL 32407-5000

Naval Air Development Center

Attn: R. Becker, Code 30
Warminster, PA 18974

Naval Weapons Center

Attn: P. Arnold, Code 30
China Lake, CA 93555-6001

Naval Surface Weapons Center

Attn: A. Glazman, Code D25
White Oak
Silver Spring, MD 20910

David W. Taylor Naval Research Center

Attn: S. Goldstein Code 1203
Bethesda, MD 20884-5000

Oceanographer of the Navy

Chief of Naval Operations

Attn: OP-096 (R. Winokur, ADM Pittenger)

U.S. Naval Observatory
34th & Mass Ave, NW
Washington, DC 20392-5100

Chief of Naval Operations
Department of the Navy
Attn: OP21T
OP21T2
Washington, DC 20350-2000

Center for Naval Analyses
Attn: (R. Bronowitz)
4401 Ford Avenue
Alexandria, VA 22302

Superintendent
Naval Postgraduate School
Monterey, CA 93943

Commander, Navy Sea Systems Command
Naval Sea Sys Com Headquarters
Washington, DC 20362-5101

Commander
Naval Air Systems Command
Naval Air Sys Com Headquarters
Washington, DC 20361-0001

Asst Secretary of the Navy
(Research, Engineering & Systems)
Navy Dept
Washington, DC 20350-2000

John Hopkins University
Applied Physics Laboratory
John Hopkins Road
Laurel, MD 20707

SAIC
Attn: (Dr. D. Rubenstein)
P.O. Box 1303
1710 Goodridge Dr.
McLean, VA 22102

Sverdrup Technology
Attn: (Dr. Ransford)
Stennis Space Center, MS 39529

Admiralty Research Establishment
Attn: (Dr. John Scott)
Southwell, Portland
Dorset DT5-2JS
United Kingdom

SACLANT Research Center
Attn: (Dr. Henry T. Perkins)
400 Via San Bartolomeo
19026 La Spezia, Italy

Oregon State University
Attn: (Dr. Robert Miller)
College of Oceanography
Oceanography Admin. Bldg. 104
Corvallis, OR 97331

REPORT DOCUMENTATION PAGE			Form Approved OBM No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. Agency Use Only (Leave blank).		2. Report Date. July 1990		3. Report Type and Dates Covered.
4. Title and Subtitle. Ocean Simulation Model for Internal Waves Computer Source Code			5. Funding Numbers. Program Element No. 0602435N Project No. 03590 Task No. 0M0G Accession No. DN258014	
6. Author(s). K. D. Saunders, S. A. Briggs and D. Rubenstein*				
7. Performing Organization Name(s) and Address(es). Naval Oceanographic and Atmospheric Research Laboratory Ocean Science Directorate Stennis Space Center, Mississippi 39529-5004			8. Performing Organization Report Number. NOARL Technical Note 59	
9. Sponsoring/Monitoring Agency Name(s) and Address(es). Naval Oceanographic and Atmospheric Research Laboratory Requirements and Assessment Office Stennis Space Center, Mississippi 39529-5004			10. Sponsoring/Monitoring Agency Report Number. NOARL Technical Note 59	
11. Supplementary Notes. *Science Applications International Corporation, McLean Virginia				
12a. Distribution/Availability Statement. Approved for public release; distribution is unlimited.			12b. Distribution Code.	
13. Abstract (Maximum 200 words). This technical note contains the source code for the first level ocean simulation model and associated test and display programs. This model provides simulations of internal wave activity based on average oceanographic conditions at a given location. The code is written in FORTRAN 77 and should be easily ported to a wide variety of computers and operating systems. This technical note is intended primarily for persons implementing and/or modifying the code on their own systems.				
14. Subject Terms. (U) dynamical oceanography; (U) physical oceanography; (U) ocean simulation (U) modeling			15. Number of Pages. 156	
			16. Price Code.	
17. Security Classification of Report. Unclassified	18. Security Classification of This Page. Unclassified	19. Security Classification of Abstract. Unclassified	20. Limitation of Abstract. SAR	