A

P-Vector Module for z-Coordinate

The P-Vector module for z-coordinate was developed to compute absolute velocity from hydrographic data. This package is stored in the enclosed DVD-Rom and includes the makefile, source codes, plotting files (using Matlab), and a sample case. The sample case is located at a subdirectory called sample.dir. The test data set is the WOA climatological (T, S) fields for the North Atlantic Ocean. The gird resolution is set to be $1^\circ \times 1^\circ$ with 50 vertical levels (i.e., $n_x = 360, n_y = 180, n_z = 50$). The users may change for their own sakes. Chenwu Fan and Carlos Lozano helped in developing these codes.

A.1 Makefile

```
*******************************************************
SRC = whoi.f pvector.f pv.f pvpar.f userinp.f setmask.f matopen.f
matclose.f matotal.f matout.f

OBJS= $(SRC:.f=.o)
FFLAGS= -g

pvexe:    $(OBJS)
          f77 $(FFLAGS) -o pvexe $(OBJS)
pvector.o:    pstate.h pvector.f
          f77 $(FFLAGS) -c pvector.f
pv.o:    pstate.h pv.f
          f77 $(FFLAGS) -c pv.f
userinp.o:    pstate.h userinp.f
          f77 $(FFLAGS) -c userinp.f
whoi.o:    whoi.f
          f77 -g -c whoi.f
pvpar.o:    pstate.h pvpar.f
          f77 -g -c pvpar.f
setmask.o:    setmask.f
```

```
        f77 -g -c setmask.f matopen.o:    matopen.f
        f77 -g -c matopen.f matclose.o:    matclose.f
        f77 -g -c matclose.f
matotal.o:    matotal.f
        f77 -g -c matotal.f
matout.o:    pstate.h matout.f
        f77 $(FFLAGS) -c matout.f
```

A.2 Main Subdirectories

There are three major directories (*pvector*, *pvector_nc* and *pvector_rotation*) of the enclosed DVD-Rom for computing the absolute velocity in z-coordinate system. Among them, *pvector* and *pvector_nc* are used for (x, y)-axes aligning with (latitude, longitude); and *pvector_rotation* is used for (x, y)-axes not aligning with (latitude, longitude). The output of *pvector* is in the ASCII format. The output of *pvector_nc* is in the NetCDF format.

A.2.1 Subdirectory (pvector)

In this subdirectory, the input and output data are in the ASCII format. All the input and output files are arranged from west to east and norht to south. A sample data set (North Atlantic) is located in sample.dir. The following procedures are taken for running the sample case. The commands are listed using *italics*. Since all the data files are in the ASCII format, running this program is slow and needs large disk space.

(a) Create pvexe (executive file) in the pvectorcode director

$< make$

(b) Go to sample.dir and run pvexe

../pvexe $<$ *pv.inp*

for getting ASCII output data.

(c) Use the following matlab files to plot the output data.

uvlev.m - plot vertical velocity at different levels
tsrlev.m - plot temp, salt and density at different levels.

(d) Note that the input data order as:

West to east, top to bottom, north to south.

(e) Note that the output data order as:

Top to bottom, west to east, north to south.

A.2.2 Subdirectory (pvector_nc)

In this subdirectory, the input data are either in the ASCII or binary format, and the output in the NetCDF format. You must have NetCDF library in your system. If you do not have the NetCDF lib (library), you may download the netcdf library and the matlab-NetCDF interface from http://www.unidata.ucar.edu/software/netcdf/.

You may need to change the location of netcdf lib in the Makefile *mknc*. You can use *mknc* to compile the code in code_nc subdirectory. Two sample sets are located in the two subdirectories *Sample_NA_ASCII* and *Sample_NA_bin*. *Sample_NA_ASCII* contains input data in the ASCII format for the North Atlantic Ocean (similar to pvector/sample.dir) and one output file in the NetCDF format. *Sample_NA_bin* contains input data in the binary format for the North Atlantic Ocean and the output file in the NetCDF format. For the input file pvnc.inp, note that (a) ts (the first number on line 3): 1-ASCII data, 2-binary, (b) line 8: description of the netcdf file, and (c) line 9: the author.

A.2.3 Subdirectory (pvector_rotation)

When the horizontal coordinate system is not aligned with the longitude and latitude, you should use the subdirectory *pvector_rotation*. Different from *pvectorcode* and *pvector_nc*, use of *pvector_rotation* needs input of longitude and latitude data at each station. Line 2 of *pvxy.inp* is used for naming the 2D longitude and latitude data file. A sample set (named Sample_Arctic) is taken as an example with *Arctic Ocean.inp* as the control data file. The second line in that file shows the latitude, longitude, y, x, and name of data file. The data file can be either binary or ASCII data. The first number of line 3, 'ts' is the key word of the input data type:

topography data always in ASCII format.
temperature and salinity data:

ts	temperature and salinity data type
1	ascii data with order (mx,mz,my)
-1	ascii data with order (mz,mx,my)
2	binary data with order (mx,mz,my)
-2	binary data with order (mz,mx,my)

latitude, longitude, x, and y data:

```
      |ts|=1        ASCII data
      |ts|=2        binary data
   and with the order
                    lat(mx, my)
                    lon(mx, my)
                    x(mx, my)
                    y(mx, my)
```

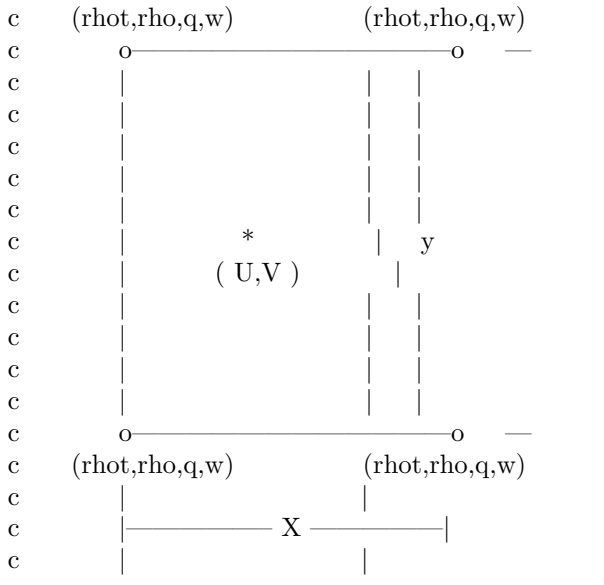
It is noted that the output velocity (u, v) is in the directions of (x, y) axes not of longitude and latitude.

A.3 Major FORTRAN Programs

A.3.1 ASCII Format (pv.f)

This FORTRAN program is used for the coordinate system with the latitude as the x-axis and the longitude as the y-axis.

```
      program pv

c
c    — tempfile: temperature data file name
c    — saltfile: salitnity data file name
c    — topofile: topography data file name
c     The Grid desired as:
c
c      (rhot,rho,q)              (rhot,rho,q)
c          o————————*————————o      —
c          |       ( U,V )       |     |
c          |                     |     |
c          |                     |     |
c          |                     |     |
c          |                     |
c          *(w)                 (w)*     Z
c          |                     |
c          |                     |     |
c          |                     |     |
c          |                     |     |
c          |       ( U,V )       |     |
c          o————————*————————o      —
c      (rhot,rho,q)              (rhot,rho,q)
c          |                     |
c          |———————— X , Y ——————|
c          |                     |
c
```




```
c
c   Notes:
c   1. Setup.
c     For a given application the program is configured
c     internally by defining the parameters from the input data.
c     See pv.inp in the subdirectory smaple.dir.
c
c     mx,my,mz (the dimension for rho,rhot,q,temp,salt)
c     mx1=mx-1,my1=my-1,mz1=mz-1 (the dimension for u,v,w)
c
c     mx is the number of longitudinal points.
c     my is the number of latitudinal points.
c     mz is the number of depth points.
c
c     The following data should be filled up by user in the
c     subroutine pvpar
c
c     The point (1,1,1) is at the surface in the North West
c     edge of the grid.
c
c     spherical             [(0) dx=dy, (1) dx(j)=dy*cos(lat(j)]
c     global/regional       [(1) global, (0) regional]
c
c     grid_size (in degree)  [grid size spacing is uniform
c                       along longitude and latitude]
c
c     deep (1:mz)                [depth of z-levels (m).
c                         The origin is at the ocean surface
c                         and z is negative downwards.]
```

```
c       lat0                    [latitude (degree) at the NW corner]
c
c       lon0                    [longitude (degree) at the NW corner]
c
c       ktop                    [In the inversion procedure use
c                               only the levels [ktop:mz]]
c
c       ts                      read either TS or RHO RHOT
c          1                    read temp & salt from ascii data files
c          2                    read temp & salt from binary data files
c
c
c
c   2. Inputs:
c
c       stopog(mx,my)           topography for the subroutine read_topo
c                               see description in the subroutine header.
c
c       if(ts>=1):
c       temp(mx,my,mz)          temperature
c       salt(mx,my,mz)          salinity
c                               read in subroutine read_ts
c       if(ts==2):              Temperature and salinity show non-number.
c                               The bottom topography is reset at the same
c                               depth with T and S taking non-numbers.
c                               Read in the subroutine read_ts.
c       if(ts=0):
c       rho(mx,my,mx)           density
c       rhot(mx,my,mz)          potential density
c
c    3. Outputs:
c
c      u(mx1,my1,mz),v(mx1,my1,mz) absolute horizontal velocity
c      (m/s)
c      px(mx1,my1,mz),py(mx1,my1,mz) P-vector
c      invmask(mx1,my1)
c
c                          =-1        for inverse not being accomplished
c                          =0         for land
c                          >0         for inverse being accomplished
c
c      [1:invmask(i,j)]
c                          levels at which the absolute horizontal
c                          velocity is estimated.
c
```

```
c-----------------------------------------------------------------
--
c
c – volume grid size (user defined) ----------------------------
--
c
      include 'pstate.h'
      common /maxval6/ veval,vmax,ii,jj
      real veval(nz),vmax(6,nz)
      integer ii(6,nz),jj(6,nz)
c ---------------------------------------------------------------
      real temp(nx,2,nz),salt(nx,2,nz)
      real tempgb(nx,nz),saltgb(nx,nz)
c---------------------------------------------------------------
      integer j,spherical
      real lat0,lon0
      integer ts,ktop,sect,sect1,sect2
      real sims,pref,grid_size
      real radius,omega
      real f0
      real deep(nz)
      integer mx1,my1,stopog(nx,ny)
      real latmin
      integer iuvuni,iunpxy,iunrho,iuvunig,iunq
      integer k,ll
      character *30 topofile,tempfile,saltfile
c
      iuvuni=15
      iunpxy=26
      iunrho=17
      iuvunig=28
      iunq=19

      latmin=5.0
      zero=0.0
      one=1.0
      pi = 3.1415926535
      deg2rad = pi/180
c
c – cntrl parameters
c
      call pvpar(deep,grid_size,lon0,lat0,ktop,pref,ts,spherical,
     &     sims,topofile,tempfile,saltfile)
      my1=my-1
      mx1=mx-1
      if(global) mx1=mx

c – physical and numerical constants
      omega = 7.292e-5
      radius = 6.371e6
```

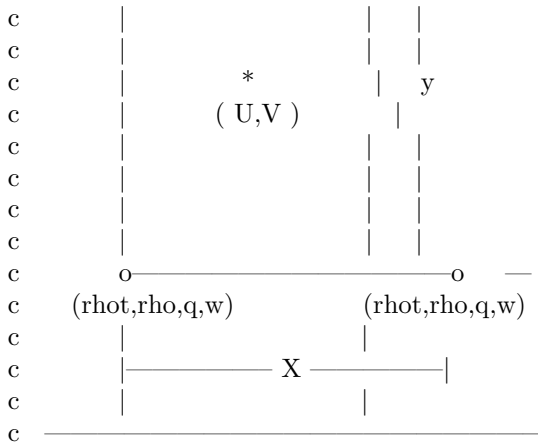
```
c-----------------------------------------------------------------------
c – grid spacing along latitude
c
      dy = grid_size*deg2rad*radius

      do j=1,my
         lat(j) = lat0 - (j-1)*grid_size
      enddo
c
c – grid spacing along longitude
c
      if (spherical.eq.0) then
       do j= 1, my
         dx(j) = dy
       enddo
      else

       do j= 1, my
         dx(j) = dy*cos(lat(j)*deg2rad)
       enddo
      endif
c
c – Coriolis (1/s)
c
      do j=1,my
       ff(j) = 2.*omega*sin(lat(j)*deg2rad)
       if(lat(j).lt.latmin .and. lat(j).ge.0.0)
     &    ff(j)=2.*omega*sin(latmin*deg2rad)
       if(lat(j).gt.-latmin .and. lat(j).le.0.0)
     &    ff(j)=-2.*omega*sin(latmin*deg2rad)
       enddo
       f0=0.5*(abs(ff(1))+abs(ff(my)))

       call read_topo(stopog,deep,ts,topofile,tempfile)

       call setmask(mx,my,istopog,rmask,invmask,global)

       do k=ktop,mz
        do ll=1,6
           vmax(i,k)=0.0
        enddo
        veval(k)=0.0
       enddo
       if(ts) then
        open(8,file=tempfile)
        open(9,file=saltfile)
       endif
```

```
      call matopen(iuvuni,iunpxy,iunrho,iuvunig,iunq)
      sect1=2
      sect2=1
      call userinp(stopog,deep,temp,salt,pref,ts,sect2,0)
      call pvector(stopog,deep,ktop,sims,sect1,sect2,f0,0)

      if(global) then
       do i=1,mx
          do k=1,mz
             tempgb(i,k)=temp(i,1,k)
             saltgb(i,k)=salt(i,1,k)
          enddo
       enddo
      endif

      do j=1,my1
       sect=sect2
       sect2=sect1
       sect1=sect
       call userinp(stopog,deep,temp,salt,pref,ts,sect2,j)
       call pvector(stopog,deep,ktop,sims,sect1,sect2,f0,j)
c
c – output
c
c
c – this output is designed for simple matlab inputs
c
        call matout(iuvuni,iunpxy,iunrho,iuvunig,iunq,
      &          temp,salt,sect2,f0)

      enddo
      close(8)
      if(ts) close(9)
      call matclose(iuvuni,iunpxy,iunrho,iuvunig,iunq)
      do k=ktop,mz
       veval(k)=veval(k)/(mx1*my1)
       print 77,k,-deep(k),veval(k)*100.0,(vmax(i,k)*100.0,i=1,6)
       print 88,(ii(i,k),jj(i,k),i=1,6)
      enddo
77     format(/i4,f8.0,7f8.2)
88     format(20x,6(2x,2i3))
      call matotal(mx1,my1,mz,istopog,rmask,invmask,deep,
      &           grid_size,lon0,lat0,ktop,pref,global)
      stop
      end
```

A.3.2 NetCDF Format (pvnc.f)

This FORTRAN program is used for the global coordinate system with the latitude as the x-axis and the longitude as the y-axis. The data format is in NetCDF.

```
**********************************************************
***
      program pvnc
c-------------------------------------------------------------
c
c-------------------------------------------------------------
c
c — tempfile: temperature data file name (binary data file)
c — saltfile: salitnity data file name (binary data file)
c — topofile: topography data file name
c — denfile: density data file name
c
c    The Grid desired as:
c
c       (rhot,rho,q)              (rhot,rho,q)
c            o-----------*-------------o      —
c            |         ( U,V )        |     |
c            |                        |     |
c            |                        |     |
c            |                        |     |
c            |                        |
c            *(w)                    (w)*      Z
c            |                        |
c            |                        |     |
c            |                        |     |
c            |                        |     |
c            |         ( U,V )        |     |
c            o-----------*-------------o      —
c       (rhot,rho,q)              (rhot,rho,q)
c            |                        |
c            |------------ X , Y -----------|
c            |                        |
c
c
c
c       (rhot,rho,q,w)                (rhot,rho,q,w)
c            o-----------------------------o    —
c            |                           |    |
c            |                           |    |
c            |                           |    |
```




```
c
c    Notes:
c    1. Setup.
c      For a given application the program is configured
c    internally by defining the parameters from the input data.
c
c    mx,my,mz (the dimension for rho,rhot,q,temp,salt)
c    mx1=mx-1,my1=my-1,mz1=mz-1 (the dimension for u,v,w)
c
c    mx is the number of longitudinal points.
c    my is the number of latitudinal points.
c    mz is the number of depth points.
c
c    The following data should be fill in by the user in the
c    subroutine pvpar
c
c    The point (1,1,1) is at the surface in the North West
c    edge of the grid.
c
c    spherical            [(0) dx=dy, (1) dx(j)=dy*cos(lat(j)]
c    global               [(1) global, (0) not global]
c
c    grid_size (in degrees) [The grid size spacing is uniform
c                  along longitude and latitude]
c
c    deep (1:mz)          [ The depth of the levels (m).
c                  The origin is at the ocean surface
c                  and z is negative downwards.]
c lat0               [ latitude (degrees) at NW corner. ]
c
c lon0               [ longitude (degrees) at NW corner]
c
```

```
c
c     ktop            [In the inversion procedure use
c                    only the levels [ktop:mz]]
c
c     ts              read either TS or RHO RHOT
c           1         read temp & salt from ascii data files
c           2         read temp & salt from binary data files
c
c

c      2. Inputs:
c
c      stopog(mx,my)       read in subroutine read_topo
c                          see description in subroutine header.
c
c      if(ts=1):
c      temp(mx,mz,my)       temperature
c      salt(mx,mz,my)       salinity
c                            read in subroutine read_ts
c      if(ts=0):
c      rho(mx,mz,my)       density
c      rhot(mx,mz,my)      potential density
c
c       3. Outputs: one NetCDF data format file
c
c-------------------------------------------------------------------
c
c      The binary output data file is named as pvoutbin.dat.
c      The structure of this file is given by
c      mx1,my1,mz,grid_size,lat0,lon0,ktop,pref,global
c      deep                                    (mz)
c      rho (j=my)                              (mx*mz)
c      rhot (j=my)                             (mx*mz)
c      stopog (j=my)                             (mx)
c      istopog (j=my)                            (mx)
c     *********
c     (the follow is my1 crosesection from j=my1 to 1, each section as:)
c      rmask                                    (mx1)
c      invmask                                   (mx1)
c      px                                      (mx1*mz)
c      py                                      (mx1*mz)
c      pz                                      (mx1*mz)
c      ug                                      (mx1*mz)
c      vg                                      (mx1*mz)
c      u                                      (mx1*mz)
```

```
c    v                                  (mx1*mz)
c    w                                  (mx1*mz)
c    rho                                (mx*mz)
c    rhot                               (mx*mz)
c    stopog                               (mx)
c    istopog                              (mx)
c   *********
c
c------- volume grid size (user defined) ------------------------
      include 'pstate.h'
      include 'netcdf.inc'

      common /maxval6/ veval,vmax,ii,jj
      real veval(nz),vmax(6,nz)
      integer ii(6,nz),jj(6,nz)
c-------------------------------------------------------
      real temp(nx,nz),salt(nx,nz)
c-------------------------------------------------------
      real deg2rad,pi,one
c-------------------------------------------------------
      integer j,spherical,mx1,my1
c     integer my_handler
      integer ts,ktop,sect,sect1,sect2
      real sims,pref,grid_size,lat0,lon0
      real radius,omega,f0
      real deep(nz), r1d(nx+ny)
      integer stopog(nx,ny), istw(3), ictw(3),
     &     dim2d(2), dim3d(3), ist2(2), ist3(3), ict2(2), ict3(3),
     &     dimr3d(3), ist3r(3), ict3r(3)
      integer*2 i1d(nx*(ny+nz))
      real latmin
      character*30 topofile,tempfile,saltfile
      character titl*80, author*30, created*9, ncflnm*30

      real*8 facu, facut, facw, fact, ofstst, ofstr

      data facu, facut, facw/1.0d-4,2.0d-2,1.0d-6/
      data fact, ofstst, ofstr/1.0d-3, 1.5d1, 1.28d3/

      data ist2,ist3,istw,ict2,ict3,ictw/16*1/
      data ist3r, ict3r/6*1/
c
      rlim=32760.0
      latmin=5.0
      zero=0.0
```

```
      one=1.0
      pi = 3.1415926535
      deg2rad = pi/180
c
c – cntrl parameters
c
      call pvpar(deep,grid_size,lon0,lat0,ktop,pref,ts,spherical,
     &     sims,topofile,tempfile,saltfile,titl,author,ncflnm)
      my1=my-1
      mx1=mx-1
      if(global) mx1=mx

c – physical and numerical constants
      omega = 7.292e-5
      radius = 6.371e6

c-----------------------------------------------------------------
c – gridspacing along latitude
c
      dy = grid_size*deg2rad*radius

      do j=1,my
        lat(j) = lat0 - (j-1)*grid_size
      enddo
c
c – grid spacing (m) along longitude
c
      if (spherical.eq.0) then
        do j= 1, my
          dx(j) = dy
        enddo
      else

        do j= 1, my
          dx(j) = dy*cos(lat(j)*deg2rad)
        enddo
      endif
c
c – Coriolis (1/s)
c
      do j=1,my
        ff(j) = 2.*omega*sin(lat(j)*deg2rad)
        if(lat(j).lt.latmin .and. lat(j).ge.0.0)
```

```
     &    ff(j)=2.*omega*sin(latmin*deg2rad)
          if(lat(j).gt.-latmin .and. lat(j).lt.0.0)
     &    ff(j)=-2.*omega*sin(latmin*deg2rad)
         enddo
         f0=0.5*(abs(ff(1))+abs(ff(my)))

         call read_topo(stopog,deep,ts,topofile,tempfile,saltfile)

         call setmask(mx,my,istopog,rmask,invmask,global)

         do k=ktop,mz
          do ll=1,6
           vmax(i,k)=0.0
          enddo
          veval(k)=0.0
         enddo

         call date(created)


         IRET = NF_CREATE(ncflnm,NF_CLOBBER,NCID)
         CALL ERRCHECK(IRET)

         IRET= F_PUT_ATT_TEXT(NCID,NF_GLOBAL,'title',80,titl)
         CALL ERRCHECK(IRET)
         IRET = NF_PUT_ATT_TEXT(NCID,NF_GLOBAL,'author',39,
        &          author//created)
         CALL ERRCHECK(IRET)
c
c     defind dimensions (lat, lon, lev)
c
         IRET = NF_DEF_DIM(NCID,'lat',my,IDJDIM)
         CALL ERRCHECK(IRET)
         IRET = NF_DEF_DIM(NCID,'latuv',my1,IDJUVDIM)
         CALL ERRCHECK(IRET)
         IRET = NF_DEF_DIM(NCID,'latw',my1-1,IDJWDIM)
         CALL ERRCHECK(IRET)
         IRET = NF_DEF_DIM(NCID,'lon',mx,IDIDIM)
         CALL ERRCHECK(IRET)
         IRET = NF_DEF_DIM(NCID,'lonuv',mx1,IDIUVDIM)
         CALL ERRCHECK(IRET)
         IRET = NF_DEF_DIM(NCID,'lonw',mx1-1,IDIWDIM)
         CALL ERRCHECK(IRET)
         IRET = NF_DEF_DIM(NCID,'lev',mz,IDKDIM)
         CALL ERRCHECK(IRET)
```

```
c
c     defined coordinate data (lat, lon, deep)
c
          IRET = NF_DEF_VAR(NCID,'lat',NF_float,1,
     &     IDJDIM,IDYCOORD)
          CALL ERRCHECK(IRET)
          IRET = NF_DEF_VAR(NCID,'latuv',NF_float,1,
     & IDJUVDIM,IDYUVCOORD)
          CALL ERRCHECK(IRET)
          IRET = NF_DEF_VAR(NCID,'latw',NF_float,1,IDJWDIM,
     & IDYWCOORD)
          CALL ERRCHECK(IRET)
          IRET = NF_DEF_VAR(NCID,'lon',NF_float,1,
     &IDIDIM,IDXCOORD)
          CALL ERRCHECK(IRET)
          IRET = NF_DEF_VAR(NCID,'lonuv',NF_float,1,
     &           IDIUVDIM,IDXUVCOORD)
          CALL ERRCHECK(IRET)
          IRET = NF_DEF_VAR(NCID,'lonw',NF_float,1,
     &           IDIWDIM,IDXWCOORD)
          CALL ERRCHECK(IRET)
          IRET = NF_DEF_VAR(NCID,'depth',NF_float,1,
     &           IDKDIM,IDZCOORD)
          CALL ERRCHECK(IRET)
          IRET = NF_PUT_ATT_TEXT(NCID,
     &           IDZCOORD,'units',6,'meters')
          CALL ERRCHECK(IRET)
          IRET = NF_PUT_ATT_TEXT(NCID,
     &            IDYCOORD,'long_name',28,
     &                  'Latitudinal Position for rho')
          CALL ERRCHECK(IRET)
          IRET = NF_PUT_ATT_TEXT(NCID,IDYUVCOORD,
     &'long_name',28,'Latitudinal Position for u,v')
          CALL ERRCHECK(IRET)
          IRET = NF_PUT_ATT_TEXT(NCID,IDYWCOORD,
     &'long_name',26, 'Latitudinal Position for w')
          CALL ERRCHECK(IRET)
          IRET = NF_PUT_ATT_TEXT(NCID,IDXCOORD,
     &'long_name',29, 'Longitudinal Position for rho')
          CALL ERRCHECK(IRET)
          IRET = NF_PUT_ATT_TEXT(NCID,IDXUVCOORD,
     &'long_name',29, 'Longitudinal Position for u,v')
          CALL ERRCHECK(IRET)
          IRET = NF_PUT_ATT_TEXT(NCID,IDXWCOORD,
     &'long_name',27, 'Longitudinal Position for w')
          CALL ERRCHECK(IRET)
          IRET = NF_PUT_ATT_TEXT(NCID,IDZCOORD,
     &'long_name',27,'Vertical Position for u,v,w')
          CALL ERRCHECK(IRET)
```

```
  dimr3d(1)=IDKDIM
  dimr3d(2)=IDIDIM
  dimr3d(3)=IDJDIM

  dim3d(1)=IDKDIM
  dim3d(2)=IDIUVDIM
  dim3d(3)=IDJUVDIM

  dim2d(1)=IDIUVDIM
  dim2d(2)=IDJUVDIM

  IRET = NF_DEF_VAR(NCID,'mask',NF_SHORT,2,
&          dim2d,IDMASK)
  CALL ERRCHECK(IRET)
  IRET = NF_DEF_VAR(NCID,'maskinv',NF_SHORT,2,
&          dim2d,IDMASKINV)
  CALL ERRCHECK(IRET)

  IRET = NF_DEF_VAR(NCID,'Temperature',NF_SHORT,3,
&          dimr3d,IDT)
  CALL ERRCHECK(IRET)
  IRET = NF_DEF_VAR(NCID,'Salinity',NF_SHORT,3,
&          dimr3d,IDS)
  CALL ERRCHECK(IRET)
  IRET = NF_DEF_VAR(NCID,'Density',NF_SHORT,3,
&          dimr3d,IDR)
  CALL ERRCHECK(IRET)
  IRET = NF_DEF_VAR(NCID,'Potential_Density',NF_SHORT,3,
&          dimr3d,IDRT)
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDT,'scale_factor',
&          NF_double,1,fact)
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDS,'scale_factor',
&          NF_double,1,fact)
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDR,'scale_factor',
&          NF_double,1,fact)
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDRT,'scale_factor',
&          NF_double,1,fact)
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDT,'add_offset',
```

```
     &           NF_double,1,ofstst)
        CALL ERRCHECK(IRET)
        IRET = NF_PUT_ATT_DOUBLE(NCID,IDS,'add_offset',
     &           NF_double,1,ofstst)
        CALL ERRCHECK(IRET)
        IRET = NF_PUT_ATT_DOUBLE(NCID,IDR,'add_offset',
     &           NF_double,1,ofstr)
        CALL ERRCHECK(IRET)
        IRET = NF_PUT_ATT_DOUBLE(NCID,IDRT,'add_offset',
     &           NF_double,1,ofstr)
        CALL ERRCHECK(IRET)

        IRET = NF_DEF_VAR(NCID,'u',NF_SHORT,3,dim3d,IDU)
        CALL ERRCHECK(IRET)
        IRET = NF_DEF_VAR(NCID,'v',NF_SHORT,3,dim3d,IDV)
        CALL ERRCHECK(IRET)
        dim3d(2)=IDIWDIM
        dim3d(3)=IDJWDIM
        IRET = NF_DEF_VAR(NCID,'w',NF_SHORT,3,dim3d,IDW)
        CALL ERRCHECK(IRET)

        dim2d(1)=IDIUVDIM
        dim2d(2)=IDJUVDIM
        IRET = NF_DEF_VAR(NCID,'utl',NF_SHORT,2,
     &            dim2d,IDUTL)
        CALL ERRCHECK(IRET)
        IRET = NF_DEF_VAR(NCID,'vtl',NF_SHORT,2,dim2d,
     &            IDVTL)
        CALL ERRCHECK(IRET)

        IRET = NF_PUT_ATT_TEXT(NCID,IDU,'long_name',14,
     &                  '3-D u-velosity')
        CALL ERRCHECK(IRET)
        IRET = NF_PUT_ATT_TEXT(NCID,IDU,'units',3,'m/s')
        CALL ERRCHECK(IRET)
        IRET = NF_PUT_ATT_DOUBLE(NCID,IDU,'scale_factor',
     &         NF_double,1,facu)
        CALL ERRCHECK(IRET)
        IRET = NF_PUT_ATT_TEXT(NCID,IDV,'long_name',14,
     &         '3-D v-velosity')
        CALL ERRCHECK(IRET)
```

```
  IRET = NF_PUT_ATT_TEXT(NCID,IDV,'units',3,'m/s')
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDV,'scale_factor',
&                    NF_double,1,facu)
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_TEXT(NCID,IDW,'long_name',14,
&                    '3-D w-velosity')
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_TEXT(NCID,IDW,'units',3,'m/s')
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDW,'scale_factor',
&                    NF_double,1,facw)
  CALL ERRCHECK(IRET)

  IRET = NF_PUT_ATT_TEXT(NCID,IDUTL,'long_name',32,
&                    'Vertically Integrated u-velosity')
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_TEXT(NCID,IDUTL,'units',6,'m**2/s')
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDUTL,'scale_factor',
&                    NF_double,1,facut)
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_TEXT(NCID,IDVTL,'long_name',32,
&                    'Vertically Integrated v-velosity')
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_TEXT(NCID,IDVTL,'units',6,'m**2/s')
  CALL ERRCHECK(IRET)
  IRET = NF_PUT_ATT_DOUBLE(NCID,IDVTL,'scale_factor',
&                    NF_double,1,facut)
  CALL ERRCHECK(IRET)

  IRET = NF_ENDDEF(NCID)
  CALL ERRCHECK(IRET)

c
c     write coordinates
c
  r1d(1)=lon0
  do i=2,mx
     r1d(i)=r1d(i-1)+grid_size
  enddo
```

```
IRET = NF_PUT_VAR_real(NCID,IDXCOORD,r1d)
CALL ERRCHECK(IRET)
r1d(1)=lon0+0.5*grid_size
do i=2,mx1
 r1d(i)=r1d(i-1)+grid_size
enddo
IRET = NF_PUT_VAR_real(NCID,IDXUVCOORD,r1d)
CALL ERRCHECK(IRET)
do i=1,mx1-1
 r1d(i)=(r1d(i)+r1d(i+1))/2.0
enddo
IRET = NF_PUT_VAR_real(NCID,IDXWCOORD,r1d)
CALL ERRCHECK(IRET)
IRET = NF_PUT_VAR_real(NCID,IDYCOORD,lat)
CALL ERRCHECK(IRET)
do j=1,my1
 r1d(j)=(lat(j)+lat(j+1))/2.0
enddo
IRET = NF_PUT_VAR_real(NCID,IDYUVCOORD,r1d)
CALL ERRCHECK(IRET)
do j=1,my1-1
 r1d(j)=(r1d(j)+r1d(j+1))/2.0
enddo
IRET = NF_PUT_VAR_real(NCID,IDYWCOORD,r1d)
CALL ERRCHECK(IRET)
IRET = NF_PUT_VAR_real(NCID,IDZCOORD,deep)
CALL ERRCHECK(IRET)
do i=1,mx1
  do j=1,my1
    i1d(mx1*(j-1)+i)=rmask(i,j)
  enddo
enddo
IRET = NF_PUT_VAR_int2(NCID,IDMASK,i1d)
CALL ERRCHECK(IRET)

if(ts.ge.2) then
  open(8,file=tempfile,access='direct',recl=mx*mz)
  open(9,file=saltfile,access='direct',recl=mx*mz)
```

```
elseif(ts.ge.1) then
  open(8,file=tempfile)
  open(9,file=saltfile)
endif
sect1=2
sect2=1

call userinp(stopog,deep,temp,salt,pref,ts,sect2,0)
call pvector(stopog,deep,ktop,sims,sect1,sect2,f0,0)
ist3r(3)=1
ict3r(1)=mz
ict3r(2)=mx

do i=1,mx
  do k=1,mz
    uuu=temp(i,k)/fact+0.5-ofstst
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
IRET = NF_PUT_VARA_int2(NCID,IDT,ist3r,ict3r,i1d)
CALL ERRCHECK(IRET)
do i=1,mx
  do k=1,mz
    uuu=salt(i,k)/fact+0.5-ofstst
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
IRET = NF_PUT_VARA_int2(NCID,IDS,ist3r,ict3r,i1d)
CALL ERRCHECK(IRET)
do i=1,mx
  do k=1,mz
    uuu=rho(i,sect2,k)/factr+0.5-ofstr
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
```

```
                IRET = NF_PUT_VARA_int2(NCID,IDR,ist3r,ict3r,i1d)
                CALL ERRCHECK(IRET)
                do i=1,mx
                   do k=1,mz
                     uuu=rhot(i,sect2,k)/factr+0.5-ofstr
                     uuu=max(-rlim,uuu)
                     uuu=min(rlim,uuu)
                     i1d(mz*(i-1)+k)=uuu
                   enddo
                enddo
                IRET = NF_PUT_VARA_int2(NCID,IDRT,ist3r,ict3r,i1d)
                CALL ERRCHECK(IRET)

                ict2(1)=mx1
                ict2(2)=1
                ict3(1)=mz
                ict3(2)=mx1
                ict3(3)=1
                ictw(1)=mz
                ictw(2)=mx1-1
                ictw(3)=1
                do j=1,my1
c                  print *,' j=',j
                   sect=sect2
                   sect2=sect1
                   sect1=sect
                   call userinp(stopog,deep,temp,salt,pref,ts,sect2,j)
                   call pvector(stopog,deep,ktop,sims,sect1,sect2,f0,j)
c
c - output
c
                   ist2(2)=j
                   ist3(3)=j
                   ist3r(3)=j+1
                   istw(3)=j-1
                   do i=1,mx
                      do k=1,mz
                       uuu=temp(i,k)/fact+0.5-ofstst
                       uuu=max(-rlim,uuu)
                       uuu=min(rlim,uuu)
                       i1d(mz*(i-1)+k)=uuu
                      enddo
                   enddo
```

```
IRET = NF_PUT_VARA_int2(NCID,IDT,ist3r,ict3r,i1d)
CALL ERRCHECK(IRET)
do i=1,mx
  do k=1,mz
    uuu=salt(i,k)/fact+0.5-ofstst
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
IRET = NF_PUT_VARA_int2(NCID,IDS,ist3r,ict3r,i1d)
CALL ERRCHECK(IRET)
do i=1,mx
  do k=1,mz
    uuu=rho(i,sect2,k)/factr+0.5-ofstr
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
IRET = NF_PUT_VARA_int2(NCID,IDR,ist3r,ict3r,i1d)
CALL ERRCHECK(IRET)
do i=1,mx
  do k=1,mz
    uuu=rhot(i,sect2,k)/factr+0.5-ofstr
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
IRET = NF_PUT_VARA_int2(NCID,IDRT,ist3r,ict3r,i1d)
CALL ERRCHECK(IRET)

do i=1,mx1
  i1d(i)=invmask(i,j)
enddo

IRET = NF_PUT_VARA_int2(NCID,IDMASKINV,ist2,
&         ict2,i1d)
CALL ERRCHECK(IRET)
```

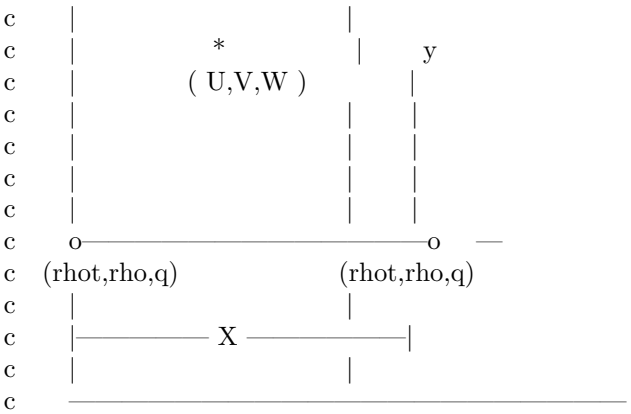
```
do i=1,mx1
  do k=1,mz
    uuu=u(i,sect2,k)/facu+0.5
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
IRET = NF_PUT_VARA_int2(NCID,IDU,ist3,ict3,i1d)
CALL ERRCHECK(IRET)
do i=1,mx1
  do k=1,mz
    uuu=v(i,sect2,k)/facu+0.5
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
IRET = NF_PUT_VARA_int2(NCID,IDV,ist3,ict3,i1d)
CALL ERRCHECK(IRET)
if(j.gt.1) then
do i=1,mx1-1
  do k=1,mz
    uuu=w(i,k)/facw+0.5
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(mz*(i-1)+k)=uuu
  enddo
enddo
IRET = NF_PUT_VARA_int2(NCID,IDW,istw,ictw,i1d)
CALL ERRCHECK(IRET)
endif

do i=1,mx1
 utt=0.0
 do k=2,invmask(i,j)
    utt=utt+(deep(k-1)-deep(k))*
```

```
&                    (u(i,sect2,k-1)+u(i,sect2,k))/2.0
        enddo
        utt=utt/facut+0.5
        utt=max(-rlim,utt)
        utt=min(rlim,utt)
        i1d(i)=utt
      enddo
      IRET = NF_PUT_VARA_int2(NCID,IDUTL,ist2,ict2,i1d)
      CALL ERRCHECK(IRET)
      do i=1,mx1
       utt=0.0
       do k=2,invmask(i,j)
        utt=utt+(deep(k-1)-deep(k))*
&                    (v(i,sect2,k-1)+v(i,sect2,k))/2.0
       enddo
       utt=utt/facut+0.5
       utt=max(-rlim,utt)
       utt=min(rlim,utt)
       i1d(i)=utt
      enddo
         IRET = NF_PUT_VARA_int2(NCID,IDVTL,ist2,ict2,i1d)
         CALL ERRCHECK(IRET)

      enddo
      close(8)
      close(9)

      IRET = NF_CLOSE(NCID)

      do k=ktop,mz
         veval(k)=veval(k)/(mx*my)
         print 77,k,-deep(k),veval(k)*100.0,(vmax(i,k)*100.0,i=1,6)
         print 88,(ii(i,k),jj(i,k),i=1,6)
      enddo
77       format(/i4,f8.0,7f8.2)
88       format(20x,6(2x,2i3))


      end
```

```
C=======================================
C
      SUBROUTINE ERRCHECK(IRET)
C*******************************************************
C
C       *ERRCHECK* ERROR HANDLER FOR GRAPHICAL
C       INTERFACE
C
C     AUTHOR - ROGER PROCTOR AND PATRICK LUYTEN
C
C     LAST UPDATE - 12 Jul 1999 @(COHERENS)netcdfint.f
C       8.4
C
C       DESCRIPTION - WRITE ERROR MESSAGE AND STOP
C       EXECUTION OF THE PROGRAM
C         IF AN ERROR OCCURS IN ONE OF THE netCDF
       ROUTINES
C
C      REFERENCE -
C
C      CALLING PROGRAM - CDF2D, CDF3D
C
C      EXTERNALS - NF_STRERROR
C
C*******************************************************
C
    INCLUDE 'netcdf.inc'

C
C   ARGUMENTS
C
    INTEGER IRET

    IF (IRET.NE.NF_NOERR) THEN
      WRITE (0,*) NF_STRERROR(IRET)
      STOP
    ENDIF

    RETURN
    END
```

A.3.3 Non-(Lat, Lon) Horizontal Coordinates (pvxync.f)

```
**********************************************************
**
 program pvxync
c-------------
c    This program is used for any rectangular horizontal coordinates.
c
c-------------------------------------------------------------
--
c      -- tempfile: name of temperature data file (binary data file)
c      -- saltfile: name of salitnity data file (binary data file)
c      -- ltlnfile: 2-D latitude & longitude data file name (binary)
c                              lon(mx,my), lat(mxmy), lonm(mx1,my1),
 latm(mx1,my1)
c
c    The Grid desired as:
c
c    (rhot,rho,q)                    (rhot,rho,q)
c       o-------------*---------------o      --
c       |          ( U,V,W )          |      |
c       |                        |    |
c       |                        |    |
c       |                        |
c       |                        |    Z
c       |                        |
c       |                        |    |
c       |                        |    |
c       |          ( U,V,W )          |      |
c       o-------------*---------------o      --
c    (rhot,rho,q)                  (rhot,rho,q)
c       |                           |
c       |------------ X, Y ---------------|
c       |                           |
c
c
c
c    (rhot,rho,q)                    (rhot,rho,q)
c       o-------------------------------o    --
c       |                           |      |
c       |                           |      |
c       |                           |      |
c       |                           |      |
```




```
c   Notes:
c   1. Setup.
c     For a given application the program is configured
c     internally by defining the parameters from the input data.
c
c     mx,my,mz (the dimension for rho,rhot,q,temp,salt)
c     mx1=mx-1,my1=my-1,mz1=mz-1 (the dimension for u,v)
c
c     mx is the number of longitudinal points.
c     my is the number of latitudinal points.
c     mz is the number of depth points.
c
c     The following data should be filled by the user for the
c     subroutine pvpar
c
c     The point (1,1,1) is at the surface in the North West
c     edge of the grid.
c
c     deep (1:mz)             [ The depth of the levels (m).
c                           The origin is at the ocean surface
c                           and z is negative downwards.]
c     lon&lat data file name    [ 2-D binary data (degrees)]
c
c
c     ktop                  [In the inversion procedure use
c                           only the levels [ktop:mz]]
c
c     ts              read either TS or RHO RHOT
c          1          ascii data with order (mx,mz,my)
c         -1          ascii data with order (mz,mx,my)
c          2          binary data with order (mx,mz,my)
c         -2          binary data with order (mz,mx,my)
c
```

```
c     2. Inputs:
c
c       lat(mx,my), lon(mx,my) 2-D data
c
c      if(ts>0):
c       temp(mx,mz,my)     temperature
c       salt(mx,mz,my)     salinity
c      if(ts<0):
c       temp(mz,mx,my)     temperature
c       salt(mz,mx,my)     salinity
c
c     3. Outputs:  one Netcdf data format file
c
c-----------------------------------------------------------------
c
c---- volume grid size (user defined) ---------------------------
      include 'pstate.h'
      include 'netcdf.inc'

      common /maxval6/ veval,vmax,ii,jj
      real veval(nz),vmax(6,nz)
      integer ii(6,nz),jj(6,nz)
c-----------------------------------------------------------------
      real temp(nx,2,nz),salt(nx,2,nz)
c-----------------------------------------------------------------
      real deg2rad,pi,one
c-----------------------------------------------------------------
      integer j,mx1,my1
      integer ts,ktop,sect,sect1,sect2
      real sims,pref
      real radius,omega,f0
      real deep(nz), r1d(nx*ny)
      integer ist3t(3), ict3t(3),
     &     dim2d(2), dim3d(3), ist2(2), ist3(3), ict2(2), ict3(3)
      integer*2 i1d(nx*(ny+nz))
      real latmin
      character*30 tempfile,saltfile,ltlnfile,pvoutfile
      character titl*80, author*30, created*9

      real*8 facu, facw, ofstu,
     &     facr, ofstr, fact, ofstt, facs, ofsts

      data facu, facw, ofstu/1.0d-4,1.0d-7,0.0d0/
      data facr, ofstr/1.0d-3, 28.0d0/
      data fact, ofstt/1.0d-3, 10.0d0/
```

```
      data facs, ofsts/1.0d-3, 20.0d0/

      data ist2,ist3,ist3t,ict2,ict3,ict3t/16*1/
c
      latmin=5.0
      zero=0.0
      one=1.0
      pi = 3.1415926535
      deg2rad = pi/180
c
c — cntrl parameters
c
      call pvpar(deep,ktop,pref,ts,sims,tempfile,saltfile,
     &          ltlnfile,pvoutfile,titl,author)
      my1=my-1
      mx1=mx-1

c — physical and numerical constants
      omega = 7.292e-5
      radius = 6.371e6

      deg2m=deg2rad*radius

c
c ——     read lat lon binary data
c
       if(abs(ts).eq.1) then
        open(18,file=ltlnfile)
         read(18,*) ((lat(i,j),i=1,mx),j=1,my)
         read(18,*) ((lon(i,j),i=1,mx),j=1,my)
         read(18,*) ((y(i,j),i=1,mx),j=1,my)
         read(18,*) ((x(i,j),i=1,mx),j=1,my)
        close(18)
       else
        open(18,file=ltlnfile,access='direct',recl=mx*my)
         read(18,rec=1) ((lat(i,j),i=1,mx),j=1,my)
         read(18,rec=2) ((lon(i,j),i=1,mx),j=1,my)
         read(18,rec=3) ((y(i,j),i=1,mx),j=1,my)
         read(18,rec=4) ((x(i,j),i=1,mx),j=1,my)
        close(18)
       endif
```

```
c
c     convert units from km to m
c
      do i=1,mx
       do j=1,my
        x(i,j)=x(i,j)*1000.0
        y(i,j)=y(i,j)*1000.0
       enddo
      enddo

c-------------------------------------------------------------
c -   gridspacing
c
      do i=1,mx1
       i1=i+1
       do j=1,my1
        j1=j+1
        lonm(i,j)=(lon(i,j)+lon(i1,j)+lon(i,j1)+lon(i1,j1))/4.0
        latm(i,j)=(lat(i,j)+lat(i1,j)+lat(i,j1)+lat(i1,j1))/4.0
        dx(i,j)=(x(i1,j)+x(i1,j1)-x(i,j)-x(i,j1))/2.0
        dy(i,j)=(y(i,j1)+y(i1,j1)-y(i,j)-y(i1,j))/2.0
       enddo
      enddo

 66    continue
c
c - Coriolis (1/s) at the corner of the cell
c
     do i=1,mx
       do j=1,my
         if(lat(i,j).lt.latmin .and. lat(i,j).ge.0.0) then
          ff(i,j)=2.*omega*sin(latmin*deg2rad)
         elseif(lat(i,j).gt.-latmin .and. lat(i,j).lt.0.0) then
          ff(i,j)=-2.*omega*sin(latmin*deg2rad)
         else
          ff(i,j)=2.*omega*sin(lat(i,j)*deg2rad)
         endif
       enddo
     enddo
c
c - Coriolis (1/s) at the center of the cell
c
```

```
      fmax=0.0
      fmin=100.0
      do i=1,mx1
       do j=1,my1
        if(latm(i,j).lt.latmin .and. latm(i,j).ge.0.0) then
         fm(i,j)=2.*omega*sin(latmin*deg2rad)
        elseif(latm(i,j).gt.-latmin .and. latm(i,j).lt.0.0) then
         fm(i,j)=-2.*omega*sin(latmin*deg2rad)
        else
         fm(i,j) = 2.*omega*sin(latm(i,j)*deg2rad)
        endif
         fmax=max(fmax,abs(fm(i,j)))
         fmin=min(fmin,abs(fm(i,j)))
        enddo
      enddo
      f0=0.5*(fmin,fmax)

      call read_topo(ts,tempfile,saltfile)

      call setmask(mx,my,istopog,rmask,invmask,global)

      do k=ktop,mz
       do ll=1,6
          vmax(i,k)=0.0
       enddo
       veval(k)=0.0
      enddo

      call date(created)

     print *,created
     IRET = NF_CREATE(pvoutfile,NF_CLOBBER,NCID)
     CALL ERRCHECK(IRET)
     IRET = F_PUT_ATT_TEXT(NCID,NF_GLOBAL,'title',80,titl)
     CALL ERRCHECK(IRET)
     IRET = NF_PUT_ATT_TEXT(NCID,NF_GLOBAL,'author',39,
    &                author//created)
     CALL ERRCHECK(IRET)


    c
    c     defind dimensions (lat, lon, lev)
    c
```

```
      IRET = NF_DEF_DIM(NCID,'lat',my,IDJTDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'lon',mx,IDITDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'latuvw',my1,IDJDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'lonuvw',mx1,IDIDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'lev',mz,IDKDIM)
      CALL ERRCHECK(IRET)
c
c     defined coordinate data (deep)
c


     IRET = NF_DEF_VAR(NCID,'depth',NF_float,1,
     &         IDKDIM,IDZCOORD)
     CALL ERRCHECK(IRET)
     IRET = NF_PUT_ATT_TEXT(NCID,IDZCOORD,
     &         'units',6,'meters')
     CALL ERRCHECK(IRET)
c
c     defined lon & lat as 2-D variables
c
      dim2d(1)=IDITDIM
      dim2d(2)=IDJTDIM
      IRET = NF_DEF_VAR(NCID,'lat',NF_float,2,dim2d,IDTLAT)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'lon',NF_float,2,dim2d,IDTLON)
      CALL ERRCHECK(IRET)
      dim2d(1)=IDIDIM
      dim2d(2)=IDJDIM
      IRET = NF_DEF_VAR(NCID,'latuvw',NF_float,2,
     &         dim2d,IDLAT)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'lonuvw',NF_float,2,
     &         dim2d,IDLON)
      CALL ERRCHECK(IRET)
c
c      defined 2-D & 3-D variables
c
```

```
   dim2d(1)=IDITDIM
   dim2d(2)=IDJTDIM
   IRET = NF_DEF_VAR(NCID,'itopo',NF_SHORT,2,
  &          dim2d,IDITOPO)
   CALL ERRCHECK(IRET)
   dim3d(1)=IDKDIM
   dim3d(2)=IDITDIM
   dim3d(3)=IDJTDIM
   IRET = NF_DEF_VAR(NCID,'temp',NF_SHORT,3,
  &        dim3d,IDTEMP)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_TEXT(NCID,IDTEMP,'units',8,
  &         'degree C')
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDTEMP,'scale_factor',
  &           NF_double,1,fact)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDTEMP,'add_offset',
  &            NF_double,1,ofstt)
   CALL ERRCHECK(IRET)
   IRET = NF_DEF_VAR(NCID,'salt',NF_SHORT,3,
  &           dim3d,IDSALT)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_TEXT(NCID,IDSALT,'units',3,'psu')
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDSALT,'scale_factor',
  &            NF_double,1,facs)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDSALT,'add_offset',
  &            NF_double,1,ofsts)
   CALL ERRCHECK(IRET)
   IRET = NF_DEF_VAR(NCID,'rhot',NF_SHORT,3,dim3d,
  &            IDRHOT)
   CALL ERRCHECK(IRET)
   IRET = F_PUT_ATT_TEXT(NCID,IDRHOT,'units',6,'kg/mˆ3')
   CALL ERRCHECK(IRET)
   IRET = F_PUT_ATT_DOUBLE(NCID,IDRHOT,'scale_factor',
  &            NF_double,1,facr)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDRHOT,'add_offset',
   &                    NF_double,1,ofstr)
   CALL ERRCHECK(IRET)
```

```
      dim3d(2)=IDIDIM
      dim3d(3)=IDJDIM
      dim2d(1)=IDIDIM
      dim2d(2)=IDJDIM
      IRET = NF_DEF_VAR(NCID,'mask',NF_SHORT,2,
     &           dim2d,IDMASK)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'maskinv',NF_SHORT,2,
     &           dim2d,IDMASKINV)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'u',NF_SHORT,3,dim3d,IDU)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'v',NF_SHORT,3,dim3d,IDV)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'w',NF_SHORT,3,dim3d,IDW)
      CALL ERRCHECK(IRET)

      dim2d(1)=IDIDIM
      dim2d(2)=IDJDIM
      IRET = NF_DEF_VAR(NCID,'utl',NF_FLOAT,2,dim2d,IDUTL)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'vtl',NF_FLOAT,2,dim2d,IDVTL)
      CALL ERRCHECK(IRET)

      IRET = NF_PUT_ATT_TEXT(NCID,IDU,'long_name',14,
     &                  '3-D u-velosity')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDU,'units',3,'m/s')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_DOUBLE(NCID,IDU,'scale_factor',
     &                NF_double,1,facu)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDV,'long_name',14,
     &                  '3-D v-velosity')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDV,'units',3,'m/s')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_DOUBLE(NCID,IDV,'scale_factor',
     &                NF_double,1,facu)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDW,'long_name',14,
     &                  '3-D W-velosity')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDW,'units',3,'m/s')
```

```
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_DOUBLE(NCID,IDW,'scale_factor',
     &                NF_double,1,facw)
      CALL ERRCHECK(IRET)

      IRET = NF_PUT_ATT_TEXT(NCID,IDUTL,'long_name',29,
     &                'Vertical Integrate u-velosity')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDUTL,'units',6,'m**2/s')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDVTL,'long_name',29,
     &                'Vertical Integrate v-velosity')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDVTL,'units',6,'m**2/s')
      CALL ERRCHECK(IRET)

      IRET = NF_ENDDEF(NCID)
      CALL ERRCHECK(IRET)

c
c     write lon lat & depth data
c

      do j=1,my
        do i=1,mx
          r1d(i+mx*(j-1))=lon(i,j)
        enddo
      enddo
      IRET = NF_PUT_VAR_real(NCID,IDTLON,r1d)
      CALL ERRCHECK(IRET)

      do j=1,my
        do i=1,mx
          r1d(i+mx*(j-1))=lat(i,j)
        enddo
      enddo
      IRET = NF_PUT_VAR_real(NCID,IDTLAT,r1d)
      CALL ERRCHECK(IRET)
      do j=1,my
        do i=1,mx
          i1d(i+mx*(j-1))=istopog(i,j)
        enddo
      enddo
      IRET = NF_PUT_VAR_int2(NCID,IDITOPO,i1d)
      CALL ERRCHECK(IRET)
```

```
   do j=1,my1
    do i=1,mx1
      tt1=(lon(i,j)+lon(i+1,j))/2.0
      if(abs(tt1-lon(i,j)).gt.100.0) tt1=tt1+180.0
      tt2=(lon(i,j+1)+lon(i+1,j+1))/2.0
      if(abs(tt2-lon(i,j+1)).gt.100.0) tt2=tt2+180.0
      ttt=(tt1+tt2)/2.0
      if(abs(ttt-tt1).gt.100.0) ttt=ttt+180.0
      if(ttt.gt.180) ttt=ttt-360.0
      r1d(i+mx1*(j-1))=ttt
    enddo
   enddo
IRET = NF_PUT_VAR_real(NCID,IDLON,r1d)
CALL ERRCHECK(IRET)
do j=1,my1
 do i=1,mx1
    r1d(i+mx1*(j-1))=(lat(i,j)+lat(i,j+1)
* +lat(i+1,j)+lat(i+1,j+1))/4.0
 enddo
enddo
IRET = NF_PUT_VAR_real(NCID,IDLAT,r1d)
CALL ERRCHECK(IRET)
IRET = NF_PUT_VAR_real(NCID,IDZCOORD,deep)
CALL ERRCHECK(IRET)

do i=1,mx1
 do j=1,my1
    i1d(mx1*(j-1)+i)=rmask(i,j)
 enddo
enddo
IRET = NF_PUT_VAR_int2(NCID,IDMASK,i1d)
CALL ERRCHECK(IRET)
if(abs(ts).eq.2) then
 open(8,file=tempfile,access='direct',recl=mx*mz)
 open(9,file=saltfile,access='direct',recl=mx*mz)
elseif(abs(ts).eq.1) then
 open(8,file=tempfile)
 open(9,file=saltfile)
else
 print *,' Must input temp & salt (|ts|=1 or 2)!'
 stop
endif
sect1=2
sect2=1
```

```
call userinp(deep,temp,salt,pref,ts,sect2,0)
call pvector(deep,ktop,sims,sect1,sect2,f0,0)
ict3t(1)=mz
ict3t(2)=mx
ist3t(3)=1

do i=1,mx
 do k=1,mz
  r1d(mz*(i-1)+k)=temp(i,sect2,k)
 enddo
enddo
call real2int2(mx*mz,r1d,fact,ofstt,i1d)
IRET = NF_PUT_VARA_int2(NCID,IDTEMP,ist3t,ict3t,i1d)
CALL ERRCHECK(IRET)

do i=1,mx
 do k=1,mz
  r1d(mz*(i-1)+k)=salt(i,sect2,k)
 enddo
enddo
call real2int2(mx*mz,r1d,facs,ofsts,i1d)
IRET = NF_PUT_VARA_int2(NCID,IDSALT,ist3t,ict3t,i1d)
CALL ERRCHECK(IRET)

do i=1,mx
 do k=1,mz
  r1d(mz*(i-1)+k)=rhot(i,sect2,k)
 enddo
enddo
call real2int2(mx*mz,r1d,facr,ofstr,i1d)
IRET = NF_PUT_VARA_int2(NCID,IDRHOT,ist3t,ict3t,i1d)
CALL ERRCHECK(IRET)

ict2(1)=mx1
ict2(2)=1
ict3(1)=mz
ict3(2)=mx1
ict3(3)=1
do j=1,my1
c     print *,' j=',j
    sect=sect2
    sect2=sect1
    sect1=sect
    call userinp(deep,temp,salt,pref,ts,sect2,j)
    call pvector(deep,ktop,sims,sect1,sect2,f0,j)
```

```
c
c – output
c
      do i=1,mx1
        i1d(i)=invmask(i,j)
      enddo
      ist2(2)=j
      IRET = NF_PUT_VARA_int2(NCID,IDMASKINV,ist2,
     &       ict2,i1d)
      CALL ERRCHECK(IRET)

      ist3t(3)=j+1
      do i=1,mx
        do k=1,mz
          r1d(mz*(i-1)+k)=temp(i,sect2,k)
        enddo
      enddo
      call real2int2(mx*mz,r1d,fact,ofstt,i1d)
      IRET = NF_PUT_VARA_int2(NCID,IDTEMP,ist3t,ict3t,i1d)
      CALL ERRCHECK(IRET)
      do i=1,mx
        do k=1,mz
          r1d(mz*(i-1)+k)=salt(i,sect2,k)
        enddo
      enddo
      call real2int2(mx*mz,r1d,facs,ofsts,i1d)
      IRET = NF_PUT_VARA_int2(NCID,IDSALT,ist3t,ict3t,i1d)
      CALL ERRCHECK(IRET)

      do i=1,mx
        do k=1,mz
          r1d(mz*(i-1)+k)=rhot(i,sect2,k)
        enddo
      enddo
      call real2int2(mx*mz,r1d,facr,ofstr,i1d)
      IRET = NF_PUT_VARA_int2(NCID,IDRHOT,ist3t,ict3t,i1d)
      CALL ERRCHECK(IRET)
      ist3(3)=j

      do i=1,mx1
        do k=1,mz
          r1d(mz*(i-1)+k)=u(i,k)
        enddo
      enddo
```

```
    call real2int2(mx1*mz,r1d,facu,ofstu,i1d)
    IRET = NF_PUT_VARA_int2(NCID,IDU,ist3,ict3,i1d)
    CALL ERRCHECK(IRET)

    do i=1,mx1
      do k=1,mz
        r1d(mz*(i-1)+k)=v(i,k)
      enddo
    enddo
    call real2int2(mx1*mz,r1d,facu,ofstu,i1d)
    IRET = NF_PUT_VARA_int2(NCID,IDV,ist3,ict3,i1d)
    CALL ERRCHECK(IRET)

    do i=1,mx1
      do k=1,mz
        r1d(mz*(i-1)+k)=w(i,k)
      enddo
    enddo
    call real2int2(mx1*mz,r1d,facw,ofstu,i1d)
    IRET = NF_PUT_VARA_int2(NCID,IDW,ist3,ict3,i1d)
    CALL ERRCHECK(IRET)
    do i=1,mx1
      r1d(i)=0.0
      do k=2,invmask(i,j)
        r1d(i)=r1d(i)+(deep(k-1)-deep(k))*
&           (u(i,k-1)+u(i,k))/2.0
      enddo
    enddo
    IRET = NF_PUT_VARA_real(NCID,IDUTL,ist2,ict2,r1d)
    CALL ERRCHECK(IRET)
    do i=1,mx1
      r1d(i)=0.0
      do k=2,invmask(i,j)
        r1d(i)=r1d(i)+(deep(k-1)-deep(k))*
&           (v(i,k-1)+v(i,k))/2.0
      enddo
    enddo
    IRET = NF_PUT_VARA_real(NCID,IDVTL,ist2,ict2,r1d)
    CALL ERRCHECK(IRET)

    enddo
    close(8)
    close(9)

    IRET = NF_CLOSE(NCID)

    do k=ktop,mz
      veval(k)=veval(k)/(mx*my)
```

```
 print 77,k,-deep(k),veval(k)*100.0,(vmax(i,k)*100.0,i=1,6)
 print 88,(ii(i,k),jj(i,k),i=1,6)
enddo
77      format(/i4,f8.0,7f8.2)
88      format(20x,6(2x,2i3))
 end
```

A.4 Include File (pstate.h)

```
c-----------------------------------------------------------------
      integer nx,ny,nz
      parameter (nx=360,ny=180,nz=50)
      common /pstate/rho,rhot,
     &           q,px,py,pz,qx,qy,qz,u,v,w,rhox,rhoy,
     &           rhotx,rhoty,rhotz,ug,vg,fac_scale,power,
     &           lat,ff,zero,one,pi,deg2rad,
     &           dx,dy,
     &           mx,my,mz,global,istopog,
     &           qmask,rmask,invmask

      real rho(nx,2,nz),rhot(nx,2,nz)
      real q(nx,2,nz)
      real px(nx,nz),py(nx,nz),pz(nx,nz)
      real qx(nx,nz),qy(nx,nz),qz(nx,nz)
      real u(nx,2,nz),v(nx,2,nz),w(nx,nz)
      real fac_scale,power
      real rhox(nx,nz),rhoy(nx,nz)
      real rhotx(nx,nz),rhoty(nx,nz),rhotz(nx,nz)
      real ug(nx,nz),vg(nx,nz)
      real ff(ny),lat(ny)
      real dx(ny),dy,one,zero,pi,deg2rad
      integer mx,my,mz,global
      integer istopog(nx,ny),rmask(nx,ny),invmask(nx,ny)
```

A.5 Data Input

The *subroutine userinp* is used to read (T, S, H) data into the P-vector inverse model. Here, H is the water depth.

```
*************************************************
   subroutine userinp(stopog,deep,temp,salt,pref,ts,sect,j)
   include 'pstate.h'
   integer i,j,k,ts,sect
   real temp(nx,2,nz),salt(nx,2,nz),deep(nz),pref
   real pressure,saltd,tempd,dum,svan,tempt,theta
   integer stopog(nx,ny)
```

```
c
c — clear local array
c
   do i=1,mx
      do k=1,mz
         rho(i,sect,k)=0.0
         rhot(i,sect,k)=0.0
      enddo
   enddo
c — ts is a keyword, if ts=1 means input temp ans salt,
c    otherwise input is rho
c
c ——— (User dependent inputs) ——————————
c
c
c — read stopog(mx,my)
c — stopog(i,j) is the last 'wet' level in the column (i,j)
c    iunit=10
c    call read_topo(stopog,deep,iunit)
c
c — read temperature and salinity
c
     read(8,*) ((temp(i,sect,k),i=1,mx),k=1,mz)
     read(9,*) ((salt(i,sect,k),i=1,mx),k=1,mz)
c———————— (end of user dependent inputs)————————
c
c — compute in situ density RHO and potential density RHOT
c
c    Note: To minimize roundoff use sig=rho-1000.
c    svan is UNESCO (Fofonoff) subroutine.

    do i=1, mx
     do k=1, istopog(i,j+1)
       pressure= abs(deep(k))
       saltd=salt(i,sect,k)
       tempd=temp(i,sect,k)
       dum = svan(saltd,tempd,pressure,rho(i,sect,k) )
       if (pref .lt.0) then
       rhot(i,sect,k)=rho(i,sect,k)
       else
         tempt= theta(saltd,tempd,pressure,pref)
         dum = svan(saltd,tempt,pref,rhot(i,sect,k))
```

```
            endif
          enddo
        enddo
      return
     end

      subroutine read_topo(stopog,deep,iunit,ts,topofile,
     &                     saltfile)
c ------------------------------
c     USER supplied subroutine to:
c
c     read stopog(1:mx,1:my) where
c     stopog(i,j) is the deepth and
c     istopog(i,j) is the last wet point at (i,j);
c     that is, at (i,j) the levels istopog(i,j)+1,..,mz
c     if any, are land points.
c ------------------------------
      include 'pstate.h'
      integer iunit,i,j,k,ts
      integer stopog(nx,ny),minto
      real deep(nz)
      character*30 topofile,saltfile
c-----------------------------------------------------------
open(iunit,file=topofile,form='formatted')
do j=1,my
      read(iunit,*) (stopog(i,j),i=1,mx)
enddo
close(iunit)
minto=1000
do i=1,mx
      do j=1,my
         minto=min(minto,stopog(i,j))
      enddo
enddo
if(minto.lt.-2) then
      do i=1,mx
         do j=1,my
            istopog(i,j)=0
            do k=1,mz
               if(deep(k).ge.stopog(i,j)) istopog(i,j)=k
            enddo
         enddo
      enddo
```

```
    else
      do i=1,mx
        do j=1,my
          istopog(i,j)=stopog(i,j)
          stopog(i,j)=deep(istopog(i,j))
        enddo
        enddo
    endif
    if(ts.eq.2) then
      open(iunit,file=saltfile)
        do j=1,my
          read(iunit,*) ((w(i,k),i=1,mx),k=1,mz)
          do i=1,mx
            minto=0
            do k=1,mz
              if(w(i,k).gt.-80.0) minto=k
            enddo
              istopog(i,j)=min(istopog(i,j),minto)
          enddo
        enddo
      close(iunit)
    endif

    return
    end
```

A.6 Equation of State (whoi.f)

The program *whoi.f* is used to calculate density from (T, S).

```
**************************************************
C Program whoi.f
C JUNE 24 1982
      REAL FUNCTION SVAN(S,T,P0,SIGMA)
C MODIFIED RCM
C **************************************************
C SPECIFIC VOLUME ANOMALY (STERIC ANOMALY) BASED ON
       1980 EQUATION
C OF STATE FOR SEAWATER AND 1978 PRACTICAL SALINITY
       SCALE.
C REFERENCES
C MILLERO, ET AL (1980) DEEP-SEA RES.,27A,255-264
C MILLERO & POISSON 1981,DEEP-SEA RES.,28A PP 625-629.
C BOTH ABOVE REFERENCES ARE ALSO FOUND IN UNESCO
       REPORT 38 (1981)
```

```
C      UNITS:
C     PRESSURE    P0    DECIBARS
C     TEMPERATURE    T    DEG    CELSIUS (IPTS-68)
C     SALINITY    S    PSU    (IPSS-78)
C     SPEC. VOL. ANA. SVAN    M**3/KG *1.0E-8
C     DENSITY ANA.    SIGMA    KG/M**3
C     CHECK VALUE: SVAN = 981.30190 M**3/KG FOR S = 40 PSU,
C     T = 40 DEG C, P0= 10000 DECIBARS.
C ************************************************************
C ****************************
C CHECK VALUE: SIGMA = 59.82037 KG/M**3 FOR S = 40 PSU,
C T = 40 DEG C, P0= 10000 DECIBARS.
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
C EQUIV
      EQUIVALENCE (E,D,B1),(BW,B,R3),(C,A1,R2)
      EQUIVALENCE (AW,A,R1),(KW,K0,K)
C ********************
C DATA
      DATA R3500,R4/1028.1063,4.8314E-4/
      DATA DR350/28.106331/
C    R4 IS REFERED TO AS C IN MILLERO & POISSON 1981
C CONVERT PRESSURE TO BARS AND TAKE SQUARE
ROOT SALINITY.
      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
   R1 = ((((6.536332E-9*T-1.120083E-6)*T+1.001685E-4)*T
   X-9.095290E-3)*T+6.793952E-2)*T-28.263737
C SEAWATER DENSITY ATM PRESS.
C COEFFICIENTS INVOLVING SALINITY
C R2 = A IN NOTATION OF MILLERO & POISSON 1981
   R2 = (((5.3875E-9*T-8.2467E-7)*T+7.6438E-5)*T-4.0899E-3)*T
   X+8.24493E-1
C R3 = B IN NOTATION OF MILLERO & POISSON 1981
   R3 = (-1.6546E-6*T+1.0227E-4)*T-5.72466E-3
C INTERNATIONAL ONE-ATMOSPHERE EQUATION OF STATE OF
SEAWATER
   SIG = (R4*S + R3*SR + R2)*S + R1
C SPECIFIC VOLUME AT ATMOSPHERIC PRESSURE
   V350P = 1.0/R3500
   SVA = -SIG*V350P/(R3500+SIG)
   SIGMA=DR350-(SVA/(V350P*(V350P+SVA)))
```

```
C SCALE SPECIFIC VOL. ANAMOLY TO NORMALLY REPORTED
   UNITS
    SVAN=SVA*1.0E+8
    IF(P.EQ.0.0) RETURN
C ***************************************************************
C ****** NEW HIGH PRESSURE EQUATION OF STATE FOR
            SEAWATER ********
C ***************************************************************
C COMPUTE COMPRESSION TERMS
    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
C
    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
    X-0.1194975
    A = (D*SR + C)*S + AW
C
    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
    X+148.4206)*T-1930.06
    K0 = (B1*SR + A1)*S + KW
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 SCALE SPECIFIC VOL. ANAMOLY TO NORMALLY REPORTED UNITS
    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 PSU 0 DEG. C & ATMOSPHERIC
      PRES.
C 2) DR35P: DENSITY ANAMOLY 35 PSU 0 DEG. C , WITH PRES.
      VARIATION
```

```
C 3) DVAN : DENSITY ANAMOLY VARIATIONS INVOLVING SPECFIC
     VOL. ANAMOLY
C *****************************************************************
C CHECK VALUE: SIGMA = 59.82037 KG/M**3 FOR S = 40 PSU,
C T = 40 DEG C, P0= 10000 DECIBARS.
C *****************************************************
      D350=GAM/PK
      DR35P=R3500*D350
      DVAN=SVA/(V350P*(V350P+SVA))
      SIGMA=DR350+DR35P-DVAN
      RETURN
      END
C SAL78 FCN ********** OCT 24 1979 *******
      REAL FUNCTION SAL78(CND,T,P,C1535,M)
C ***************************************
C
C FUNCTION TO CONVERT CONDUCTIVITY TO SALINITY (M = 0)
  OR
C SALINITY TO CONDUCTIVITY (M = 1,CND = SAL)
C*****************************************************************
C  REFERENCES: ALSO LOCATED IN UNESCO REPORT # 37 1981
C PRACTICAL SALINITY SCALE 1978: E.L. LEWIS IEEE OCEAN
  ENG. JAN. 1980
C CONDUCTIVITY AT S=35.,T=15,P=0: CULKIN & SMITH IEEE
  OCEAN ENG. 1980.
C *****************************************************************
C C1535 = 42.914 MMHO/CM ADOPTED VALUE OF ABSOLUTE
  CONDUCTIVITY AT
C SALINITY 35 PSU AND TEMPERATURE 15 DEG CELSIUS
  (IPTS-68) FOR
C ATMOSPHERIC (0 DECIBARS) PRESSURE
C SAL78: RETURNS ZERO FOR CONDUCTIVITY: < 0.0005 MMHO/CM.
C SAL78: RETURNS ZERO FOR SALINITY: < 0.02    PSU
C UNITS:
C     PRESSURE      P       DECIBARS
C     TEMPERATURE     T       DEG CELSIUS (IPTS-68)
C     CONDUCTIVITY    CND     MMHO/CM.
C     SALINITY        SAL78     PSU (IPSS-78)
C CHECKVALUE: FOR C1535=42.914 MMHO/CM
C     SAL78 = 81.02555 :S=40 PSU,T=40 DEG C,P=10000
      DECIBARS:M=0
```

```
C     SAL78 = 40.00000 FOR G = 81.02555,T=40 DEG C,P=10000
      DECIBARS:M=1
C N FOFONOFF, REVISED JUNE 8 1982
C C ***********************************************************
C INTERNAL FUNCTIONS
C ************************************************************
C PRACTICAL SALINITY SCALE 1978 DEFINITION WITH
TEMPERATURE CORRECTION
C XT=T-15.0 : XR=SQRT(RT)
  SAL(XR,XT) =((((2.7081*XR-7.0261)*XR+14.0941)*XR+25.3851)*XR
  X-0.1692)* XR+0.0080
C TEMPERATURE CORRECTION TO 1978 PRACTICAL SALINITY
  SCALE: (XT=T-15.0)
  X  +(XT/(1.0+0.0162*XT))*(((((-0.0144*XR+
  X  0.0636)*XR-0.0375)*XR-0.0066)*XR-0.0056)*XR+0.0005)
C DSAL(XR,XT) EXPLICIT FUNCTION FOR DERIVATIVE
  OF SAL(XR,XT)
C WITH RESPECT TO CONDUCTIVITY RATIO XR.
  DSAL(XR,XT) =((((13.5405*XR-28.1044)*XR+42.2823)*XR+50.7702)*XR
  X  -0.1692)+(XT/(1.0+0.0162*XT))*((((-0.0720*XR+0.2544)*XR
  X  -0.1125)*XR-0.0132)*XR-0.0056)
C FUNCTION RT35 : CONDUCTIVITY RATIO C(35,T,0)/C(35,15,0)
  VARIATION
C WITH TEMPERATURE.
  RT35(XT) = (((1.0031E-9*XT-6.9698E-7)*XT+1.104259E-4)*XT
  X   + 2.00564E-2)*XT + 0.6766097
C POLYNOMIALS OF RP: CONDUCTIVITY RATIO C(S,T,P)/C(S,T,0)
  PRESSURE
C VARIATION: COEFFICIENTS PRESSURE SET FOR UNITS OF
  DECIBARS!!
C C(XP) POLYNOMIAL CORRESPONDS TO A1-A3 CONSTANTS:
  LEWIS 1980
   C(XP) = ((3.989E-15*XP-6.370E-10)*XP+2.070E-5)*XP
   B(XT) = (4.464E-4*XT+3.426E-2)*XT + 1.0
C A(XT) POLYNOMIAL CORRESPONDS TO B3 & B4 CONSTANTS:
  LEWIS 1980
  pcA(XT) = -3.107E-3*XT + 0.4215
C************************************************************
C TEST FOR ZERO OR NEGATIVE ABSOLUTE CONDUCTIVITY
C SET DEFAULT CONDUCTIVITY VALUE AT S=35, T=15 &
ATMOSPHERIC PRES.
  IF(C1535.LE.0.0) C1535=42.914
C ********************************
```

```
C ZERO SALINITY/CONDUCTIVITY TRAP
  SAL78=0.0
  IF((M.EQ.0).AND.(CND.LE.5E-4)) RETURN
  IF((M.EQ.1).AND.(CND.LE.0.02)) RETURN
C *********************************************
  DT = T - 15.0
C SELECT BRANCH FOR SALINITY (M=0) OR CONDUCTIVITY
  (M=1)
  IF(M.EQ.1) GO TO 10
C *********************************************
C CONVERT CONDUCTIVITY TO SALINITY
  R = CND/C1535
  RT = R/(RT35(T)*(1.0 + C(P)/(B(T) + A(T)*R)))
  RT = SQRT(ABS(RT))
  SAL78 = SAL(RT,DT)
  RETURN
C ********* END OF CONDUCTIVITY TO SALINITY SECTION ***
C *****************************************************
C INVERT SALINITY TO CONDUCTIVITY BY THE
C NEWTON-RAPHSON ITERATIVE METHOD.
C *****************************************************
C FIRST APPROXIMATION
  RT = SQRT(CND/35.0)
    SI = SAL(RT,DT)
    N = 0
C
C ITERATION LOOP BEGINS HERE WITH A MAXIMUM OF
     10 CYCLES
C
  RT = RT + (CND - SI)/DSAL(RT,DT)
    SI = SAL(RT,DT)
    N = N + 1
    DELS = ABS(SI - CND)
    IF((DELS.GT.1.0E-4).AND.(N.LT.10))GO TO 15
C
C ***************************END OF ITERATION LOOP ********
C
C COMPUTE CONDUCTIVITY RATIO
    RTT = RT35(T)*RT*RT
    AT = A(T)
    BT = B(T)
    CP = C(P)
    CP = RTT*(CP + BT)
    BT = BT - RTT*AT
```

```
C
C SOLVE QUADRATIC EQUATION FOR R: R=RT35*RT*(1+C/AR+B)
C
      R = SQRT(ABS(BT*BT + 4.0*AT*CP)) - BT
C CONDUCTIVITY RETURN
      SAL78 = 0.5*C1535*R/AT
      RETURN
      END
C DEPTH FCN ***** OCT 7 1980 *****
      REAL FUNCTION DEPTH(P0,LAT)
C ******************************
C DEPTH IN METERS FROM PRESSURE IN BARS USING
C SAUNDERS AND FOFONOFF'S METHOD.
C DEEP-SEA RES., 1976,23,109-111.
C FORMULA REFITTED FOR 1980 EQUATION OF STATE
C UNITS:
C      PRESSURE       P0       DECIBARS
C      DEPTH          DEPTH    METERS
C CHECKVALUE: DEPTH = 9712.654 M FOR P=10000 DECIBARS,
   LATITUDE=30 DEG
C     ABOVE FOR STANDARD OCEAN: T=0 DEG. CELSUIS ; S=35 PSU
C
   REAL LAT
C
X = SIN(LAT/57.29578)
C   SCALE PRESSURE TO BARS
P=P0/10.
C*************************
X = X*X
C GR= GRAVITY VARIATION WITH LATITUDE: ANON (1970)
   BULLETIN GEODESIQUE
      GR = 9.780318*(1.0+(5.2788E-3+2.36E-5*X)*X) + 1.092E-5*P
      DEPTH = (((-1.82E-11*P+2.279E-7)*P-2.2512E-3)*P+97.2659)*P
      DEPTH=DEPTH/GR
      RETURN
      END
C CPSW FCN *** OCT 3 1980 ****
      REAL FUNCTION CPSW(S,T,P0)
C ***************************
C SPECIFIC HEAT OF SEAWATER J/KG
C SALINITY PSU (IPSS-78), TEMPERATURE DEG. CELSIUS (IPTS-68),
C PRESSURE IN DECIBARS.
```

```
C REF: MILLERO ET AL,1973,JGR,78,4499-4507
C PRESSURE VARIATION FROM LEAST SQUARES POLYNOMIAL
C DEVELOPED BY FOFONOFF 1980.
C CHECK VALUE: CPSW = 3850.309 J/KG FOR S = 40 PSU,
C T = 40 DEG C, P0= 10000 DECIBARS
C   SCALE PRESSURE TO BARS
   P=P0/10.
C**************************
C SQRT SALINITY FOR FRACTIONAL TERMS
   SR = SQRT(ABS(S))
C SPECIFIC HEAT CP0 FOR P=0 (MILLERO ET AL 1973)
   A = (-1.3839E-3*T+0.107276)*T-7.6444
   B = (5.3539E-5*T-4.0772E-3)*T+0.17709
   C = (((2.093236E-5*T-2.654387E-3)*T+0.1412855)*T
  X    -3.720283)*T+4217.4
   CP0 = (B*SR + A)*S + C
C CP1 PRESSURE AND TEMPERATURE TERMS FOR S = 0
   A = (((1.7168E-8*T+2.0357E-6)*T-3.13885E-4)*T+1.45747E-2)*T
  X   -0.49592
   B = (((2.2956E-11*T-4.0027E-9)*T+2.87533E-7)*T-1.08645E-5)*T
  X   +2.4931E-4
   C = ((6.136E-13*T-6.5637E-11)*T+2.6380E-9)*T-5.422E-8
   CP1 = ((C*P+B)*P+A)*P
C CP2 PRESSURE AND TEMPERATURE TERMS FOR S > 0
   A = (((-2.9179E-10*T+2.5941E-8)*T+9.802E-7)*T-1.28315E-4)*T
  X   +4.9247E-3
   B = (3.122E-8*T-1.517E-6)*T-1.2331E-4
   A = (A+B*SR)*S
   B = ((1.8448E-11*T-2.3905E-9)*T+1.17054E-7)*T-2.9558E-6
   B = (B+9.971E-8*SR)*S
   C = (3.513E-13*T-1.7682E-11)*T+5.540E-10
   C = (C-1.4300E-12*T*SR)*S
   CP2 = ((C*P+B)*P+A)*P
C SPECIFIC HEAT RETURN
   CPSW = CP0 + CP1 + CP2
   RETURN
   END
C ATG FCN *** OCT 20 1980 ****
   REAL FUNCTION ATG(S,T,P0)
C ***************************
C ADIABATIC TEMPERATURE GRADIENT DEG C PER DECIBAR
C REF: BRYDEN,H.,1973,DEEP-SEA RES.,20,401-408
C UNITS:
```

```
C      PRESSURE       P0      DECIBARS
C      TEMPERATURE    T     DEG CELSIUS (IPTS-68)
C      SALINITY       S       PSU (IPSS-78)
C      ADIABATIC      ATG     DEG. C/DECIBAR
C CHECK: ATG=3.255976E-4 C/DBAR FOR S=40 PSU,T=40 DEG C,
P0=10000 DECIBAR
C   SCALE PRESSURE TO BARS
   P=P0/10.
C*************************
   DS = S - 35.0
   ATG = (((-2.1687E-13*T+1.8676E-11)*T-4.6206E-10)*P
  X+((2.7759E-10*T-1.1351E-8)*DS+((-5.4481E-12*T
  X+8.733E-10)*T-6.7795E-8)*T+1.8741E-6))*P
  X+(-4.2393E-7*T+1.8932E-5)*DS
  X+((6.6228E-9*T-6.836E-7)*T+8.5258E-5)*T+3.5803E-4
C    SCALE ATG TO PER DECIBAR
    ATG=0.1*ATG
C****************
    RETURN
    END
C THETA FCN ****** OCT 20 1980 ******
REAL FUNCTION THETA(S,T0,P0,PR)
C **********************************
C TO COMPUTE LOCAL POTENTIAL TEMPERATURE AT PR
C USING BRYDEN 1973 POLYNOMIAL FOR ADIABATIC LAPSE
C RATE AND RUNGE-KUTTA 4-TH ORDER INTEGRATION
C ALGORITHM.
C REF: BRYDEN,H.,1973,DEEP-SEA RES.,20,401-408
C FOFONOFF,N.,1977,DEEP-SEA RES.,24,489-491
C UNITS:
C      PRESSURE       P0     DECIBARS
C      TEMPERATURE    T0     DEG CELSIUS (IPTS-68)
C      SALINITY       S       PSU (IPSS-78)
C      REFERENCE PRS    PR      DECIBARS
C      POTENTIAL TMP.  THETA    DEG CELSIUS
C CHECKVALUE: THETA= 36.89072 C,S=40 PSU,T0=40 DEG C,
C P0=10000 DECIBARS,PR=0 DECIBARS
C
C    SET-UP INTERMEDIATE TEMPERATURE AND PRESSURE
      VARIABLES
   P=P0
   T=T0
C**************
   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
C SVEL FCN **** OCT 4 1980 *****
      REAL FUNCTION SVEL(S,T,P0)
C ******************************
C SOUND SPEED SEAWATER CHEN & MILLERO 1977,JASA,62,
C 1129-1135 UNITS:
C     PRESSURE      P0     DECIBARS
C     TEMPERATURE   T      DEG CELSIUS (IPTS-68)
C     SALINITY      S      PSU (IPSS-78)
C     SOUND SPEED   SVEL   METERS/SECOND
C CHECKVALUE: 1731.995 M/S FOR P0=10000 DECIBARS,
T=40 DEG C, S=40 PSU
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 BRVAL ***** BRUNT-VAISALA FREQ *****
C USES 1980 EQUATION OF STATE
   FUNCTION BVFRQ(S,T,P,NOBS,PAV,E)
C ***********************************
C UNITS:
C     PRESSURE      P0     DECIBARS
C     TEMPERATURE     T     DEG CELSIUS (IPTS-68)
C     SALINITY        S       PSU (IPSS-78)
C     BOUYANCY FREQ    BVFRQ    CPH
C     N**2          E       RADIANS/SECOND
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 R MILLARD
C JULY 12 1982
C COMPUTES N IN CYCLES PER HOUR, & E=N**2 IN RAD/SEC**2
C AFTER FORMULATION OF BRECK OWEN'S
   REAL*4 P(1),T(1),S(1)
   E=0.0
   BVFRQ=0.0
   IF(NOBS.LT.2) RETURN
   CXX=0.0
   CX=0.0
```

```
   CXY=0.0
   CY=0.0
C COMPUTE LEAST SQUARES ESTIMATE OF SPECIFIC VOLUME
   ANAMOLY GRADIENT
   DO 20 K=1,NOBS
  20 CX =CX+P(K)
   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
C
   IF(VBAR.EQ.0.0) RETURN
   E = -.96168423E-2*DVDP/(VBAR)**2
   BVFRQ = 572.9578*SIGN(SQRT(ABS(E)),E)
   RETURN
   END
```

A.7 P-Vector Calculation

The subroutine *pvector* is used to compute the P-vector.

```
**********************************************

      subroutine pvector(stopog,deep,ktop,sims,sect1,sect2,f0,j)
      include 'pstate.h'
      common /maxval6/ veval,vmax,ii,jj
      real veval(nz),vmax(6,nz)
      integer ii(6,nz),jj(6,nz)
      integer sect1,sect2
```

```
c-----------------------------------------------------------------------
  real aa,aa1,aa2,aa3,dr,dq,drdq
  real fac
c-----------------------------------------------------------------------
  real dz,dz1,dz2
  real zero,one,two,small,sims
  real g,rho0
  integer i,j,i1,j1,k,kk,kk1,kk2,kbot
  integer mx1,mz1,kpv(nz),mkpv,ktop
  integer stopog(nx,ny)
  real uref,vref,deep(nz)
  real minlat
c-----------------------------------------------------------------------
  real f0,small_rhotz
  real a11,a12,a22,t11,t12,t22,ff1,ff2,wk,rxz,ryz
  integer jms,jmn
c – physical constants
  g=9.81
  rho0=1028.0
c – numerical constants
  zero=0.0
  one=1.0
  two=2.0
  small=1.e-7
  minlat=5.0
  jms=1
  jmn=0
  mz1=mz-1
  mx1=mx-1
  if(global.gt.0) mx1=mx

  small_rhotz=1.0e-20
c    f0=0.5*(abs(ff(1))+abs(ff(my)))
c
c – clear local array
c
     do i=1,mx
      do k=1,mz
        rhox(i,k)=0.0
        rhoy(i,k)=0.0
        rhotx(i,k)=0.0
        rhoty(i,k)=0.0
        rhotz(i,k)=0.0
        q(i,sect2,k)=0.0
```

```
        qx(i,k)=0.0
        qy(i,k)=0.0
        qz(i,k)=0.0
        px(i,k)=0.0
        py(i,k)=0.0
        pz(i,k)=0.0
        ug(i,k)=0.0
        vg(i,k)=0.0
        u(i,sect2,k)=0.0
        v(i,sect2,k)=0.0
        w(i,k)=0.0
            enddo
                enddo

c
c – calaulate potential velocity q= f drhot/dz:
c
      do i=1,mx
       kbot=istopog(i,j+1)
       if(kbot.ge.2) then
         q(i,sect2,1)=ff(j+1)*(rhot(i,sect2,1)-rhot(i,sect2,2))/
     &            (deep(1)-deep(2))/f0
         q(i,sect2,kbot)=ff(j+1)*(rhot(i,sect2,kbot-1)-
     &            rhot(i,sect2,kbot))/(deep(kbot-1)-deep(kbot))/f0
         do k=2,kbot-1
          dz1=deep(k-1)-deep(k)
          dz2=deep(k)-deep(k+1)
          dz= dz1+dz2
          q(i,sect2,k)=ff(j+1)/f0*((rhot(i,sect2,k-1)-rhot(i,sect2,k))


     &                          *dz2/(dz1*dz)+(rhot(i,sect2,k)-
     &                          rhot(i,sect2,k+1))*dz1/(dz2*dz))
                 enddo
               endif
             enddo
             if(j.eq.0) return


      c
      c –          rhotz=d rhot/dz and qz=dq/dz in the wet domain starting
     from ktop
      c            Not: rhotz and qz at the staged grid(B grid)
      c
```

```
      do i=1,mx1
        i1=i+1
        if(i1.gt.mx) i1=i1-mx
        if(rmask(i,j).gt.1) then
      kbot=rmask(i,j)
      rhotz(i,1)=0.25*(rhot(i,sect1,1)+rhot(i1,sect1,1)+
     *           rhot(i,sect2,1)+rhot(i1,sect2,1)-
     *           rhot(i,sect1,2)-rhot(i1,sect1,2)-
     *           rhot(i,sect2,2)-rhot(i1,sect2,2))/
     *           (deep(1)-deep(2))
      qz(i,1)=0.25*(q(i,sect1,1)+q(i1,sect1,1)+
     *         q(i,sect2,1)+q(i1,sect2,1)-q(i,sect1,2)-
     *         q(i1,sect1,2)-q(i,sect2,2)-
     *         q(i1,sect2,2))/(deep(1)-deep(2))
      rhotz(i,kbot)=(rhot(i,sect1,kbot-1)+
     *           rhot(i1,sect1,kbot-1)+
     *         rhot(i,sect2,kbot-1)+rhot(i1,sect2,kbot-1)-
     *         rhot(i,sect1,kbot)-rhot(i1,sect1,kbot)-
     *         rhot(i,sect2,kbot)-rhot(i1,sect2,kbot))/
     *         (deep(kbot-1)-deep(kbot))/4.0
      qz(i,kbot)=-1.0
      do k=2,kbot-1
        pcdz1=deep(k-1)-deep(k)
        dz2=deep(k)-deep(k+1)
        dz= dz1+dz2
        aa1=rhot(i,sect1,k-1)+rhot(i1,sect1,k-1)+
     *    rhot(i,sect2,k-1)+rhot(i1,sect2,k-1)
      aa2=rhot(i,sect1,k)+rhot(i1,sect1,k)+
     *    rhot(i,sect2,k)+rhot(i1,sect2,k)
      aa3=rhot(i,sect1,k+1)+rhot(i1,sect1,k+1)+
     *    rhot(i,sect2,k+1)+rhot(i1,sect2,k+1)
           rhotz(i,k)=0.25*((aa1-aa2)*dz2/(dz1*dz)+
     *    (aa2-aa3)*dz1/(dz2*dz))
           aa1=q(i,sect1,k-1)+q(i1,sect1,k-1)+q(i,sect2,k-1)
     *    +q(i1,sect2,k-1)
           aa2=q(i,sect1,k)+q(i1,sect1,k)+q(i,sect2,k)
     *    +q(i1,sect2,k)
           aa3=q(i,sect1,k+1)+q(i1,sect1,k+1)+q(i,sect2,k+1)
     *    +q(i1,sect2,k+1)
           qz(i,k)=0.25*((aa1-aa2)*dz2/(dz1*dz)+
     *    (aa2-aa3)*dz1/(dz2*dz))
           enddo
         endif
        enddo
```

```
c
c – scale the d/dz
c
      do i=1,mx1
        do k=1,mz
          rhotz(i,k)=rhotz(i,k)*fac_scale
          qz(i,k)=qz(i,k)*fac_scale
        enddo
      enddo
c
c – calculate rhotx, rhoty, rhotz, rhox and rhoy
c – rhox=d rho/dx; rhoy= d rho/dy; etc.
c
      do i=1,mx1
       i1=i+1
       if(i1.gt.mx) i1=i1-mx
       kbot=rmask(i,j)
       if(kbot.ge.1) then
          do k=1,kbot
            rhox(i,k)=(rho(i1,sect1,k)+rho(i1,sect2,k)-
     *          rho(i,sect1,k)-rho(i,sect2,k))/(dx(j)+dx(j+1))
          rhoy(i,k)=-(rho(i,sect2,k)+rho(i1,sect2,k)-
     *          rho(i,sect1,k)-rho(i1,sect1,k))/(2.0*dy)
          rhotx(i,k)=(rhot(i1,sect1,k)+rhot(i1,sect2,k)-
     *         rhot(i,sect1,k)-rhot(i,sect2,k))/(dx(j)+dx(j+1))
          rhoty(i,k)=-(rhot(i,sect2,k)+rhot(i1,sect2,k)-
     *          rhot(i,sect1,k)-rhot(i1,sect1,k))/(2.0*dy)
          qx(i,k)=(q(i1,sect1,k)+q(i1,sect2,k)-q(i,sect1,k)
*               -q(i,sect2,k))/(dx(j)+dx(j+1))
          qy(i,k)=-(q(i,sect2,k)+q(i1,sect2,k)-q(i,sect1,k)
*               -q(i1,sect1,k))/(2.0*dy)
        enddo
        qx(i,kbot)=(stopog(i1,j)+stopog(i1,j+1)-stopog(i,j)-
*               stopog(i,j+1))/(dx(j)+dx(j+1))
        qy(i,kbot)=-(stopog(i1,j+1)+stopog(i,j+1)-stopog(i1,j)-
*               stopog(i,j))/(2.0*dy)
c – compute geostrophic shear (trapezoidal rule)
          fac=g/((ff(j)+ff(j+1))*rho0)
          ug(i,kbot)=0.0
          vg(i,kbot)=0.0
          do k=kbot,2,-1
```

```
            ug(i,k-1)=ug(i,k)+fac*(rhoy(i,k)+rhoy(i,k-1))
     *                *(deep(k-1)-deep(k))
            vg(i,k-1)=vg(i,k)-fac*(rhox(i,k)+rhox(i,k-1))
     *                *(deep(k-1)-deep(k))
          enddo
        endif
      enddo
c
c – compute the p vector, pvec = (px, py, pz)
c
c
c set p-vector to zero first (for plotting purposes)
c
      do i=1,mx1
        mkpv=0
        do k=ktop,rmask(i,j)-1
          aa1=1.0d10*(rhoty(i,k)*qz(i,k)-rhotz(i,k)*qy(i,k))
          aa2=1.0d10*(rhotz(i,k)*qx(i,k)-rhotx(i,k)*qz(i,k))
          aa3=1.0d10*(rhotx(i,k)*qy(i,k)-rhoty(i,k)*qx(i,k))
          aa=sqrt(aa1**2+aa2**2+aa3**2)
          dr=sqrt(rhotx(i,k)**2+rhoty(i,k)**2+rhotz(i,k)**2)
          dq=sqrt(qx(i,k)**2+qy(i,k)**2+qz(i,k)**2)
          drdq=aa/(1.0d10*dr*dq)
          if(drdq.ge.sims) then
            aa3=aa3*fac_scale
c           aa1=aa1/fac_scale
c           aa2=aa2/fac_scale
            aa=sqrt(aa1**2+aa2**2+aa3**2)
          px(i,k)=aa1/aa
          py(i,k)=aa2/aa
          pz(i,k)=aa3/aa
          mkpv=mkpv+1
          kpv(mkpv)=k
        endif
      enddo
c
c – optimization the uref and vref
c

        if(mkpv.ge.3) then
          t11=0.0
          t12=0.0
          t22=0.0
          ff1=0.0
          ff2=0.0
```

```
      do kk=1,mkpv
        k=kpv(kk)
        if(abs(rhotz(i,k)).gt.small_rhotz) then
          kk1=max(1,k-1)
          kk2=min(mz,k+1)
          wk=((deep(kk1)-deep(kk2))*1.0e-2)**power
          rxz=rhotx(i,k)/rhotz(i,k)
     *                *fac_scale
          ryz=rhoty(i,k)/rhotz(i,k)
     *                *fac_scale
          aa=1.0+rxz**2+ryz**2
          a11=wk*py(i,k)**2*aa
          a12=-wk*px(i,k)*py(i,k)*aa
          a22=wk*px(i,k)**2*aa
          t11=t11+a11
          t12=t12+a12
          t22=t22+a22
          ff1=ff1-a11*ug(i,k)-a12*vg(i,k)
          ff2=ff2-a12*ug(i,k)-a22*vg(i,k)
        endif
      enddo
      aa=t11*t22-t12**2
      if(aa.ne.0.0) then
        uref=(ff1*t22-ff2*t12)/aa
      vref=(t11*ff2-t12*ff1)/aa
      invmask(i,j)=rmask(i,j)
      do k=1,rmask(i,j)
        u(i,sect2,k)=uref+ug(i,k)
        v(i,sect2,k)=vref+vg(i,k)
      enddo
        else
        invmask(i,j)=-1
      endif
      else
        invmask(i,j)=-1
      endif
        if(rmask(i,j).eq.0) invmask(i,j)=1
      enddo
c
c – re-scale back
c
```

```
c
c – scale here before inversion, then re-scale back at the end
c
      do k=1,mz
        do i=1,mx1
          rhotz(i,k)=rhotz(i,k)/fac_scale
          qz(i,k)=qz(i,k)/fac_scale
        enddo
      enddo

c
c – compute vertical velocity
c
      if(j.gt.1) then
      do i=1,mx1
        i1=i-1
        if(i1.lt.1 .and. global.gt.0) i1=i1+mx
        if(i1.gt.0) then
          kbot=min(istopog(i,j),max(rmask(i,j),rmask(i1,j),
     *               rmask(i,j-1),rmask(i1,j-1)))
          if(kbot.ge.2) then
            uref=0.0
            vref=0.0
            mkpv=0
            do k=i1,i,(i-i1)
      do j1=1,2
      if(u(k,j1,kbot).ne.0.0) then
        uref=uref+u(k,j1,kbot)
        vref=vref+v(k,j1,kbot)
        mkpv=mkpv+1
        endif
      enddo
      enddo
      if(mkpv.gt.0) then
        uref=uref/mkpv
        vref=vref/mkpv
        w(i,kbot)=(uref*(stopog(i+1,j)-stopog(i1,j))/dx(j)
     *           +vref*(stopog(i,j-1)-stopog(i,j+1))/dy)/2.0
        do k=kbot-1,1,-1
          uref=(u(i,sect2,k)+u(i,sect2,k+1)+u(i,sect1,k)+
     *           u(i,sect1,k+1)-u(i1,sect2,k)
     *           -u(i1,sect2,k+1)-u(i1,sect1,k)-
     *           u(i1,sect1,k+1))/(2.0*(dx(j)+dx(j+1)))
          vref=(v(i,sect1,k)+v(i,sect1,k+1)+v(i1,sect1,k)+
     *          v(i1,sect1,k+1)-v(i,sect2,k)-v(i,sect2,k+1)-
     *          v(i1,sect2,k)-v(i1,sect2,k+1))/(4.0*dy)
        w(i,k)=w(i,k+1)-(uref+vref)*(deep(k)-deep(k+1))
```

```
      enddo
     endif
    endif
   endif
  enddo
endif

do k=ktop,mz
  do i=1,mx1
    aa=sqrt(u(i,sect2,k)**2+v(i,sect2,k)**2)
    call maxn(6,vmax,aa,ii,jj,i,j,k)
    veval(k)=veval(k)+aa
  enddo
enddo

return
end

subroutine maxn(n,v,a,ii,jj,i,j,kk)
parameter(nz=50)
integer i,j,k,kk,n,i,l,ii(n,nz),jj(n,nz)
real a,v(n,nz)
if(a.ge.v(1,kk)) then
  do k=n,2,-1
    v(k,kk)=v(k-1,kk)
    ii(k,kk)=ii(k-1,kk)
    jj(k,kk)=jj(k-1,kk)
  enddo
  v(1,kk)=a
  ii(1,kk)=i
  jj(1,kk)=j
else
  do k=2,n
    if(a.ge.v(k,kk) .and. a.lt.v(k-1,kk)) then
      do l=n,k+1,-1
        v(l,kk)=v(l-1,kk)
        ii(l,kk)=ii(l-1,kk)
        jj(l,kk)=jj(l-1,kk)
      enddo
      v(k,kk)=a
      ii(k,kk)=i
      jj(k,kk)=j
    endif
  enddo
endif

return
end
```

A.8 Avoiding P-Vector Calculation

The subroutine *setmask* is used to determine the area where the P-vector calculation is not valid (necessary conditions not being satisfied).

```
------------------------------------------
******************************************
      subroutine setmask(mx,my,istopog,rmask,invmask,global)
      parameter(nx=360,ny=180)
      integer istopog(nx,ny),rmask(nx,ny),invmask(nx,ny)
      integer i,j,i1,i2,i3,j1,j2,j3,mx,my,mx1,my1
      mx1=mx-1
      if(global.eq.1) mx1=mx
      my1=my-1
      do j=1,my1
           do i=1,mx1
            rmask(i,j)=0
           invmask(i,j)=0
         enddo
      enddo
c
c – calculate rmask
c
      do i=1,mx-1
         do j=1,my1
           rmask(i,j)=min(istopog(i,j),istopog(i,j+1),istopog(i+1,j),
     &                  istopog(i+1,j+1))
         enddo
      enddo
      if(global.eq.1) then
        do j=1,my1
          rmask(mx,j)=min(istopog(1,j),istopog(1,j+1),
     &                  istopog(mx,j),istopog(mx,j+1))
        enddo
      endif
  c
  c – re-calaulate rmask to void
  c
       do i=1,mx1
        do j=1,my1
          if(global.eq.1) then
           i1=i-1
           i2=i-2
           i3=i-3
           if(i1.lt.1) i1=i1+mx
           if(i2.lt.1) i2=i2+mx
```

```
        if(i3.lt.1) i3=i3+mx
        j1=i+1
        j2=i+2
        j3=i+3
        if(j1.gt.mx) j1=j1-mx
        if(j2.gt.mx) j2=j2-mx
        if(j3.gt.mx) j3=j3-mx
       else
       i1=max(1,i-1)
       i2=max(1,i-2)
       i3=max(1,i-3)
       j1=min(mx1,i+1)
       j2=min(mx1,i+2)
       j3=min(mx1,i+3)
      endif
      kk1=max(min(rmask(i,j),rmask(i1,j),rmask(i2,j),rmask(i3,j)),
     &        min(rmask(i,j),rmask(j1,j),rmask(j2,j),rmask(j3,j)))
      i1=max(1,j-1)
      i2=max(1,j-2)
      i3=max(1,j-3)
      j1=min(my1,j+1)
      j2=min(my1,j+2)
      j3=min(my1,j+3)
      kk2=max(min(rmask(i,j),rmask(i,i1),rmask(i,i2),rmask(i,i3)),
     &        min(rmask(i,j),rmask(i,j1),rmask(i,j2),rmask(i,j3)))
      rmask(i,j)=min(kk1,kk2)
     enddo
    enddo

    return
    end
```

A.9 Data Output

The following subroutines (*matopen*, *matout*, *matotal*, *matclose*) are used for the data output.

```
**********************************************************
*
     subroutine matopen(iuvuni,iunpxy,iunrho,iunuvg,iunq)
     integer iuvuni,iunpxy,iunrho,iunuvg,iunq
     open(iuvuni,file='puv.dat')
     open(iunpxy,file='pxyz.dat')
     open(iunrho,file='prho.dat')
     open(iunuvg,file='puvg.dat')
     open(iunq,file='pqxyz.dat')
     return
     end
**********************************************************
```

```
      subroutine matotal(mx1,my1,mz,istopog,rmask,invmask,deep,
      &         grid_size,lon0,lat0,ktop,pref,global)
      parameter(nx=360,ny=180,nz=50)
      integer invmask(nx,ny),rmask(nx,ny),istopog(nx,ny)
      integer i,j,k,mx1,my1,mz,ktop,iunit
      real pref,deep(nz),grid_size,lon0,lat0
c – control
      iunit=10
      open(iunit,file='pctrl.dat',form='formatted')
      write(iunit,102) mx1,my1,mz,grid_size,lat0,
      $       lon0,ktop,pref,global
      close(iunit)

c – masks
      open(iunit,file='pmask.dat',form='formatted')
      do i=1,mx1
        do j=1,my1
          write(iunit,103)
      $         invmask(i,j),min(istopog(i,j),istopog(i,j+1),
      $         istopog(i+1,j),istopog(i+1,j+1)),rmask(i,j)
        enddo
      enddo
      close(iunit)

      open(iunit,file='pdeep.dat',form='formatted')
      do k=1,mz
        write(iunit,101) deep(k)
      enddo
      close(iunit)
101     format(1x,e18.7)
102     format(3i4,3(1x,e18.7),i4,1x,e16.7,2x,i3)
103     format(5i8)

      return

      end

**************************************************
      subroutine matout(iuvuni,iunpxy,iunrho,iunuvg,iunq,
      &                 temp,salt,sect,f0)
      include 'pstate.h'
      real temp(nx,2,nz),salt(nx,2,nz),s2day,f0
```

```
      data s2day/86400.0/
      integer iuvuni,iunpxy,iunrho,iunuvg,iunq
      integer i,i1,k,mx1,sect
      mx1=mx-1

      if(global) mx1=mx
c – primary fields
      do i=1,mx1
        do k=1,mz
          write(iuvuni,100) u(i,sect,k),v(i,sect,k),s2day*w(i,k)
        enddo
      enddo

      do i=1,mx1
        do k=1,mz
          write(iunpxy,100) px(i,k),py(i,k),pz(i,k)
        enddo
      enddo

      do i=1,mx1
        i1=i+1
        if(i1.gt.mx) i1=i1-mx
        do k=1,mz
          write(iunrho,100)
     &         .25*(temp(i,1,k)+temp(i,2,k)+temp(i1,1,k)+temp(i1,2,k)),
     &         .25*(salt(i,1,k)+salt(i,2,k)+salt(i1,1,k)+salt(i1,2,k)),
     &         .25*(rho(i,1,k)+rho(i,2,k)+rho(i1,1,k)+rho(i1,2,k)),
     &         .25*(rhot(i,1,k)+rhot(i,2,k)+rhot(i1,1,k)+rhot(i1,2,k))
        enddo
      enddo

      do i=1,mx1
        do k=1,mz
          write(iunuvg,100) ug(i,k),vg(i,k)
        enddo
      enddo

      do i=1,mx1
        do k=1,mz
          write(iunq,100) (q(i,1,k)+q(i,2,k)+q(i1,1,k)+q(i1,2,k))
     &                *f0/4.0, qx(i,k), qy(i,k), qz(i,k)
        enddo
      enddo
```

```
 100   format(4(1x,e18.7))
 101   format(1x,e18.7)
 102   format(3i4,3(1x,e18.7),i4,1x,e16.7)
 103   format(5i8)

       return
       end
```

```
      ********************************************************
      subroutine matotal(mx1,my1,mz,istopog,rmask,invmask,deep,
      &           grid_size,lon0,lat0,ktop,pref,global)
      parameter(nx=360,ny=180,nz=50)
      integer invmask(nx,ny),rmask(nx,ny),istopog(nx,ny)
      integer i,j,k,mx1,my1,mz,ktop,iunit
      real pref,deep(nz),grid_size,lon0,lat0
c – control
      iunit=10
      open(iunit,file='pctrl.dat',form='formatted')
      write(iunit,102) mx1,my1,mz,grid_size,lat0,
      $      lon0,ktop,pref,global
      close(iunit)

c – masks
      open(iunit,file='pmask.dat',form='formatted')
      do i=1,mx1
        do j=1,my1
          write(iunit,103)
      $        invmask(i,j),min(istopog(i,j),istopog(i,j+1),
      $        istopog(i+1,j),istopog(i+1,j+1)),rmask(i,j)
        enddo
      enddo
      close(iunit)

      open(iunit,file='pdeep.dat',form='formatted')
      do k=1,mz
        write(iunit,101) deep(k)
      enddo
      close(iunit)
101   format(1x,e18.7)
102   format(3i4,3(1x,e18.7),i4,1x,e16.7,2x,i3)
103   format(5i8)

      return
      end
```

```
**************************************************
*
      subroutine matclose(iuvuni,iunpxy,iunrho,iunuvg,iunq)
      integer iuvuni,iunpxy,iunrho,iunuvg,iunq
      close(iuvuni)
      close(iunpxy)
      close(iunrho)
      close(iunuvg)
      close(iunq)
      return
      end
```

B

P-Vector Module for Isopycnal-Coordinate

Similar to Appendix A, the P-Vector module for isopycnal-coordinate is presented in Appendix B. This module is also stored in the enclosed DVD-Rom including the makefile, source codes, plotting files (using Matlab), and a sample case. Rongfeng Li and Chenwu Fan helped in developing these codes. Here, the code with the NetCDF output is listed for illustration.

B.1 Main Program

```
c     This program is calculating absolute current on isopycnals
c     at this subdirector program is running from south to north
c
c     sigmapvnc.f
c
c     to compile:
c        f77 -o sigmapvnc sigmapvnc.f -L/usr/local/netcdf-3.4/lib -
lnetcdf
c     Note: you may need to correct the netcdf lib path. If there is no
c        Netcdf lib support in your system, ask your system
c        administrator to download the netcdf lib.
c



      include 'comblk.h'
      include 'netcdf.inc'

      real lon0, lat0, latmin, lonm0, latm0
      integer data_type, dim3d(3), dim2d(2),
     &    ist3(3),ict3(3),dimts3d(3), istts3(3), ictts3(3)
c        data_type 0-ASCII 1-binary
      real*8 facu, fact, ofstst, ofstr, factpv
      integer*2 i1d(im*(kl+jm)), i1s(im*kl), gb2
      integer*1 i11d(im*kl)
      character titl*80, author*20, created*9
```

```
      data facu, factpv/1.0d-4,1.0d-2/
      data fact, ofstst, ofstr/1.0d-3, 1.5d1, 30.0d0/
      data pi/3.1415926535/
      data radius/6.371e6/omega/7.292e-5/ g/9.806/
      data rho0/1025./

      data rlim/32767.0/
      data gb2 /-32767/

      data ist3,ict3/1,1,1,kl1,im1,1/
      data istts3,ictts3/1,1,1,kl,im,1/
      open(8,file='input.dat')
        read(8,*) lat0,lon0,y_gridsize,x_gridsize,data_type
        read(8,*) layers(1),drho, pref
        read(8,*) deep
        read(8,'(a20)') author
        read(8,'(a80)') titl
        read(8,*) kts,kpvect,kq
      close(8)

      do k=1,km1
        deepm(k)=(deep(k)+deep(k+1))/2.0
      enddo

      deg2rad=pi/180.
      small=1.e-6

      lonm0=lon0+x_gridsize/2.0
      latm0=lat0+y_gridsize/2.0

      latmin=5.
c ———— form depths spd (246) for interpolation ————
      spd(1)=deep(1)
      do kk=1,km-1
        dd=(deep(kk+1)-deep(kk))/nkp
        do k=nkp*(kk-1)+2,nkp*kk+1
          spd(k)=spd(k-1)+dd
        enddo
      enddo


c – calculating lat(j) lon(i),ff(j) Change them when using another
c       domain

      do j=1,jm
        lat(j)=lat0+y_gridsize*(j-1)
      enddo
```

```
      ddx=x_gridsize*deg2rad*radius
      dy=y_gridsize*deg2rad*radius

      do j=1,jm
        dx(j)=ddx*cos(lat(j)*deg2rad)
        ff(j)=2.*omega*sin(lat(j)*deg2rad)
        if(lat(j).lt.latmin .and. lat(j) .ge.0.0) then
         ff(j)=2.*omega*sin(latmin*deg2rad)
        endif
        if(lat(j).gt.-latmin .and. lat(j) .le.0.0) then
         ff(j)=-2.*omega*sin(latmin*deg2rad)
        endif
      enddo

c ———— for wnp pref=0.0 taking isopynals as follow ——
      do k=2,kl
        layers(k)=layers(1)+drho*(k-1)
        layerm(k-1)=0.5*(layers(k-1)+layers(k))
      enddo

      call date(created)

      IRET = NF_CREATE('pvout.nc',NF_CLOBBER,NCID)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,NF_GLOBAL,
     &          'title',80,titl)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,NF_GLOBAL,'author',29,
     &                  author//created)
      CALL ERRCHECK(IRET)
c
c     defind dimensions (lat, lon, lev)
c
      IRET = NF_DEF_DIM(NCID,'lat',jm,IDJDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'lat_uv',jm1,IDJUVDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'lon',im,IDIDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'lon_uv',im1,IDIUVDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'zlev',km,IDKDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'sigma_level',kl,IDsigDIM)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_DIM(NCID,'sigma_level_uv',kl1,IDsigmDIM)
      CALL ERRCHECK(IRET)
```

```
c     defined coordinate data (lat, lon, deep)
c

      IRET = NF_DEF_VAR(NCID,'lat',NF_float,1,IDJDIM,
     &           IDYCOORD)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'lat_uv',NF_float,1,IDJUVDIM,
     &           IDYUVCOORD)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'lon',NF_float,1,IDIDIM,
     &           IDXCOORD)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'lon_uv',NF_float,1,
     &           IDIUVDIM,IDXUVCOORD)
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'depth',NF_float,1,
     &           IDKDIM,IDZCOORD)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDZCOORD,
     &           'units',6,'meters')
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'sigma_level',NF_float,1,
     &           IDsigDim,IDsig)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDsig,'units',7,'kg/m**3')
      CALL ERRCHECK(IRET)
      IRET = NF_DEF_VAR(NCID,'sigma_level_uv',NF_float,1,
     &           IDsigmDim,IDsigm)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDsigm,'units',7,'kg/m**3')
      CALL ERRCHECK(IRET)

      IRET = NF_PUT_ATT_TEXT(NCID,IDYCOORD,'long_name',
     &           30, 'Latitudinal Position for T,S,R')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDYUVCOORD,
     &           'long_name',28, 'Latitudinal Position for u,v')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDXCOORD,
     &           'long_name',31, 'Longitudinal Position for T,S,R')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDXUVCOORD,
     &           'long_name',29, 'Longitudinal Position for u,v')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDZCOORD,
     &           'long_name',27, 'Vertical Position for T,S,R')
      CALL ERRCHECK(IRET)
```

```
   dim3d(1)=IDsigmDIM
   dim3d(2)=IDIUVDIM
   dim3d(3)=IDJUVDIM

   dim2d(1)=IDIUVDIM
   dim2d(2)=IDJUVDIM

   dimts3d(1)=IDsigDim
   dimts3d(2)=IDIDIM
   dimts3d(3)=IDJDIM

   IRET = NF_DEF_VAR(NCID,'ktop',NF_SHORT,2,
&          dim2d,IDKTOP)
   CALL ERRCHECK(IRET)

   IRET = NF_PUT_ATT_TEXT(NCID,IDKTOP,'long_name',21,
&          'Top Sigma Level Index')
   CALL ERRCHECK(IRET)

   IRET = NF_DEF_VAR(NCID,'kbot',NF_SHORT,2,dim2d,
&          IDKBOT)
   CALL ERRCHECK(IRET)

   IRET = NF_PUT_ATT_TEXT(NCID,IDKBOT,'long_name',24,
&          'Bottom Sigma Level Index')
   CALL ERRCHECK(IRET)

  if(kts.gt.0) then
   IRET = NF_DEF_VAR(NCID,'temperature',NF_SHORT,3,
&          dimts3d,IDT)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_TEXT(NCID,IDT,'units',1,'C')
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDT,'scale_factor',
&          NF_double,1,fact)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDT,'add_offset',
&          NF_double,1,ofstst)
   CALL ERRCHECK(IRET)
   IRET = NF_DEF_VAR(NCID,'salinity',NF_SHORT,3,
&          dimts3d,IDS)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDS,'scale_factor',
&          NF_double,1,fact)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_DOUBLE(NCID,IDS,'add_offset',
&          NF_double,1,ofstst)
   CALL ERRCHECK(IRET)
   IRET = NF_PUT_ATT_TEXT(NCID,IDS,'units',3,'PSU')
   CALL ERRCHECK(IRET)
```

```
      IRET = NF_DEF_VAR(NCID,'sigma_depth',NF_float,3,
     &           dimts3d, IDsdep)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDsdep,'units',1,'m')
      CALL ERRCHECK(IRET)
     endif

     if(kq.gt.0) then
      IRET = NF_DEF_VAR(NCID,'q',NF_float,3,dim3d,IDQ)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDQ,'long_name',19,
     &                 'Potential Vorticity')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDQ,'units',3,'1/s')
      CALL ERRCHECK(IRET)
     endif

      IRET = NF_DEF_VAR(NCID,'u',NF_SHORT,3,dim3d,IDU)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDU,'long_name',14,
     &                 '3-D u-velosity')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDU,'units',3,'m/s')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_DOUBLE(NCID,IDU,'scale_factor',
     &                 NF_double,1,facu)
      CALL ERRCHECK(IRET)

      IRET = NF_DEF_VAR(NCID,'v',NF_SHORT,3,dim3d,IDV)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDV,'long_name',14,
     &                 '3-D v-velosity')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDV,'units',3,'m/s')
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_DOUBLE(NCID,IDV,'scale_factor',
     &                 NF_double,1,facu)
      CALL ERRCHECK(IRET)

      IRET = NF_DEF_VAR(NCID,'sigma_depth_uv',NF_float,3,
     &           dim3d, IDsmdep)
      CALL ERRCHECK(IRET)
      IRET = NF_PUT_ATT_TEXT(NCID,IDsmdep,'units',1,'m')
      CALL ERRCHECK(IRET)
```

```
      if(kpvect.gt.0) then
       IRET = NF_DEF_VAR(NCID,'pvector_x',NF_BYTE,3,
     &           dim3d,IDPX)
       CALL ERRCHECK(IRET)
       IRET = NF_PUT_ATT_DOUBLE(NCID,IDPX,'scale_factor',
     &           NF_double,1,factpv)
       CALL ERRCHECK(IRET)
       IRET = NF_DEF_VAR(NCID,'pvector_y',NF_BYTE,3,
     &           dim3d,IDPY)
       CALL ERRCHECK(IRET)
       IRET = NF_PUT_ATT_DOUBLE(NCID,IDPY,'scale_factor',
     &                  NF_double,1,factpv)
       CALL ERRCHECK(IRET)
      endif

      IRET = NF_ENDDEF(NCID)
      CALL ERRCHECK(IRET)
c
c     write coordinate data
c
      r1d(1)=lon0
      do i=2,im
       r1d(i)=r1d(i-1)+x_gridsize
      enddo
      IRET = NF_PUT_VAR_real(NCID,IDXCOORD,r1d)
      CALL ERRCHECK(IRET)

      r1d(1)=lonm0
      do i=2,im1
       r1d(i)=r1d(i-1)+x_gridsize
      enddo
      IRET = NF_PUT_VAR_real(NCID,IDXUVCOORD,r1d)
      CALL ERRCHECK(IRET)

      IRET = NF_PUT_VAR_real(NCID,IDYCOORD,lat)
      CALL ERRCHECK(IRET)
      do j=1,jm1
       r1d(j)=(lat(j)+lat(j+1))/2.0
      enddo
      IRET = NF_PUT_VAR_real(NCID,IDYUVCOORD,r1d)
      CALL ERRCHECK(IRET)

      IRET = NF_PUT_VAR_real(NCID,IDZCOORD,deep)
      CALL ERRCHECK(IRET)
```

```
      IRET = NF_PUT_VAR_real(NCID,IDsig,layers)
      CALL ERRCHECK(IRET)

      IRET = NF_PUT_VAR_real(NCID,IDsigm,layerm)
      CALL ERRCHECK(IRET)
c
      is1=2
      is2=1

      ifdt=18
      ifds=19

      if(data_type.eq.0) then
        open(ifdt,file='t.txt')
        open(ifds,file='s.txt')
      else
        open(ifdt,file='t.bin',
     &       form='unformatted',access='direct',recl=km*im)
        open(ifds,file='s.bin',
     &       form='unformatted',access='direct',recl=km*im)
      endif

      j=0
      j1=j+1
      call qinput(ifdt,ifds,j1,data_type)

      call qlyddz(drho,is2,j1)

      istts3(3)=j1
      if(kts.gt.0) then
        do i=1,im
          if(kb(i,j1).le.kt(i,j1)) then
            do k=1,kl
              i1d(kl*(i-1)+k)=gb2
              i1s(kl*(i-1)+k)=gb2
            enddo
          else
            do k=1,kt(i,j1)-1
              i1d(kl*(i-1)+k)=gb2
              i1s(kl*(i-1)+k)=gb2
            enddo
            k1=1
            k2=k1+1
            do k=kt(i,j1),kb(i,j1)
              do while(lyd(k,i,is2).gt.deep(k2))
                k1=k2
                k2=k1+1
              enddo
```

```
            h1=(deep(k2)-lyd(k,i,is2))/(deep(k2)-deep(k1))
            h2=1.0-h1
            i1d(kl*(i-1)+k)=(tp(k1,i)*h1+tp(k2,i)*h2-ofstst)/fact+0.5
            i1s(kl*(i-1)+k)=(sp(k1,i)*h1+sp(k2,i)*h2-ofstst)/fact+0.5
          enddo
          do k=kb(i,j1)+1,kl
            i1d(kl*(i-1)+k)=gb2
            i1s(kl*(i-1)+k)=gb2
          enddo
        endif
      enddo

      IRET = NF_PUT_VARA_int2(NCID,IDT,istts3,ictts3,i1d)
      CALL ERRCHECK(IRET)

      IRET = NF_PUT_VARA_int2(NCID,IDS,istts3,ictts3,i1s)
      CALL ERRCHECK(IRET)

      IRET = NF_PUT_VARA_real(NCID,IDsdep,istts3,ictts3,
     &                        lyd(1,1,is2))
      CALL ERRCHECK(IRET)

      endif

      do 1000 j=1,jm1
        j1=j+1
        iiss=is1
        is1=is2
        is2=iiss

        print *,'j=',j

        call qinput(ifdt,ifds,j+1,data_type)

        call qlyddz(drho,is2,j1)

        call fmask(is2,j1)

        call qcv(is1,is2,j)

        call caluv(drho,rho0,small,is1,is2,j)
c
c – output
c
        istts3(3)=j+1
        ist3(3)=j

        if(kts.gt.0) then
          do i=1,im
            if(kb(i,j1).le.kt(i,j1)) then
```

```
        do k=1,kl
          i1d(kl*(i-1)+k)=gb2
          i1s(kl*(i-1)+k)=gb2
        enddo
      else
        do k=1,kt(i,j1)-1
          i1d(kl*(i-1)+k)=gb2
          i1s(kl*(i-1)+k)=gb2
        enddo
        k1=1
        k2=k1+1
        do k=kt(i,j1),kb(i,j1)
          do while(lyd(k,i,is2).gt.deep(k2))
            k1=k2
            k2=k1+1
          enddo
          h1=(deep(k2)-lyd(k,i,is2))/(deep(k2)-deep(k1))
          h2=1.0-h1
          i1d(kl*(i-1)+k)=(tp(k1,i)*h1+tp(k2,i)*h2-ofstst)/fact+0.5
          i1s(kl*(i-1)+k)=(sp(k1,i)*h1+sp(k2,i)*h2-ofstst)/fact+0.5
        enddo
        do k=kb(i,j1)+1,kl
          i1d(kl*(i-1)+k)=gb2
          i1s(kl*(i-1)+k)=gb2
        enddo
      endif
    enddo
    IRET = NF_PUT_VARA_int2(NCID,IDT,istts3,ictts3,i1d)
    CALL ERRCHECK(IRET)

    IRET = NF_PUT_VARA_int2(NCID,IDS,istts3,ictts3,i1s)
    CALL ERRCHECK(IRET)

    IRET = NF_PUT_VARA_real(NCID,IDsdep,istts3,ictts3,
&             lyd(1,1,is2))
    CALL ERRCHECK(IRET)
  endif
    if(kq.gt.0) then
      do i=1,im1
        if(maskb(i,j).le.maskt(i,j)) then
          do k=1,kl1
            r1d(kl1*(i-1)+k)=0.0
          enddo
        else
          do k=1,maskt(i,j)-1
            r1d(kl1*(i-1)+k)=0.0
          enddo
```

```
      do k=maskt(i,j),maskb(i,j)
        r1d(kl1*(i-1)+k)=0.25*(q(k,i,is1)+q(k,i,is2)+
&                              q(k,i+1,is1)+q(k,i+1,is2))
      enddo
      do k=maskb(i,j)+1,kl1
        r1d(kl1*(i-1)+k)=0.0
      enddo
    endif
  enddo
  IRET = NF_PUT_VARA_real(NCID,IDQ,ist3,ict3,r1d)
  CALL ERRCHECK(IRET)
 endif

 do i=1,im1
  do k=1,kl1
    uuu=u(k,i)/facu+0.5
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(kl1*(i-1)+k)=uuu
  enddo
 enddo
 IRET = NF_PUT_VARA_int2(NCID,IDU,ist3,ict3,i1d)
 CALL ERRCHECK(IRET)

 do i=1,im1
  do k=1,kl1
    uuu=v(k,i)/facu+0.5
    uuu=max(-rlim,uuu)
    uuu=min(rlim,uuu)
    i1d(kl1*(i-1)+k)=uuu
  enddo
 enddo
 IRET = NF_PUT_VARA_int2(NCID,IDV,ist3,ict3,i1d)
 CALL ERRCHECK(IRET)

 if(kpvect.gt.0) then
  do i=1,im1
    do k=1,kl1
      i11d(kl1*(i-1)+k)=px(k,i)/factpv
    enddo
  enddo
  IRET = NF_PUT_VARA_int1(NCID,IDPX,ist3,ict3,i11d)
  do i=1,im1
    do k=1,kl1
      i11d(kl1*(i-1)+k)=py(k,i)/factpv
    enddo
  enddo
  IRET = NF_PUT_VARA_int1(NCID,IDPY,ist3,ict3,i11d)
 endif
```

```
      IRET = NF_PUT_VARA_real(NCID,IDsmdep,ist3,ict3,lydm)
      CALL ERRCHECK(IRET)

1000 continue

      close(ifdt)
      close(ifds)

      do i=1,im1
        do j=1,jm1
          i1d(im1*(j-1)+i)=maskt(i,j)
        enddo
      enddo
      IRET = NF_PUT_VAR_int2(NCID,IDKTOP,i1d)
      CALL ERRCHECK(IRET)

      do i=1,im1
        do j=1,jm1
          i1d(im1*(j-1)+i)=maskb(i,j)
        enddo
      enddo
      IRET = NF_PUT_VAR_int2(NCID,IDKBOT,i1d)
      CALL ERRCHECK(IRET)

      IRET = NF_CLOSE(NCID)

      stop
      end
```

B.2 Graphic Interface

```
C
      SUBROUTINE ERRCHECK(IRET)
C*****************************************************
C
C     *ERRCHECK*    ERROR HANDLER FOR GRAPHICAL
C     INTERFACE
C
C     AUTHOR - ROGER PROCTOR AND PATRICK LUYTEN
C
C     LAST UPDATE - 12 Jul 1999 @(COHERENS)netcdfint.f 8.4
C
C
C     DESCRIPTION - WRITE ERROR MESSAGE AND STOP
C     EXECUTION OF THE PROGRAM
```

```
C       IF AN ERROR OCCURS IN ONE OF THE netCDF
C       ROUTINES
C
C       REFERENCE -
C
C       CALLING PROGRAM - CDF2D, CDF3D
C
C       EXTERNALS - NF_STRERROR
C
C*************************************************
C
      INCLUDE 'netcdf.inc'

C
C*      ARGUMENTS
C
      INTEGER IRET

      IF (IRET.NE.NF_NOERR) THEN
        WRITE (0,*) NF_STRERROR(IRET)
        STOP
      ENDIF

      RETURN

      END
c — This program is for inputing data of T and S and then
c      interpolating T,S from 56 levels into 246 levels
c
c      qinput.f cf77 qinput.f intpo.f intpo.f

      subroutine qinput(ifdt,ifds,j,data_type)

      include 'comblk.h'

      real rhoz(kp)
      integer data_type

c ———— input tp(im,56) and sp(im,56) ————————-

      if(data_type.eq.0) then
        read(ifdt,*) tp
        read(ifds,*) sp
      else
        read(ifdt,rec=j) tp
        read(ifds,rec=j) sp
      endif
```

```
      do i=1,im
       itopo(i,j)=0
       do k=1,km
        if(tp(k,i).gt.-99.0 .and. sp(k,i).gt.-99.0) then
         itopo(i,j)=itopo(i,j)+1
        else
         goto 15
        endif
       enddo
15      continue
      enddo
```

B.3 Transformation of (T, S) Data from z- to Isopycnal-Coordinate

The subroutines *qinput.f*, *interpo.f*, and *cub.f* are used to input and to transform (T, S) data from z-levels (e.g., 33 levels in WOA atlas) to n_σ isopycnal levels (e.g., 236 isopycnal levels in Sect 6.6).

```
**************************************************
*
      subroutine qinput(ifdt,ifds,deep,tp,sp,spd,tpp,spp,is1,j)

      include 'comblk.h'

      real tp(mx,km,2),sp(mx,km,2)
      real tpp(mx,kp,2),spp(mx,kp,2)
      real deep(km),spd(kp)

      real rhoz(kp)
      dimension wk(mx,kp)
      dimension x(2,km),t(kp),a(km),b(km),s(kp)

c ———— input tp(mx,33) and sp(mx,33) ————————

*     read(ifdt,rec=my-j+1) ((tp(i,k,is1),k=1,km),i=1,mx)
*     read(ifds,rec=my-j+1) ((sp(i,k,is1),k=1,km),i=1,mx)
      read(ifdt,rec=my-j+1) ((tp(i,k,is1),i=1,mx),k=1,km)
      read(ifds,rec=my-j+1) ((sp(i,k,is1),i=1,mx),k=1,km)

c     print *,'reading T and S j= ',j

      if(j .eq.99 ) then
       print *,' tp(304,99,k)= '
       print 666,(tp(304,k,is1),k=1,km)
       print *,' sp(304,99,k)= '
       print 666,(sp(304,k,is1),k=1,km)
      endif
666     format(5f12.4)
```

```
      do 22 i=1,mx
       itopo(i,j)=0
       do k=1,km
        if(tp(i,k,is1).gt.-99.0 .and. sp(i,k,is1).gt.-99.0) then
          itopo(i,j)=itopo(i,j)+1
        endif
       enddo
22      continue

c ———————— intepolate tp(mx,33) into tpp(mx,236) ————————

      call intpo(tp,wk,deep,spd,x,t,a,b,s,is1,j)

      do 55 i=1,mx
      do 55 k=1,kp
55      tpp(i,k,is1)=wk(i,k)

c ———————— intepolate sp(mx,my,56) into spp(mx,my,246) ——

      call intpo(sp,wk,deep,spd,x,t,a,b,s,is1,j)

      do 56 i=1,mx
      do 56 k=1,kp
56      spp(i,k,is1)=wk(i,k)

c —— form index field itopp(i,j) according to tpp and spp ——
      do 23 i=1,mx
       itopp(i,j)=0
       do k=1,kp
        if(tpp(i,k,is1).gt.-99. .and. spp(i,k,is1).gt.-99.) then
          itopp(i,j)=itopp(i,j)+1
        endif
       enddo
23      continue

c ———————— calculating portential density rhot ————————————

      do 57 i=1,mx
      do 57 k=1,kp
         rho(i,k,is1)=0.0
         rhot(i,k,is1)=0.0
57      continue
      do 66 i=1,mx
       if(itopp(i,j) .eq. 0) go to 66
       do k=1,itopp(i,j)
          tempd=tpp(i,k,is1)
          saltd=spp(i,k,is1)
          pressure=spd(k)
```

```
        dum=svan(saltd,tempd,pressure,rho(i,k,is1))
        tempt=theta(saltd,tempd,pressure,pref)
        dumt=svan(saltd,tempt,pref,rhot(i,k,is1))
      enddo
66      continue

      if(j .eq.99 ) then
      print *,' rhot(304,99,k)= '
      print 666,(rhot(304,k,is1),k=1,kp)
      print *,' rhot(305,99,k)= '
      print 666,(rhot(305,k,is1),k=1,kp)
      endif

      return
      end
**********************************************
      subroutine intpo(ts,wk,deep,spd,x,t,a,b,s,is1,j)
      include 'comblk.h'

      real ts(mx,km,2)
      real wk(mx,kp)
      real deep(km),spd(kp)

      dimension x(2,km),t(kp),a(km),b(km),s(kp)

c ------------------------------------------------
      do 11 i=1,mx
      do 11 k=1,kp
11      wk(i,k)=-99.9999

c ---- intepolating ts(mx,km) to wk(mx,kp) ----------

      do 100 i=1,mx
        kkb=itopo(i,j)
        if(itopo(i,j).le.1) go to 100
        q1=ts(i,1,is1)
        q2=ts(i,2,is1)
        y11=(q2-q1)/(deep(2)-deep(1))
        q3=ts(i,kkb-1,is1)
        q4=ts(i,kkb,is1)
        yn1=(q4-q3)/(deep(kkb)-deep(kkb-1))

        dd=deep(kkb)
        if(dd.gt.0.0.and.dd.le.1000.) then
          m=ifix(dd/10.+1.e-10)+1
        else if(dd.gt.1000.0 .and. dd .le.2500.) then
           m=ifix((dd-1000.)/20.+1.e-10)+101
        else if(dd.gt.2500.0.and.dd.le.5500.) then
             m=ifix((dd-2500.)/50.+1.e-10)+176
        endif
```

```
      if(i.eq.304 .and. j.eq.99) then
        print *,' for B point i=304, j=99 '
        print *,' m= ',m,' kb= ',kkb,' deep(kkb)= ',deep(kkb)
      endif

      do k=1,kp
        t(k)=0.0
        s(k)=0.0
      enddo
      do kk=1,m
        t(kk)=spd(kk)
      enddo

      do k=1,km
        x(1,k)=deep(k)
        x(2,k)=ts(i,k,is1)
        a(k)=0.0
        b(k)=0.0
      enddo

      call cub(km,kkb,m,kp,x,t,a,b,s,y11,yn1)

      do kk=1,m
        wk(i,kk)=s(kk)
      enddo

100     continue

      return
      end

************************************************

subroutine cub(n,ns,m,mb,x,t,a,b,s,y11,yn1)
      dimension x(2,n),a(n),b(n),t(mb),s(mb)

      a(1)=0.0
      b(1)=y11
c       n1=n-1
      n1=ns-1
      do 1 j=2,n1
        h1=x(1,j)-x(1,j-1)
        h=x(1,j+1)-x(1,j)
        af=h1/(h+h1)
        bt=3.*((1.-af)*(x(2,j)-x(2,j-1))/h1+af*(x(2,j+1)-x(2,j))/h)
        a(j)=-af/(2.+(1.-af)*a(j-1))
1       b(j)=(bt-(1.-af)*b(j-1))/(2.+(1.-af)*a(j-1))
c       a(n)=yn1
        a(ns)=yn1
2       a(n1)=a(n1)*a(n1+1)+b(n1)
        n1=n1-1
        if(n1.gt.0) go to 2
```

```
      do 5 j=1,m
        i=1
3       if(t(j).le.x(1,i+1)) go to 4
        i=i+1
        go to 3
4       h=x(1,i+1)-x(1,i)
        h1=(x(1,i+1)-t(j))/h
        h2=(t(j)-x(1,i))/h
5       s(j)=(3.-2.*h1)*h1**2*x(2,i)+(3.-2.*h2)*h2**2*x(2,i+1)+
     &       (1.-h1)*h1**2*h*a(i)-(1.-h2)*h2**2*h*a(i+1)

      return
      end
```

B.4 Calculation of Level Depth, Layer Thickness, and Potential Vorticity

The subroutine *qlyddz* is used to calculate the level depth (lyd), layer thickness (dz) and potential vorticity (q).

```
**********************************************************

      subroutine qlyddz(drho,layers,spd,is1,j)

      include 'comblk.h'

      real layers(mz),spd(kp),rhoz(kp)

c ------------ Calculate the level depth ------------
      do 88 i=1,mx
      do 88 k=1,mz
88      lyd(i,k,is1)=9999.9

      do 99 k=1,kp
99      rhoz(k)=0.0
      do 11 i=1,mx
        mb=itopp(i,j)
        if(mb.le.1) goto 11
       do k=1,mb
        rhoz(k)=rhot(i,k,is1)
       enddo
       do k=1,mz
        rhof=layers(k)
        lyd(i,k,is1)=dpint(kp,mb,spd,rhoz,rhof,1)
       enddo
11    continue
```

```
c ———————— Calculate the layer thickness ————————
      do 12 i=1,mx
      do 12 k=1,mz-1
        dd=lyd(i,k+1,is1)-lyd(i,k,is1)
        if (lyd(i,k,is1).ge.lyd(i,k+1,is1).or.
     &                  lyd(i,k+1,is1).ge.9999.9) then
          dz(i,k,is1)=0.0
        else
          dz(i,k,is1)=dd
        endif
12    continue

c ———————— calculating q=drho/rhok*f/dz ————————
      do 63 i=1,mx
      do 63 k=1,mz-1
63      q(i,k,is1)=9999.9

      do 64 i=1,mx
      do 64 k=2,mz-1
         if (dz(i,k-1,is1).gt.0.0 .and. dz(i,k,is1).gt.0.0) then
           rhok=1000.0+(layers(k)+0.5*drho)
           fact=drho/rhok
           q(i,k,is1)=fact*ff(j)/dz(i,k,is1)
         endif
64     continue

      if(j.eq.100) then
        print *,' '
      print *, 'lyd(304,100,k)lyd(305,100,k)dz(304,100,k)dz(305,100,k)'
        print *,' '
          do k=1,mz-1
            dd1=lyd(304,k,is1)
            dd2=lyd(305,k,is1)
            print 56,k,dd1,dd2,dz(304,k,is1),dz(305,k,is1)
          enddo
      endif
      if(j.eq.99) then
        print *,' '
      print *,'lyd(304,99,k)lyd(305,99,k)dz(304,99,k)dz(305,99,k)'
        print *,' '
          do k=1,mz-1
            dd1=lyd(304,k,is1)
            dd2=lyd(305,k,is1)
            print 56,k,dd1,dd2,dz(304,k,is1),dz(305,k,is1)
          enddo
      endif
56     format(1x,i4,4f12.4)

      return
      end
```

B.5 Calculation of Absolute Velocity

The subroutine *caluvg* is used to calculate the abosulte velocity.

```
**************************************************
      subroutine caluvg(drho,rho0,small,is1,is2,j)

      include 'comblk.h'

      real hx(mx-1,mz-1),hy(mx-1,mz-1)
      real dhx(mz-1),dhy(mz-1)
c     real dumk(mz-1),dvmk(mz-1),hm(mz-1)
      real hm(mz-1)
      real pxm(mz-1),pym(mz-1)

c ————————— calculate hx and hy from South to North ——
      do 10 k=1,mz-1
      do 10 i=1,mx-1
         hx(i,k)=0.0
         hy(i,k)=0.0
         u(i,k)=0.0
         v(i,k)=0.0
         ug(i,k)=0.0
         vg(i,k)=0.0
10      continue

      do 21 i=1,mx-1
        ktop=maskt(i,j)
        kbot=maskb(i,j)
        if(ktop.gt.0 .and.ktop.lt.kbot) then
          do k=ktop,kbot
            q1=dz(i,k,is1)
            q2=dz(i+1,k,is1)
            q3=dz(i+1,k,is2)
            q4=dz(i,k,is2)
            if(q1 .gt.0.0 .and.q2 .gt.0.0 .and.
     &         q3 .gt.0.0 .and.q4 .gt.0.0) then
            hx(i,k)=(q3+q2-q4-q1)/(dx(j+1)+dx(j))
            hy(i,k)=(q3+q4-q2-q1)/(2.*dy)
            endif
          enddo
        endif
21    continue
```

```
c ———————— start to calculate current ————————
      do 1000 i=1,mx-1
        invmask(i,j)=0
       do k=1,mz-1
        u(i,k)=0.0
        v(i,k)=0.0
       enddo
1000   continue

      do 2000 i=1,mx-1
       ktop=maskt(i,j)+5
       kbot=maskb(i,j)-1

       if(ktop.gt.0 .and. ktop.lt.kbot) then
        fuv=0.5*(ff(j)+ff(j+1))
        if (fuv.eq.0.0) goto 2000
        fac=g*drho/(fuv*rho0)

c – calculate d(Mk-Mkbot)/dy/fac, d(Mk-Mkbot)/dx/fac
      do k=1,mz-1
       dhx(k)=0.0
       dhy(k)=0.0
       hm(k)=0.0
       pxm(k)=0.0
       pym(k)=0.0
      enddo

      do k=ktop,kbot
       dhx(k)=hx(i,k)
       dhy(k)=hy(i,k)
       pxm(k)=px(i,k)
       pym(k)=py(i,k)
       hm(k)=0.25*(dz(i,k,is1)+dz(i+1,k,is1)+
     &                  dz(i+1,k,is2)+dz(i,k,is2))
       ug(i,k)=-pmkm(mz,k,kbot,dhy)*fac
       vg(i,k)=pmkm(mz,k,kbot,dhx)*fac
      enddo

c —— optmxization the uref and vref ——————————
      t11=0.0
      t12=0.0
      t22=0.0
      ff1=0.0
      ff2=0.0
```

```
      do k=ktop,kbot
        wk=hm(k)**2
        a11=wk*pym(k)**2
        a12=-wk*pxm(k)*pym(k)
        a22=wk*pxm(k)**2
        t11=t11+a11
        t12=t12+a12
        t22=t22+a22
c       ff1=ff1-a11*dumk(k)-a12*dvmk(k)
c       ff2=ff2-a12*dumk(k)-a22*dvmk(k)
        ff1=ff1-a11*ug(i,k)-a12*vg(i,k)
        aff2=ff2-a12*ug(i,k)-a22*vg(i,k)
      enddo
        aa=t11*t22-t12**2
        if(abs(aa).ge.1.e-10) then
          uref=(ff1*t22-ff2*t12)/aa
          vref=(t11*ff2-t12*ff1)/aa
          invmask(i,j)=0
      do k=ktop,kbot
c       u(i,k)=uref+dumk(k)
c       v(i,k)=vref+dvmk(k)
        u(i,k)=uref+ug(i,k)
        v(i,k)=vref+vg(i,k)
      enddo
     else
       invmask(i,j)=-1
     endif
   endif
2000  continue

  if(j.eq.99) then
   print *,' '
   print *, 'u(304,99,k) v(304,99,k) '
   print *,' '
    do k=1,mz-1
     print 56,k,u(304,k),v(304,k)
    enddo
  endif
56  format(1x,i4,1x,2(1x,f9.6))

  return
  end
```

B.6 Functions

Several functions are difined for the computation.

```
**********************************************************
      function dpint(n,mb,dep,r,rhof,key)
       dimension dep(n),r(n)
       data eps/1.e-20/

       do i=1,mb-1
        if((r(i)-rhof)*(r(i+1)-rhof).le.0.0) goto 10
       enddo
       dpint=9999.9
       return

10      continue
       if(key.eq.1) then
        i2=i+1
        x1=(rhof-r(i2))/(r(i)-r(i2)+eps)
        dpint=dep(i)*x1+dep(i2)*(1.0-x1)
        return
       elseif(key.eq.2) then
        if(i.eq.1) then
         i1=1
         i2=2
         i3=3
        elseif(i.eq.mb-1) then
         i1=i-1
         i2=i
         i3=mb
        elseif((rhof-r(i))/(r(i+1)-r(i)+eps).le.0.5) then
         i1=i-1
         i2=i
         i3=i+1
        else
         i1=i
         i2=i+1
         i3=i+2
        endif
        dpint=dep(i1)*(rhof-r(i2))*(rhof-r(i3))/
     &              ((r(i1)-r(i2))*(r(i1)-r(i3))+eps)
     &     + dep(i2)*(rhof-r(i1))*(rhof-r(i3))/
     &              ((r(i2)-r(i1))*(r(i2)-r(i3))+eps)
     &     + dep(i3)*(rhof-r(i1))*(rhof-r(i2))/
     &              ((r(i3)-r(i1))*(r(i3)-r(i2))+eps)
        return
```

```
      else
       if(i.eq.1) then
        i1=1
        i2=2
        i3=3
        i4=4
       elseif(i.eq.mb-1) then
        i1=mb-3
        i2=mb-2
        i3=mb-1
        i4=mb
       else
        i1=i-1
        i2=i
        i3=i+1
        i4=i+2
       endif
       dpint=dep(i1)*(rhof-r(i2))*(rhof-r(i3))*(rhof-r(i4))/
     &         ((r(i1)-r(i2))*(r(i1)-r(i3))*(r(i1)-r(i4))+eps)
     &      +dep(i2)*(rhof-r(i1))*(rhof-r(i3))*(rhof-r(i4))/
     &         ((r(i2)-r(i1))*(r(i2)-r(i3))*(r(i2)-r(i4))+eps)
     &      +dep(i3)*(rhof-r(i1))*(rhof-r(i2))*(rhof-r(i4))/
     &         ((r(i3)-r(i1))*(r(i3)-r(i2))*(r(i3)-r(i4))+eps)
     &      +dep(i4)*(rhof-r(i1))*(rhof-r(i2))*(rhof-r(i3))/
     &         ((r(i4)-r(i1))*(r(i4)-r(i2))*(r(i4)-r(i3))+eps)
       return
      endif

      end
c ------------------------------------------------------------
      function pmkm(kl,kk,km,dh)
      real dh(kl-1)

      sumh=0.0
      do k=1,kl-1
       sumh=sumh+dh(k)
      enddo

      sumrok=0.0
      do j=1,kl-1-kk
       sumi=0.0
       do i=1,j
        sumi=sumi+dh(kl-i)
       enddo
       sumrok=sumrok+(sumh-sumi)
      enddo
```

```
      sumrom=0.0
       do j=1,kl-1-km
        sumi=0.0
        do i=1,j
          sumi=sumi+dh(kl-i)
        enddo
        sumrom=sumrom+(sumh-sumi)
      enddo

      pmkm=(sumrok-sumrom)
      end
```

B.7 Avoiding P-Vector Calculation

The subroutine *fmask* is used to determine the area where the P-vector calculation is not valid (necessary conditions not being satisfied).

```
**************************************************
*
      subroutine fmask(is1,j,key)

      include 'comblk.h'

      do 11 i=1,mx
         kt(i,j)=0
11       kb(i,j)=0
      do 12 i=1,mx-1
         maskt(i,j)=0
12       maskb(i,j)=0

      do 22 i=1,mx

        if(dz(i,1,is1).gt.0.1e-20) then
          kt(i,j)=1
        else
          do k=2,mz-1
            if(dz(i,k-1,is1).le.0.0.and.dz(i,k,is1).gt.0.1e-20) then
              kt(i,j)=k
              goto 22
            endif
          enddo
        endif
22    continue
```

```
      do 23 i=1,mx
         kb(i,j)=kt(i,j)
        do k=kt(i,j)+1,mz-1
         if(dz(i,k,is1).gt.0.0) then
           kb(i,j)=kb(i,j)+1
         endif
      enddo
23     continue

      if(key.eq.0) return

      do 25 i=1,mx-1
        msmax=max(kt(i,j),kt(i+1,j),kt(i+1,j+1),kt(i,j+1))
        msmin=min(kb(i,j),kb(i+1,j),kb(i+1,j+1),kb(i,j+1))
        if(msmax.ge.msmin) then
          maskt(i,j)=0
          maskb(i,j)=0
          maskc(i,j)=0
        else
          maskt(i,j)=msmax
          maskb(i,j)=msmin
          maskc(i,j)=8
        endif
25    continue

      if (j.eq.99 ) then
        print *,' kt(304,99) kt(305,99) kt(305,100) kt(304,100)'
        print 55, kt(304,99),kt(305,99),kt(305,100),kt(304,100)
        print *,' '
        print *,' maskt(304,99)= ',maskt(304,99)
        print *,' '
        print *,' kb(304,99) kb(305,99) kb(305,100) kb(304,100)'
        print 55, kb(304,99),kb(305,99),kb(305,100),kb(304,100)
        print *,' '
        print *,' maskb(304,99)= ',maskb(304,99)
        print *,' '
      endif
55     format (1x,i5,5x,i5,5x,i5,5x,i5)

      return
      end
```

C

Thermohaline Parametric Model

The codes for calculating the autocorrelation function and decorrelation scale from observational (T, S) profiles were originally written by Steven D. Haeger in 1994, modified by Charles Fralick in 1995 and by Qianqian Wang in 1998.

C.1 Makefile

```
*********************************************************
FFLAGS = -C     # check array bounds (default f77 compiling.)

GMODEL.OBJ = gmodel.o linr.o rms.o gfit.o grad3a.o grad3b.o dep1.o \
          g3at.o grad1.o dep2.o dep3t.o grad2.o dep3.o bump.o inout4.o\
          dep4.o dep4t.o grad4.o dep5.o dpall2.o
gmodel: $(GMODEL.OBJ)
          f77 -o $@ -n     $(GMODEL.OBJ) -L/usr/local/disspla11/lib \
```

C.2 Gradient Calculated from (T, S) Profiles

The program, *gradstat.f*, is use to calculate the vertical gradient from the (T, S) profile data, and to plot the gradient versus depth.

```
C ************************************************
C *                  PROGRAM GRADSTAT                *
C ************************************************
```

```
C INPUT FILE: 3 (MASTER FILE - UNFORMATTED)
C
C *** DEFINITION OF TEMP AND DEPTH VARIABLES ***
C
C  T: READ FROM MASTER FILE. PASSED TO LINR TO BE
C  INTERPOLATED.
C  T2: INTERPOLATED TEMP PASSED BACK FROM LINR AND
C  PASSED TO TOP.
C  GRAD: GRADIENT COMPUTED FROM T2
C  D: READ FROM MASTER FILE AND PASS TO LINR.
C  D2: DEPTH PASSED BACK FROM LINR.
C  D3: DEPTHS FOR GRADIENT PLOT (2 LESS THAN D2)
C
      CHARACTER IDENT*10,PARM*1,HEADER*60
      DIMENSION IFLAG(8),T(2500),D(2500)
      DIMENSION GRAD(201),D3(201)
      DIMENSION T2(201),D2(201)
      IU=3
      IEND=0
      ICNT=0
      ICNT1=0
      IERR0 = 0
       IERR1=0
       IERR2=0
       IERR3=0
       IERR4=0
      N=0
      IDAT=0
      IFIRST=0
C
C  M= NO OF SETS N= NO OF PTS EACH SET
C  X ARRAY CONTAINS COMPUTED MODEL COEFFICIENTS-
C  SEE COMMENTS IN TOP
C  FOR DESCRIPTION
C
C
       WRITE(6,*) ('WHAT IS MAX DEPTH OF GRAD MODEL? (M)')
       READ(5,*) DMAX
       WRITE(6,*) ('WHAT IS MIN DEPTH TO ACCEPT PROFILE? (M)')
       READ(5,*) DMIN
       WRITE(6,*) ('ENTER STARTING PROFILE')
       READ(5,*) IFIRST
       WRITE(6,*) ('HOW MANY PROFILES?')
       READ(5,*) NUMPRO
```

```
C
C
C  READ HEADER RECORD FROM MASTER FILE
C
     READ(3,ERR=9500) PARM,HEADER
     WRITE(6,600) PARM,HEADER
600 FORMAT(/,1X,'INPUT FILE DATA TYPE: ',A1,/, 1X,'HEADER:
',A60,/)
C
     WRITE(4,620) PARM,DMIN,DMAX
620 FORMAT(1X,A1,3X,F6.0,3X,F6.0)
C
C  LOOP THROUGH UNWANTED PROFILES
C
     IF(IFIRST.NE.1) THEN
     DO 100 I=1,IFIRST-1
     READ(3,END=9000,ERR=9500)
     NPROF,(IFLAG(J),J=1,8),XLAT,XLON,
    *JPROF,ICLAS,IPAT,IMASS,IPROV,
*WDEP1,WDEP2,IYEAR,IMON,IDAY,XHOUR,IDENT,ISOURC,NC
*YC1,NCYC2,
    *EXTRA,IEXTRA,(D(J),T(J),J=1,NCYC1+NCYC2)
     ICNT = ICNT+1
100 CONTINUE
   END IF
C
C READ DESIRED PROFILES FROM MASTER FILE
C
   DO 12 I=1,NUMPRO
   READ(3,END=9001,ERR=9500) NPROF,(IFLAG(J),J=1,8),XLAT,XLON,
   *JPROF,ICLAS,IPAT,IMASS,IPROV,
   *WDEP1,WDEP2,IYEAR,IMON,IDAY,XHOUR,IDENT,ISOURC,NCYC1,
   *NCYC2,
   *EXTRA,IEXTRA,(D(J),T(J),J=1,NCYC1+NCYC2)
   NCYC=NCYC1+NCYC2
   ICNT1 = ICNT1+1
C
C PROFILE HAS BEEN READ - CHECK FLAG(1) FOR UNWANTED
C PROFILE
C
   IF(IFLAG(1).NE.0) THEN
   IERR0 = IERR0+1
   GO TO 12
   END IF
```

```
C
C  CHECK TO SEE IF PROFILE EXTENDS TO DEPTH OF DMIN
C
      IF(D(NCYC).LT.DMIN) THEN
      IERR1 = IERR1+1
      GO TO 12
      END IF
      IKEEP = IKEEP+1
C
C  INTERPOLATE PROFILE TO D(NCYC) OR DMAX, WHICHEVER
C  WHICHEVER IS GREATER WITH LINR
C
     DELX = DMIN/50.
     IF(DELX.LT.2.) DELX = 2.
     BOT = DMAX
     IF(D(NCYC).LT.DMAX) BOT=D(NCYC)
     WRITE(6,*) ('DELX,BOT,D(NCYC) = '),DELX,BOT,D(NCYC)
     CALL LINR(DELX,NCYC,0.,BOT,ICT,D,T,D2,T2)
     WRITE(6,*) ('ICT = '),ICT
C
C  CALCULATE GRADIENT
C
     GMAX = 0.
     GMIN = 999.
     DO 20 J=1,ICT-1
     GRAD(J) = (T2(J+1)-T2(J)) / DELX
     D3(J) = D2(J) + (DELX/2.)
     IF(GRAD(J).GT.GMAX) GMAX = GRAD(J)
     IF(GRAD(J).LT.GMIN) GMIN = GRAD(J)
20   CONTINUE
     WRITE(6,605) GMIN,GMAX
605  FORMAT(1X,',GMIN,GMAX = ',2F8.4)
C
C
C WRITE FILE TO PLOT GRADIENT
C
     XMIN = -.1
     XMAX = .1
     DO 21 L=1,10
     IF(GMIN.LT.XMIN) XMIN = XMIN - .1
     IF(GMAX.GT.XMAX) XMAX = XMAX + .1
21 CONTINUE
     WRITE(11,33) GMIN,GMAX,XMIN,XMAX,ICNT,DMIN,DMAX
```

```
C
      GO TO 9002
C
9000 WRITE(6,650) ICNT
 650 FORMAT(1X,'BEGINNING PROFILE NOT FOUND',/,
     *1X,'EOF ENCOUNTERED AFTER PROFILE',I5,'. END
     * PROGRAM.')
      GO TO 1000
C
9001 IEND = 1
9002 LAST = IFIRST+ICNT1-1
      WRITE(6,651) ICNT1,IFIRST,LAST
651  FORMAT(1X,I5,' PROFILES READ FROM MASTER FILE',/,
     *1X,'FIRST PROFILE:',I5,5X,'LAST PROFILE:',I5)
      GO TO 1000
C
9500 ITOT = ICNT+ICNT1
      WRITE(6,652) ITOT
652 FORMAT(1X,'READ ERROR AFTER RECORD NUMBER
     *(PROFILE)',I5,/,
     *1X,'*CHECK MASTER FILE* PROGRAM TERMINATED.')
C
1000 CONTINUE
      END
```

C.3 Main Program

The software, gmodel.f, is used to calculate the thermohaline parameters from observational (T, S) profiles for nonpolar region (see Sect. 2.4). User may use different numbers of (depth, gradient) parameters (called *coefficient*s in the code).

```
C ***************************************************
C *                                                 *
C *                  PROGRAM GMODEL                 *
C *                                                 *
C ***************************************************
```

```
C
C
C  VERSION FOR YELLOW SEA CALCULATION (Section 2.4)
C
C  THIS PROGRAM READ PROFILES FROM THE MASTER FILE
C  AND COMPUTES
C  THE GRADIENT VS DEPTH.
C
C  INPUT FILE: 3 (MASTER FILE - UNFORMATTED)
C
C  *** DEFINITION OF TEMP AND DEPTH VARIABLES ***
C
C   T: READ FROM MASTER FILE. PASSED TO LINR TO BE
C   INTERPOLATED.
C   T2: INTERPOLATED TEMP PASSED BACK FROM LINR AND
C   PASSED TO TOP.
C   TM: MODELED TEMPS COMPUTED FROM MODELED
C   GRADIENTS
C   GRAD: GRADIENT COMPUTED FROM T2
C   XG: MODELED GRADIENT PASSED BACK FROM GFIT
C   D: READ FROM MASTER FILE. PASSED TO LINR.
C   D2: DEPTH PASSED BACK FROM LINR.
C   D3: DEPTHS FOR GRADIENT PLOT (ONE LESS THAN D2)
C
      CHARACTER IDENT*10,PARM*1,HEADER*60,IOPT*1
      CHARACTER HD*60
      DIMENSION IFLAG(8),T(2500),D(2500)
      DIMENSION GRAD(400),D3(400),XG(400)
      DIMENSION T2(401),D2(401),TM(401)
      DIMENSION CD(7),CG(7),COEF(11),X2(12)
      IU=3
      IEND=0
      ICNT=0
      ICNT1=0
      IERR0 = 0
       IERR1=0
       IERR2=0
       IERR3=0
       IERR4=0
      N=0
      IDAT=0
      IFIRST=0
C
      WRITE(6,604)
```

```
 604 FORMAT(//,1X,' — PROGRAM GMODEL —',/,
    *1X,'RUN BATHTAG FIRST IF DMIN WILL BE LESS THAN
     DMAX',/)
C
      CALL INOUT4
99    FORMAT (A25)
      OPEN (2,FILE='fguess.dat',FORM='FORMATTED',STATUS=
      'OLD')
      WRITE(6,*) ('WHAT IS MAX DEPTH OF GRAD MODEL? (M)')
      READ(5,*) DMAX
      WRITE(6,*) ('WHAT IS MIN DEPTH TO ACCEPT PROFILE?
      (M)')
      READ(5,*) DMIN
      WRITE(6,*) ('ENTER STARTING PROFILE')
      READ(5,*) IFIRST
      WRITE(6,*) ('HOW MANY PROFILES?')
      READ(5,*) NUMPRO
C
      WRITE(6,*) ('DO YOU WANT TO WRITE OUT COEFS?
      (Y OR N)')
      READ(5,500) IOPT
500   FORMAT(A1)
C*********************************************************
C READ HEADER RECORD FROM MASTER FILE
C*********************************************************
      READ(3,ERR=9500) PARM,HEADER
      WRITE(6,600) PARM,HEADER
600   FORMAT(/,1X,'INPUT FILE DATA TYPE: ',A1,/,
      *1X,'HEADER: ',A60,/)
      HD=('REJECTED PROFILES FROM GMODEL')
      WRITE(9) PARM,HD
      WRITE(8) PARM,HEADER
C
      WRITE(6,621) DMAX,DMIN
621   FORMAT(/,1X,'GRADIENT MODEL EXTENDS TO ',F6.0,' M',/,
      *1X,'PROFILES AT LEAST ',F4.0,' M WILL BE ACCEPTED',/)
C*********************************************************
C COMPUTE DELZ AND WRITE HEADER TO OUTPUT FILE
C*********************************************************
```

```
      DELZ = DMAX/400.
      IF(DELZ.LT.0.5) DELZ = 0.5
C
      IF(IOPT.EQ.'Y') THEN
      WRITE(4,609) DELZ,DMIN,DMAX
      WRITE(7,609) DELZ,DMIN,DMAX
609   FORMAT(1X,3F8.2)
      END IF
C*********************************************************
C  LOOP THROUGH UNWANTED PROFILES
C*********************************************************
       IF(IFIRST.NE.1) THEN
       DO 100 I=1,IFIRST-1
       READ(3,END=9000,ERR=9500)NPROF,(IFLAG(J),J=1,8),
       XLAT, XLON,
      *JPROF,ICLAS,IPAT,IMASS,IPROV,WDEP1,WDEP2,
       IYEAR,IMON,
      *IDAY,XHOUR,IDENT,ISOURC,NCYC1,NCYC2,
      *EXTRA,ICRUZ,(D(J),T(J),J=1,NCYC1+NCYC2)
       ICNT = ICNT+1
100    CONTINUE
       END IF
C*****************************************************************
C  READ DESIRED PROFILES FROM MASTER FILE
C*****************************************************************
     DO 12 I=1,NUMPRO
     READ(3,END=9001,ERR=9500) NPROF,(IFLAG(J),J=1,8),XLAT,
     XLON,
    *JPROF,ICLAS,IPAT,IMASS,IPROV,
    *WDEP1,WDEP2,IYEAR,IMON,IDAY,XHOUR,IDENT,ISOURC,
     NCYC1,
    *NCYC2,EXTRA,ICRUZ,(D(J),T(J),J=1,NCYC1+NCYC2)
     NCYC=NCYC1+NCYC2
     ICNT1 = ICNT1+1
     WRITE(6,*) ('PROFILE '),I
C*************************************************************
C  PROFILE HAS BEEN READ - CHECK FLAG(1) FOR UNWANTED
C  PROFILE
C*************************************************************
```

```
      IF(NPROF.EQ.42.OR.NPROF.EQ.29.OR.NPROF.EQ.315.
     *OR.NPROF.EQ.1234) THEN
       IERR0 = IERR0+1
       GO TO 12
      END IF
C*****************************************************************
C  CHECK TO SEE IF PROFILE EXTENDS TO DEPTH OF DMIN
C*****************************************************************
     IF(D(NCYC).LT.DMIN) THEN
      IERR1 = IERR1+1
      GO TO 12
     END IF
     IKEEP = IKEEP+1
C*****************************************************************
C  INTERPOLATE PROFILE TO D(NCYC) OR DMAX, WHICHEVER
C  WHICHEVER IS GREATEST.
C*****************************************************************
     BOT = DMAX
      IF(D(NCYC).LT.DMAX) BOT=D(NCYC)
      CALL LINR(DELZ,NCYC,0.,BOT,ICT,D,T,D2,T2)
C*****************************************************************
C  CALCULATE GRADIENT
C*****************************************************************
        GMAX = -999.
        GMIN = 999.
        DO 20 L=1,ICT-1
        GRAD(L) = (T2(L+1)-T2(L)) / DELZ
        D3(L) = D2(L) + (DELZ/2.)
        IF(GRAD(L).GT.GMAX) GMAX = GRAD(L)
        IF(GRAD( L).LT.GMIN) GMIN = GRAD(L)
  20   CONTINUE
C       WRITE(6,605) GMIN,GMAX
  605   FORMAT(//,1X,',GMIN,GMAX = ',2F8.4)
C*****************************************************************
C  CALL GRADFIT (GFIT) TO COMPUTE MODELED GRADIENT
C*****************************************************************
       ICT2 = ICT-1
       CALL GFIT(DELZ,ICT2,D3,GRAD,XG,D2,T2,CD,CG)
C*****************************************************************
C  COMPUTE MODELED TEMPERATURE FROM MODELED
    GRADIENT
C*****************************************************************
```

```
      TM(1) = T2(1)
      DO 30 L=2,ICT
 30   TM(L) = XG(L-1) * (D2(L)-D2(L-1)) + TM(L-1)
C*************************************************************
C  COMPUT FINAL RMS ERROR IN TEMPERATURE SPACE
C*************************************************************
   CALL RMS(TM,T2,ICT,R)
C*************************************************************
C  FILL COEFFICIENT ARRAY (ORIGINAL COEFS: DEPTHS AND
C  GRADIENTS)
C*************************************************************
C
      COEF(1) = T2(1)
      COEF(2) = CD(2)
     COEF(3) = CD(3)
     COEF(4) = CD(4)
     COEF(5) = CD(5)
     COEF(6) = CD(6)
     COEF(7) = CG(1)
     COEF(8) = CG(3)
     COEF(9) = CG(5)
     COEF(10) = CG(6)
     COEF(11) = CG(7)
C*************************************************************
C  IF DESRIRED, WRITE COEFFICIENTS TO UNIT 4
C*************************************************************
      IF(IOPT.EQ.'N') GO TO 24
      IF(R.GT.0.13) THEN
      GO TO 24
      END IF
     WRITE(4,610)NPROF,XLAT,XLON,WDEP2,(COEF(K),K=1,11),
     *NCYC1,IMASS
610  FORMAT(I5,2(F8.3,1X),F6.0,1X,F5.2,1X,5(F8.2,1X),5F8.4,I5,I4)
      WRITE(28,615) XLAT,XLON,COEF(1),COEF(2),COEF(3)
615  FORMAT(2(f8.3,2x),f5.2,2x,f8.2,2x,f8.2)
 24  CONTINUE
C
C*************************************************************
C  COMPUTE TEMPS AT DEPTHS OF COARSE GRADIENTS:
   THESE WILL
C  BE THE ACTUAL COEFFICIENTS FOR INTERPOLATION
C*************************************************************
```

```
      X2(1) = T2(1)
      CALL LINR(CD(2),ICT,CD(2),CD(2),ICT4,D2,TM,DOUT,TOUT)
      X2(2) = TOUT
      X2(8) = DOUT
      CALL LINR(CD(3),ICT,CD(3),CD(3),ICT4,D2,TM,DOUT,TOUT)
      X2(3) = TOUT
      X2(9) = DOUT
      CALL LINR(CD(4),ICT,CD(4),CD(4),ICT4,D2,TM,DOUT,TOUT)
      X2(4) = TOUT
      X2(10) = DOUT
      CALL LINR(CD(5),ICT,CD(5),CD(5),ICT4,D2,TM,DOUT,TOUT)
      X2(5) = TOUT
      X2(11) = DOUT
      CALL LINR(CD(6),ICT,CD(6),CD(6),ICT4,D2,TM,DOUT,TOUT)
      X2(6) = TOUT
      X2(12) = DOUT
      X2(7) = TM(ICT)
C ********************************************************
C     IF DESIRED, WRITE INTERPOLATED COEFS TO UNIT 7 AND
C     MASTER FILE PROFILE TO UNIT 8 (TO USE FOR STD
      DEPTH INTERP).
C     PROFILES NOT PASSING RMS TEST ARE WRITTEN TO
      MASTER FILE C 9.
C ********************************************************
      IF(IOPT.EQ.'N') GO TO 25
      IF(R.GT.0.13) THEN
      IRMS = IRMS+1
      WRITE(9) NPROF,(IFLAG(J),J=1,8),XLAT,XLON,
     *  JPROF,ICLAS,IPAT,IMASS,IPROV,
     *  WDEP1,WDEP2,IYEAR,IMON,IDAY,XHOUR,IDENT,ISOURC,
     *  NCYC1,NCYC2,
     *  EXTRA,ICRUZ,(D(J),T(J),J=1,NCYC1+NCYC2)
      GO TO 25
      END IF
C
      WRITE(7,611) NPROF,XLAT,XLON,WDEP2,(X2(K),K=1,12)
  611 FORMAT(I5,2(F8.3,1X),F6.0,7F8.3,5F8.2)
C
      WRITE(8) NPROF,(IFLAG(J),J=1,8),XLAT,XLON,
     * JPROF,ICLAS,IPAT,IMASS,IPROV, WDEP1,WDEP2,IYEAR,
      IMON,
     * IDAY,XHOUR,IDENT,ISOURC,NCYC1,NCYC2,
     * EXTRA,ICRUZ,(D(J),T(J),J=1,NCYC1+NCYC2)
```

```
 25   CONTINUE
C
 12   CONTINUE
      GO TO 9002
C
9000  WRITE(6,650) ICNT
650   FORMAT(1X,'BEGINNING PROFILE NOT FOUND',/,
     *1X,'EOF ENCOUNTERED AFTER PROFILE',I5,'.
      END PROGRAM.')
      GO TO 1000
C
9001  IEND = 1
9002  LAST = IFIRST+ICNT1-1
      WRITE(6,651) ICNT1,IFIRST,LAST
651   FORMAT(1X,I5,' PROFILES READ FROM MASTER FILE',/,
     *1X,'FIRST PROFILE:',I5,5X,'LAST PROFILE:',I5)
      GO TO 1000
C
9500  ITOT = ICNT+ICNT1
      WRITE(6,652) ITOT
652   FORMAT(1X,'READ ERROR AFTER RECORD NUMBER
     *(PROFILE)',I5,/,
     *1X,'*** CHECK MASTER FILE *** PROGRAM TERMINATED.')
C
 1000 CONTINUE
C
      WRITE(6,653) IERR1,IKEEP
653   FORMAT(/,1X,I5,' PROFILES NOT DEEP ENOUGH',/,
     *1X,I5,' PROFILES KEPT FOR PROCESSING')
C
      WRITE(6,654) IRMS
654   FORMAT(/,1X,I5,' PROFILE REJECTED FOR RMS GT 0.13 -
      SEE UNIT
     *9')
      CALL DONEPL
      END
```

C.4 Subroutines

C.4.1 Interpolation

The subroutine *LINR* is used for linear interpolation to fill the gaps in the input data.

```
**************************************************************
**
      SUBROUTINE LINR(DELX,N,XBGN,XEND,ICT,X,Y,AX,AY)
C     DELX=DESIRED INTERPOLATION INTERVAL, X AND Y
      ARE INPUT
C     WITH X=INDEP.
C     VARIABLE, AX AND AY ARE OUTPUT, N=LENGTH OF X,
      XBGN AND
C     XEND=DESIRED
C     BEGINNING AND ENDING VALUES OF AX,ICT=LENGTH OF AX
      DIMENSION X(*),Y(*),AX(*),AY(*)
      NN=N-1
      ICT=((XEND-XBGN)/DELX)+1.0001
      DO 1 J=1,ICT
      A=J-1
    1 AX(J)=A*DELX+XBGN
      KK=1
      DO 2 J=1,NN
      JJ=J+1
      SLOPE=(Y(JJ)-Y(J))/(X(JJ)-X(J))
      DO 3 K=KK,ICT
      I=K
      IF(AX(K).GT.X(JJ).AND.AX(K).LE.X(N)) GO TO 2
    3 AY(K)=SLOPE*(AX(K)-X(J))+Y(J)
      GO TO 4
    2 KK=I
    4 RETURN
      END
```

C.4.2 Iteration

As described in Sect. 2.4.3, the subroutine *BUMP* is used to perform an increment change in depth data; the soubroutine *RMS* is used to compute the root mean square error; and the subroutine *DEPALL* is used to cnduct the iteration.

```
*********************************************************
***
      SUBROUTINE BUMP(CD,LEVEL,DINC,IB,ISIGN,IGO)
C
C ******************************************
C *                                        *
C *       SUBROUTINE BUMP                  *
C *                                        *
C ******************************************
C
C
```

```
C    *** PARAMETERS PASSED TO BUMP
C
C    CD: DEPTH ARRAY (7) FOR COARSE DEPTHS (GRADIENT
      SPACE)
C    LEVEL: DEPTH BEING ADJUSTED IN CALLING PROGRAM
      (2-6)
C    DINC: DEPTH INTERVAL (METERS) FOR ADJUSTMENTS
C    IB: FLAG TO INDICATE WHAT TYPE OF BUMP DESIRED –
C      IB = 0: DONT BUMP ANY DEPTH
C      IB = 1: BUMP ONLY DEPTHS ABOVE (IF ADJ UPWARD)
C      IB = 2: BUMP ONLY DEPTHS BELOW (IF ADJ DOWNWARD)
C      IB = 3: BUMP ANY DEPTH (DEPENDING IF GOING UP OR
       DOWN)
C   ISIGN: FLAG TO INDICATE DIRCTION OF DEPTH ADJUSTMENT
C        (1 = ADDING DEPTHS, -1 = SUBTRACTING DEPTHS)
C
C   *** PARAMETERS PASSED BACK TO CALLING PROGRAM
C
C    CD: DEPTHS DEFINED ABOVE (BUT NOW USUALLY
      DIFFERENT)
C    IGO: FLAG TO TELL WHERE TO GO IN CALLING PROGRAM
C        (0 = CONTINUE, 1 = GO TO END OF CALLING SUBROUTINE)
C
C
     DIMENSION CD(7)
C
     IGO = 0
     L = LEVEL
     F1 = CD(1)
     F7 = CD(7)
C
     IF(ISIGN.EQ.1) GO TO 10
     IF(ISIGN.EQ.-1) GO TO 20
C
C ******************************************************
C CASE FOR ADDING DEPTHS IN CALLING SUBROUTINE
C ******************************************************
C
C DONT BUMP
C
```

```
10 CONTINUE
   IF(IB.EQ.0.OR.IB.EQ.1) THEN
   IF(CD(L).GE.CD(L+1)) THEN
   CD(L) = CD(L) - DINC
   IGO = 1
   END IF
   RETURN
   END IF
C
C  BUMP DEPTHS BELOW
C
     IF(IB.EQ.2.OR.IB.EQ.3) THEN
     LL = L
     DO 15 K=1,7-LL
     IF(CD(L).GE.CD(L+1)) THEN
      CD(L+1) = CD(L+1) + DINC
      L = L+1
     END IF
15   CONTINUE
C
C    IF BOTTOM DEPTH WAS BUMPED, MOVE ALL NECESSARY DEPTHS
C    UP
C
     IF(CD(7).GT.F7) THEN
      L = 7
      DO 17 K=1,7-LL
      CD(L) = CD(L) - DINC
        L = L-1
17      CONTINUE
        IGO = 1
        END IF
        RETURN
        END IF
C
C ****************************************************************
C CASE FOR SUBTRACTING DEPTHS IN CALLING SUBROUTINE
C ****************************************************************
C
```

```
C    DONT BUMP
C
    20 CONTINUE
       IF(IB.EQ.0.OR.IB.EQ.2) THEN
       IF(CD(L).LE.CD(L-1)) THEN
       CD(L) = CD(L) + DINC
       IGO = 1
       END IF
       RETURN
       END IF
C
C    BUMP DEPTHS ABOVE
C
       IF(IB.EQ.1.OR.IB.EQ.3) THEN
       LL = L
       DO 25 K=1,LL-1
       IF(CD(L).LE.CD(L-1)) THEN
       CD(L-1) = CD(L-1) - DINC
       pcL = L-1
       END IF
    25 CONTINUE
C
C    IF TOP DEPTH WAS BUMPED, MOVE ALL NECESSARY
     DEPTHS
C    BACK DOWN
C
  26 IF(CD(1).LT.F1) THEN
     L = 1
     DO 27 K=1,LL
     CD(L) = CD(L) + DINC
     L = L+1
27   CONTINUE
     IGO = 1
     GO TO 26
     END IF
     RETURN
     END IF
C
     END
  ************************************************
     SUBROUTINE RMS(XG,GRAD,ICT2,R)
     DIMENSION XG(*),GRAD(*)
     R = 0
     DO 10 J=1,ICT2
10   R = R + (XG(J)-GRAD(J))**2.
     RT = XG(J)-GRAD(J)
     R = (R/ICT2)**0.5
     RETURN
     END
```

```
**************************************************************

   SUBROUTINE
   *DEPALL(DELX,SUR,BOT,ICT2,CD,CG,D3,GRAD,R,TOL,NI)
C
C ***********************************************************
C SUBROUTINE DEPALL
C MAKE INITIAL ADJUSTMENT OF D1,D2,D3,D4 (CD(2) - CD(5))
C ALSO ADJUST CD(6) AFTER ABOVE DEPTHS ARE ADJUSTED
C ***********************************************************
C
C
      DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400)
      DIMENSION RR(400)
C
       IKEEP = 0
C      WRITE(6,606) (CD(J),J=1,7)
 606 FORMAT(1X,'INITIAL CDS: ',7F6.1)
C
C  CHECK FIRST TO SEE WHETHER TO ADD OR SUBTRACT
    DEPTHS
C
       ICNT = 0
       RMIN = 999.
       DINC = DELX
       R1 = R
C
C  COMPUTE DIFFERENCE BETWEEN CD(2) AND CD(1) AND
C  SUBTRACT
C  THIS FROM CD(2) THROUGH CD(5) FOR STARTING DEPTHS
C
       FIRST = CD(2)
       DIFF = CD(2) - CD(1)
       DO 5 K=2,5
    5 CD(K) = CD(K) - DIFF + DINC
C
C  FILL MODELED GRADIENT ARRAY WITH LINR USING NEW
    DEPTHS
C
       DO 80 M=1,ICT2
       CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
       CALL RMS(XG,GRAD,ICT2,R)
       ICNT = ICNT+1
C
       RR(ICNT) = R
       IF(RR(M).LE.RMIN) THEN
       IKEEP = ICNT
       RMIN = RR(M)
       END IF
```

```
C
      DO 6 K=2,5
    6 CD(K) = CD(K) + DINC
      IF(CD(5).GE.CD(6)) GO TO 85
   80 CONTINUE
C
C
   85 CONTINUE
      NEWCD2 = (IKEEP-1)*DINC
C
C ADJUST CD(2) THROUGH CD(5) TO OPTIMUM DEPTHS
C
      DIFF = CD(2) - NEWCD2
      DO 15 K=2,5
   15 CD(K) = CD(K) - DIFF + DINC
C
C COMPUTE INITIAL DEPTH FOR CD(6): 1/4 OF THE
C  DISTANCE BETWEEN
C CD(5) AND CD(7).
C
     DIFF2 = CD(6) - CD(5)
     CD(6) = CD(5) + 0.25*DIFF2
     CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
C
C     WRITE(6,604) (CD(J),J=1,7)
  604 FORMAT(1X,'NEW CDS: ',7F6.1)
      ADJ = FIRST - CD(2)
C     WRITE(6,605) ICNT,ADJ,R
  605 FORMAT(1X,' DEPALL ITNS:',I3,4X,'ADJ: ',F6.0,3X,
     *'RMS = ',F7.4)
      RETURN
      END
```

C.4.3 Mixed Layer Depth

The subroutine *DEP1* is used to determine the mixed layer depth.

```
   ************************************************************
      SUBROUTINE DEP1(DELX,SUR,BOT,ICT2,CD,CG,D3,
     *GRAD,R,TOL,NI,IB,IDUM)
C
C ************************************************************
C
C     SUBROUTINE DEP1
C     ADJUST D1 (CD(2)) - DEPTH OF MIXED LAYER
C
```

```
C     ************************************************
C
        DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400)
C
C       CHECK FIRST TO SEE WHETHER TO ADD OR
        SUBTRACT
C       DEPTHS
C
        ICNT = 0
        FIRST = CD(2)
        DINC = DELX
        R1 = R
        CD(2) = CD(2) + DINC
        CALL BUMP(CD,2,DINC,IB,1,IGO)
        IF(IGO.EQ.1) GO TO 199
C
C       FILL MODELED GRADIENT ARRAY WITH LINR USING
C       NEW DEPTHS
C
        CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
        CALL RMS(XG,GRAD,ICT2,R)
        ICNT = ICNT+1
C
        IF(ABS(R-R1).LT.TOL) GO TO 199
c
c       IF(R.LT.R1) GO TO 100    @ GOING RIGHT WAY
c       IF(R.GT.R1) THEN    @ GOING WRONG WAY
c
        IF(R.LT.R1) GO TO 100
        IF(R.GT.R1) THEN
c
        CD(2) = CD(2) - DINC
        R = R1
        GO TO 150
        END IF
C
C       ADJUST D1 BY ADDING DEPTHS
C
  100   IF = 0
  102   IF(ICNT.GT.NI) GO TO 199
        R1 = R
        CD(2) = CD(2) + DINC
        CALL BUMP(CD,2,DINC,IB,1,IGO)
        IF(IGO.EQ.1) GO TO 199
C
        CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
        CALL RMS(XG,GRAD,ICT2,R)
        ICNT = ICNT+1
```

```
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
      IF(R.LT.R1) GO TO 102
      IF(R.GT.R1.AND.IF.NE.4) THEN
      IF = IF+1
      CD(2) = CD(2) - DINC
      DINC = DINC/2.
      R = R1
      GO TO 102
      END IF
C
      GO TO 199
C
C     ADJUST D1 BY SUBSTRACTING DEPTHS
C
 150 IF = 0
 152 IF(ICNT.GT.NI) GO TO 199
      R1 = R
      CD(2) = CD(2) - DINC
      CALL BUMP(CD,2,DINC,IB,-1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
      IF(R.LT.R1) GO TO 152
      IF(R.GT.R1.AND.IF.NE.4) THEN
      IF = IF+1
      CD(2) = CD(2) + DINC
      DINC = DINC/2.
      R = R1
      GO TO 152
      END IF
C
      GO TO 199
C
 199 CONTINUE
C
      ADJ = FIRST - CD(2)
C     WRITE(6,605) ICNT,ADJ,R
 605 FORMAT(1X,'D1 ITERATIONS: ',I3,4X,'ADJ: ',F6.0,3X,
     *'RMS = ',F7.4)
C
C
      RETURN
      END
```

C.4.4 Depth at the Top of Thermocline/Halocline

The subroutine DEP2 is used to determinine the depth at the top of thermocline (or halocline).

```
*******************************************************
      SUBROUTINE DEP2(DELX,SUR,BOT,ICT2,CD,CG,D3,GRAD,R,
      TOL,NI,*IB,IF)
C
C *******************************************************
C SUBROUTINE DEP2                                       *
C ADJUST D2 (CD(3)) - DEPTH OF TOP OF THERMOCLINE       *
C                                                       *
C *******************************************************
C
    DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400)
C
C   CHECK FIRST TO SEE WHETHER TO ADD OR
    SUBTRACT DEPTHS
C
    ICNT = 0
    FIRST = CD(3)
    DINC = DELX
    R1 = R
    CD(3) = CD(3) + DINC
    CALL BUMP(CD,3,DINC,IB,1,IGO)
    IF(IGO.EQ.1) GO TO 199
C
C FILL MODELED GRADIENT ARRAY WITH LINR USING
  NEW DEPTHS
C
    CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
    CALL RMS(XG,GRAD,ICT2,R)
    ICNT = ICNT+1
C
    IF(ABS(R-R1).LT.TOL) GO TO 199
c
c   IF(R.LT.R1) GO TO 100      @ GOING RIGHT WAY
c   IF(R.GT.R1) THEN      @ GOING WRONG WAY
    IF(R.LT.R1) GO TO 100
    IF(R.GT.R1) THEN
c
    CD(3) = CD(3) - DINC
    R = R1
    GO TO 150
    END IF
```

```
C
C    ADJUST D2 BY ADDING DEPTHS
C
 100 IF = 0
 102 IF(ICNT.GT.NI) GO TO 199
     R1 = R
     CD(3) = CD(3) + DINC
     CALL BUMP(CD,3,DINC,IB,1,IGO)
     IF(IGO.EQ.1) GO TO 199
C
     CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
     CALL RMS(XG,GRAD,ICT2,R)
     ICNT = ICNT+1
C
     IF(ABS(R-R1).LT.TOL) GO TO 199
     IF(R.LT.R1) GO TO 102
     IF(R.GT.R1.AND.IF.NE.4) THEN
      IF = IF+1
      CD(3) = CD(3) - DINC
      DINC = DINC/2.
      R = R1
      GO TO 102
     END IF
C
     GO TO 199
C
C ADJUST D2 BY SUBSTRACTING DEPTHS
C
 150 IF = 0
 152 IF(ICNT.GT.NI) GO TO 199
      R1 = R
      CD(3) = CD(3) - DINC
      CALL BUMP(CD,3,DINC,IB,-1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
     IF(ABS(R-R1).LT.TOL) GO TO 199
     IF(R.LT.R1) GO TO 152
     IF(R.GT.R1.AND.IF.NE.4) THEN
```

```
      IF = IF+1
      CD(3) = CD(3) + DINC
      DINC = DINC/2.
      R = R1
      GO TO 152
      END IF
C
      GO TO 199
C
  199 CONTINUE
C
      ADJ = FIRST - CD(3)
C     WRITE(6,605) ICNT,ADJ,R
  605 FORMAT(1X,'D2 ITERATIONS: ',I3,4X,'ADJ: ',F6.0,3X,
     *'RMS = ',F7.4)
C
    RETURN
    END
```

C.4.5 Depth at the Bottom of Thermocline/Halocline

The subroutine DEP3 is used to determinine the depth at the bottom of thermocline (or halocline).

```
*******************************************************
    SUBROUTINE DEP3(DELX,SUR,BOT,ICT2,CD,CG,D3,GRAD,R,
    TOL,NI,*IB,IF)
C
C ************************************************************
C SUBROUTINE DEP3                                            *
C ADJUST D3 (CD(4)) - DEPTH OF BOTTOM OF THERMOCLINE         *
C OR HALOCLINE                                               *
C ************************************************************
C
    DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400)
C
C   CHECK FIRST TO SEE WHETHER TO ADD OR SUBTRACT
    DEPTHS
C
    ICNT = 0
    FIRST = CD(4)
    DINC = DELX
```

```
      R1 = R
      CD(4) = CD(4) + DINC
      CALL BUMP(CD,4,DINC,IB,1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
C     FILL MODELED GRADIENT ARRAY WITH LINR USING
      NEW DEPTHS
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
c
c       IF(R.LT.R1) GO TO 100    @ GOING RIGHT WAY
c       IF(R.GT.R1) THEN    @ GOING WRONG WAY
        IF(R.LT.R1) GO TO 100
        IF(R.GT.R1) THEN
c
        CD(4) = CD(4) - DINC
        R = R1
        GO TO 150
        END IF
C
C     ADJUST D3 BY ADDING DEPTHS
C
 100   IF = 0
 102   IF(ICNT.GT.NI) GO TO 199
       R1 = R
      CD(4) = CD(4) + DINC
      CALL BUMP(CD,4,DINC,IB,1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
      IF(R.LT.R1) GO TO 102
      IF(R.GT.R1.AND.IF.NE.10) THEN
       IF = IF+1
       CD(4) = CD(4) - DINC
       DINC = DINC/2.
       R = R1
       GO TO 102
     END IF
```

```
C
    GO TO 199
C
C ADJUST D3 BY SUBSTRACTING DEPTHS
C
 150 IF = 0
 152 IF(ICNT.GT.NI) GO TO 199
      R1 = R
     CD(4) = CD(4) - DINC
     CALL BUMP(CD,4,DINC,IB,-1,IGO)
     IF(IGO.EQ.1) GO TO 199
C
    CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
    CALL RMS(XG,GRAD,ICT2,R)
    ICNT = ICNT+1
C
    IF(ABS(R-R1).LT.TOL) GO TO 199
    IF(R.LT.R1) GO TO 152
    IF(R.GT.R1.AND.IF.NE.10) THEN
     IF = IF+1
     CD(4) = CD(4) + DINC
     DINC = DINC/2.
     R = R1
     GO TO 152
     END IF
C
    GO TO 199
C
 199 CONTINUE
C
     ADJ = FIRST - CD(4)
     WRITE(6,605) ICNT,ADJ,R
605  FORMAT(1X,'D3 INTERATIONS: ',I3,4X,'ADJ: ',F6.0,3X,
  *'RMS = ',F7.4)
C
   RETURN
   END
```

C.4.6 Depth at Top of Near-Zero Gradient Below Thermocline

```
   SUBROUTINE DEP4(DELX,SUR,BOT,ICT2,CD,CG,D3,GRAD,R,
   TOL,NI,*IB,IF,IR)
C
C **********
```

```
C
C  VALUE OF IR DETERMINES INTERVAL OVER WHICH RMS IS
C  COMPUTED.
C
C ***********
C
C
      DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400)
      KEEP = BOT
C
      IF(IR.EQ.1) THEN
       BOT = CD(5)
       CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
       CALL RMS(XG,GRAD,ICT,R)
      END IF
C
C  CHECK FIRST TO SEE WHETHER TO ADD OR SUBTRACT
     DEPTHS
C
      ICNT = 0
      FIRST = CD(5)
      DINC = DELX
      R1 = R
      CD(5) = CD(5) + DINC
      CALL BUMP(CD,5,DINC,IB,1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
C  FILL MODELED GRADIENT ARRAY WITH LINR USING
     NEW DEPTHS
C
      IF(IR.EQ.1) BOT = CD(5)
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
C
C       IF(R.LT.R1) GO TO 100       @ GOING RIGHT WAY
C       IF(R.GT.R1) GO TO 150       @ GOING WRONG WAY
        IF(R.LT.R1) GO TO 100
        IF(R.GT.R1) GO TO 150

C  ADJUST D4 BY ADDING DEPTHS
C
  100  IF = 0
  102  IF(ICNT.GT.NI) GO TO 199
       R1 = R
       CD(5) = CD(5) + DINC
```

```
      CALL BUMP(CD,5,DINC,IB,1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
      IF(IR.EQ.1) BOT = CD(5)
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
      IF(R.LT.R1) GO TO 102
      IF(R.GT.R1.AND.IF.NE.4) THEN
        IF = IF+1
        CD(5) = CD(5) - DINC
        DINC = DINC/2.
        GO TO 102
      END IF
C
      GO TO 199
C
C   ADJUST D4 BY SUBSTRACTING DEPTHS
C
  150 IF = 0
  152 IF(ICNT.GT.NI) GO TO 199
        R1 = R
      CD(5) = CD(5) - DINC
      CALL BUMP(CD,5,DINC,IB,-1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
      IF(IR.EQ.1) BOT = CD(5)
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
      IF(R.LT.R1) GO TO 152
      IF(R.GT.R1.AND.IF.NE.4) THEN
        IF = IF+1
        CD(5) = CD(5) + DINC
        DINC = DINC/2.
        GO TO 152
      END IF
C
        GO TO 199
C
  199 CONTINUE
C
        ADJ = FIRST - CD(5)
```

```
C     WRITE(6,605) ICNT,ADJ,R
 605 FORMAT(1X,'D4 INTERATIONS: ',I3,4X,'ADJ: ',F6.0,3X,
    *'RMS = ',F7.4)
C
C    IF IR FLAG = 1, CALL LINR ONCE MORE WITH TRUE VALUE FOR
C    BOTTOM
C
     IF(IR.EQ.1) THEN
      BOT = KEEP
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
     END IF
C
     RETURN
     END
*******************************************************
   SUBROUTINE DEP4T(DELX,SUR,BOT,ICT2,CD,CG,D3,GRAD,R,
  *TOL3,NI,D2,T2)
C
C**********
C
C ADJUST D4 (CD(5)) - DEPTH OF TOP OF NEAR-ZERO
   GRADIENT BELOW THRMCLN
C USE TEMP FOR RMS
C
C***********
C
C
   DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400),
  *D2(401),T2(401),TM(401)
C
CCALCULATE INITIAL RMS IN TEMP SPACE
C
   ICT = ICT2+1
   CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
   TM(1) = T2(1)
   DO 29 L=2,ICT
 29 TM(L) = XG(L-1) * (D2(L) - D2(L-1)) + TM(L-1)
   CALL RMS(TM,T2,ICT,R)
C
CCHECK FIRST TO SEE WHETHER TO ADD OR SUBTRACT
   DEPTHS
C
```

```
      ICNT = 0
      FIRST = CD(5)
      DINC = DELX
      R1 = R
      CD(5) = CD(5) + DINC
      IF(CD(5).GE.CD(6)) THEN
      CD(5) = CD(5) - DINC
      GO TO 199
      END IF
C
C     FILL MODELED GRADIENT ARRAY WITH LINR USING
      NEW DEPTHS
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      TM(1) = T2(1)
      DO 30 L=2,ICT
   30 TM(L) = XG(L-1) * (D2(L) - D2(L-1)) + TM(L-1)
      CALL RMS(TM,T2,ICT,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL3) GO TO 199
C
C      IF(R.LT.R1) GO TO 100       @ GOING RIGHT WAY
C      IF(R.GT.R1) GO TO 150       @ GOING WRONG WAY
       IF(R.LT.R1) GO TO 100
       IF(R.GT.R1) GO TO 150
C
C     ADJUST D4 BY ADDING DEPTHS
C
   100 IF = 0
   102 IF(ICNT.GT.NI) GO TO 199
         R1 = R
         CD(5) = CD(5) + DINC
         IF(CD(5).GE.CD(6)) THEN
        CD(5) = CD(5) - DINC
        GO TO 199
        END IF
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      TM(1) = T2(1)
      DO 31 L=2,ICT
   31 TM(L) = XG(L-1) * (D2(L) - D2(L-1)) + TM(L-1)
      CALL RMS(TM,T2,ICT,R)
      ICNT = ICNT+1
```

```
C
    IF(ABS(R-R1).LT.TOL3) GO TO 199
    IF(R.LT.R1) GO TO 102
    IF(R.GT.R1.AND.IF.NE.4) THEN
    IF = IF+1
    CD(5) = CD(5) - DINC
    DINC = DINC/2.
    GO TO 102
   END IF
C
   GO TO 199
C
C ADJUST D4 BY SUBSTRACTING DEPTHS
C
  150 IF = 0
  152 IF(ICNT.GT.NI) GO TO 199
       R1 = R
       CD(5) = CD(5) - DINC
       IF(CD(5).LE.CD(4)) THEN
       CD(5) = CD(5) + DINC
      GO TO 199
     END IF
C
    CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
    TM(1) = T2(1)
    DO 32 L=2,ICT
 32 TM(L) = XG(L-1) * (D2(L) - D2(L-1)) + TM(L-1)
    CALL RMS(TM,T2,ICT,R)
    ICNT = ICNT+1
C
       IF(ABS(R-R1).LT.TOL3) GO TO 199
       IF(R.LT.R1) GO TO 152
       IF(R.GT.R1.AND.IF.NE.4) THEN
       IF = IF+1
       CD(5) = CD(5) + DINC
      DINC = DINC/2.
      GO TO 152
      END IF
C
       GO TO 199
C
 199 CONTINUE
C
       ADJ = FIRST - CD(5)
C     WRITE(6,605) ICNT,ADJ,R
  605 FORMAT(1X,'D4T INTERATIONS: ',I3,4X,'ADJ: ',F6.0,3X,
     *'RMS = ',F7.4)
C
     RETURN
     END
```

C.4.7 Depth between Two Last Legs of Profile

```
      SUBROUTINE DEP5(DELX,SUR,BOT,ICT2,CD,CG,D3,GRAD,R,TOL,NI,
     *IB,IF)
C
C **********
C
C ADJUST D5 (CD(6)) - DEPTH BETWEEN LAST TWO LEGS OF PROFILE
C
C ***********
C
C
      DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400)
C
C CHECK FIRST TO SEE WHETHER TO ADD OR SUBTRACT DEPTHS
C
      ICNT = 0
      FIRST = CD(6)
      DINC = DELX
      R1 = R
      CD(6) = CD(6) + DINC
      CALL BUMP(CD,6,DINC,IB,1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
C FILL MODELED GRADIENT ARRAY WITH LINR USING NEW DEPTHS
C
        CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
        CALL RMS(XG,GRAD,ICT2,R)
        ICNT = ICNT+1
C
        IF(ABS(R-R1).LT.TOL) GO TO 199
C
C       IF(R.LT.R1) GO TO 100    @ GOING RIGHT WAY
C       IF(R.GT.R1) THEN    @ GOING WRONG WAY
        IF(R.LT.R1) GO TO 100
        IF(R.GT.R1) THEN
C
        CD(6) = CD(6) - DINC
        R = R1
       GO TO 150
       END IF
C
C ADJUST D4 BY ADDING DEPTHS
C
  100 IF = 0
```

```
  102 IF(ICNT.GT.NI) GO TO 199
      R1 = R
      CD(6) = CD(6) + DINC
      CALL BUMP(CD,6,DINC,IB,1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
      IF(R.LT.R1) GO TO 102
      IF(R.GT.R1.AND.IF.NE.10) THEN
      IF = IF+1
      CD(6) = CD(6) - DINC
      DINC = DINC/2.
      R = R1
      GO TO 102
      END IF

C
      GO TO 199
C
C ADJUST D4 BY SUBSTRACTING DEPTHS
C
  150 IF = 0
  152 IF(ICNT.GT.NI) GO TO 199
      R1 = R
      CD(6) = CD(6) - DINC
      CALL BUMP(CD,6,DINC,IB,-1,IGO)
      IF(IGO.EQ.1) GO TO 199
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      CALL RMS(XG,GRAD,ICT2,R)
      ICNT = ICNT+1
C
      IF(ABS(R-R1).LT.TOL) GO TO 199
      IF(R.LT.R1) GO TO 152
      IF(R.GT.R1.AND.IF.NE.10) THEN
      IF = IF+1
      CD(6) = CD(6) + DINC
      DINC = DINC/2.
      R = R1
      GO TO 152
      END IF
```

```
C
      GO TO 199
C
 199 CONTINUE
C
      ADJ = FIRST - CD(6)
C     WRITE(6,605) ICNT,ADJ,R
 605  FORMAT(1X,'D4 INTERATIONS: ',I3,4X,'ADJ: ',F6.0,3X,
     *'RMS = ',F7.4)
C
      RETURN
      END
```

C.4.8 Adjustment of Bottom Gradient

```
   SUBROUTINE
   *GRAD3A(DELX,SUR,BOT,ICT2,CD,CG,D3,GRAD,R,TOL2,NI)
C
C *********
C
C SUBROUTINE GRAD3A
C ADJUST G3A - GRADIENT AT BOTTOM
C
C *********
C
   DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400)
C
C CHECK FIRST TO SEE WHETHER TO ADD OR SUBTRACT
   GRADIENT
C
   FIRST = CG(5)
   GINC = FIRST/25.
   IF(GINC.LT.0.005) GINC = .005
   ICNT = 0
   R1 = R
   CG(5) = CG(5) + GINC
C
C  FILL MODELED GRADIENT ARRAY WITH LINR USING NEW
C  GRADIENT
C
    CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
    CALL RMS(XG,GRAD,ICT2,R)
    ICNT = ICNT+1
C
    IF(ABS(R-R1).LT.TOL2) GO TO 199
```

```
C
C   IF(R.LT.R1) GO TO 100 @      GOING RIGHT WAY
C   IF(R.GT.R1) THEN @      GOING WRONG WAY
      IF(R.LT.R1) GO TO 100
      IF(R.GT.R1) THEN
C
      CG(5) = CG(5) - GINC
      R = R1
      GO TO 150
      END IF

C
C ADJUST GRADIENT G3A BY ADDING
C
  100 IF = 0
  102 IF(ICNT.GT.NI) GO TO 199
        R1 = R
        CG(5) = CG(5) + GINC
C
        CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
        CALL RMS(XG,GRAD,ICT2,R)
        ICNT = ICNT+1
C
        IF(ABS(R-R1).LT.TOL2) GO TO 199
        IF(R.LT.R1) GO TO 102
        IF(R.GT.R1.AND.IF.NE.10) THEN
        IF = IF+1
        CG(5) = CG(5) - GINC
        GINC = GINC/2.
        R = R1
        GO TO 102
        END IF
C
C  GO TO 199
C
C  ADJUST GRADIENT G3A BY SUBTRACTING
C
  150 IF = 0
  152 IF(ICNT.GT.NI) GO TO 199
        R1 = R
        CG(5) = CG(5) - GINC
C
        CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
        CALL RMS(XG,GRAD,ICT2,R)
        ICNT = ICNT+1
```

```
C
      IF(ABS(R-R1).LT.TOL2) GO TO 199
      IF(R.LT.R1) GO TO 152
      IF(R.GT.R1.AND.IF.NE.10) THEN
      IF = IF+1
     CG(5) = CG(5) + GINC
      GINC = GINC/2.
      R = R1
      GO TO 152
      END IF
C
      GO TO 199
C
 199 CONTINUE
C
      ADJ = FIRST - CG(5)
C     WRITE(6,605) ICNT,ADJ,R
 605 FORMAT(1X,'GRAD3A ITERATIONS: ',I3,4X,'ADJ: ',F6.0,3X,
     *'RMS = ',F7.4)
C
     RETURN
     END

**************************************************
   SUBROUTINE
   *G3AT(DELX,SUR,BOT,ICT2,CD,CG,D3,GRAD,R,TOL3,NI,D2,T2)
C
C  *********
C
C  SUBROUTINE G3AT
C  ADJUST G3A - GRADIENT AT BOTTOM
C  DO RMS WITH TEMP INSTEDAD OF GRADIENT
C
C  *********
C
      DIMENSION D3(400),GRAD(400),CD(*),CG(*),XG(400),
      *D2(401),T2(401),TM(401)
C
C  CHECK FIRST TO SEE WHETHER TO ADD OR SUBTRACT
    GRADIENT
C
      ICT = ICT2+1
      GINC = CG(5)/25.
      IF(GINC.LT.0.0005) GINC = .0005
      FIRST = CG(5)
```

```
      ICNT = 0
      CG(5) = CG(5) + GINC
C
C     FILL MODELED GRADIENT ARRAY WITH LINR USING NEW
C     GRADIENT
C
      CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
      TM(1) = T2(1)
      DO 30 L=2,ICT
   30 TM(L) = XG(L-1) * (D2(L) - D2(L-1)) + TM(L-1)
      CALL RMS(TM,T2,ICT,R)
      ICNT = ICNT+1
       R1 = R
C
     IF(ABS(R-R1).LT.TOL3) GO TO 199
C
C      IF(R.LT.R1) GO TO 100 @ GOING RIGHT WAY
C      IF(R.GT.R1) THEN @ GOING WRONG WAY
     if(r.lt.r1) go to 100
     if(r.gt.r1) then
c
        CG(5) = CG(5) - GINC
        R = R1
        GO TO 150
     END IF
C
C  ADJUST GRADIENT G3A BY ADDING
C
   100 IF = 0
   102 IF(ICNT.GT.NI) GO TO 199
     R1 = R
     CG(5) = CG(5) + GINC
C
     CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
     TM(1) = T2(1)
     DO 31 L=2,ICT
   31 TM(L) = XG(L-1) * (D2(L) - D2(L-1)) + TM(L-1)
     CALL RMS(TM,T2,ICT,R)
     ICNT = ICNT+1
C
     IF(ABS(R-R1).LT.TOL3) GO TO 199
     IF(R.LT.R1) GO TO 102
     IF(R.GT.R1.AND.IF.NE.6) THEN
      IF = IF+1
      CG(5) = CG(5) - GINC
      GINC = GINC/2.
```

```
      R = R1
      GO TO 102
    END IF
C
    GO TO 199
C
C  ADJUST GRADIENT G3A BY SUBTRACTING
C
  150 IF = 0
  152 IF(ICNT.GT.NI) GO TO 199
    R1 = R
    CG(5) = CG(5) - GINC
C
    CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
    TM(1) = T2(1)
    DO 32 L=2,ICT
  32 TM(L) = XG(L-1) * (D2(L) - D2(L-1)) + TM(L-1)
    CALL RMS(TM,T2,ICT,R)
    ICNT = ICNT+1
C
    IF(ABS(R-R1).LT.TOL3) GO TO 199
    IF(R.LT.R1) GO TO 152
    IF(R.GT.R1.AND.IF.NE.6) THEN
      IF = IF+1
      CG(5) = CG(5) + GINC
      GINC = GINC/2.
      R = R1
      GO TO 152
    END IF
C
    GO TO 199
C
  199 CONTINUE
C
    CALL LINR(DELX,7,SUR,BOT,ICT2,CD,CG,D3,XG)
    TM(1) = T2(1)
    DO 33 L=2,ICT
  33 TM(L) = XG(L-1) * (D2(L) - D2(L-1)) + TM(L-1)
    CALL RMS(TM,T2,ICT,R)
C
    ADJ = FIRST - CG(5)
C     WRITE(6,605) ICNT,ADJ,R
  605 FORMAT(1X,'G3AT ITERATIONS: ',I3,4X,'ADJ: ',F6.4,3X,
     *'RMS = ',F7.4)
C
    RETURN
    END
```

C.4.9 Data Input/Output

```
      SUBROUTINE INOUT4
      CHARACTER*25 FOUT4,FOUT5,FOUT8,FOUT9,FIN3,allcoef
      PRINT *,'ENTER FILE NAME FOR COEF FILE (for unit 4)'
      READ(5,99)FOUT4
      print *,'Enter file name for MATLAB use (unit 28)'
      read(5,99)allcoef
      PRINT *,'ENTER FILE NAME FOR RECONSTRUCTED COEF
     (for unit 7)'
      READ(5,99)FOUT5
      PRINT *,'ENTER FILE NAME FOR INPUT MASTER FILE'
      READ(5,99) FIN3
      PRINT *,'ENTER FILE NAME FOR ACCEPTED PROFILES'
      READ(5,99) FOUT8
      PRINT *,'ENTER FILE NAME FOR REJECT FILE'
      READ(5,99) FOUT9
   99 FORMAT(A25)
      OPEN(4,FILE=FOUT4,FORM='FORMATTED',STATUS=
     'UNKNOWN')
      OPEN(28,file=allcoef,form='formatted',status='unknown')
      OPEN(7,FILE=FOUT5,FORM='FORMATTED',STATUS=
     'UNKNOWN')
      OPEN(3,FILE=FIN3,FORM='UNFORMATTED',STATUS='OLD')
      OPEN(8,FILE=FOUT8,FORM='UNFORMATTED',STATUS=
     'UNKNOWN')
      OPEN(9,FILE=FOUT9,FORM='UNFORMATTED',STATUS=
     'UNKNOWN')
      RETURN
      END
```

D

Autocorrelation Matrix Calculated from (T, S) Profile Data

```
C PROGRAM corre.f (f77 corre.f -o corre; Running See corre.com)
C CALCULATE AUTOCORELATION MATRICES FROM
   PROFILE DATA.
C JAPAN/EAST SEA EXAMPLE
C Longitude 127.5E–142E(mx=(142-127.5)/(15/60)+1= 59)
C Latitude 32.5N–50N(my=( 50- 32.5)/(15/60)+1= 71)
C*******************************************************
         parameter (maxpts =141330 , mx=59,my=71,Lv=28)
         real covar(64,64), var1(64,64), var2(64,64)
         integer num(64,64)
         real corr(64,64)
         character obsfile*80,infile*80,ofil1*80,ofil2*80,ofil3*80
         dimension itimlg(64), idstlg(64),tobs(200),tmean(mx,my,Lv)
         dimension anomly(maxpts,1) ,tlmean(mx,my)
         common /LL/dlat(my),dlon(mx),grid1
         common // zlat(maxpts,1), zlon(maxpts,1), time(maxpts,1),
   1     lagdd(maxpts) , lagtt(maxpts),
   2     zlati,zloni,timei,jpt,numpts(64),incdlg,inctlg,level
         crt=-50.0
         level = 1
         iloop = 0
C READ Parameter and FILE NAME
        print *,'Enter Layer-num: nk(1 or 2...or 28)'
        read(*,*) nk
c       print *,'obs.point filename'
         read(*,*)obsfile
c       print *,'grad.point filename'
        read(*,*)infile
        read(*,*)ofil1
        read(*,*)ofil2
        read(*,*)ofil3
```

```
C  OPEN THE INPUT FILE
       open(3,FILE=obsfile,form='formatted',STATUS='old')
       open(4,FILE=infile,form='formatted',STATUS='old')
       open(15,FILE='corrin.dat',form='formatted',STATUS='old')
C OPEN THE OUTPUT FILE.
            n=0
       open(16,FILE=ofil1,form='formatted', STATUS='unknown')
       open(17,file=ofil2,form='formatted', STATUS='unknown')
       open(18,file=ofil3,form='formatted', STATUS='unknown')
C READ IN THE MAXIMUM TIME(day)/DISTANCE(km)
LAGS AND THE LAG INCREMENT.
C READ IN THE LAT/LON BOUNDS OF THE AREA.
C        mxtlag- maximum TIME(days) difference allowed column
C        inctlg- time difference bin size
C        mxdlag- maximum DISTANCE(km) difference allowed rows
C        incdlag- distance difference bin size
         read(15,*) mxtlag,inctlg,mxdlag,incdlg,ntb,ndb
C        ex.input 25(day) 1(day) 250(km) 10(km) from file corrin.dat
         numtbn = mxtlag / inctlg + 1
         if ( numtbn .gt. ntb) stop ' more than ntb time bins'
         numdbn = mxdlag / incdlg + 1
         if ( numdbn .gt. ndb) stop ' more than ndb distance bins'
         write(16,'(6i6)') mxtlag,inctlg,mxdlag,incdlg,numtbn,numdbn
C     READ IN TEMPERATURE PROFILE DATA FOR THE MEAN
         write(16,*)'input grad point filename',infile
         grid=15.0
         grid1=grid/60.
    do   100 j=1,my
    do   101 i=1,mx
         read(4,*) dlon(i),xla,(tmean(i,my-j+1,k),k=1,28)
 101     continue
         dlat(my-j+1)=xla
 100     continue
         write(16,120)dlon(1),dlon(mx),dlat(1),dlat(my),grid1,mx,my
         write(6,120)dlon(1),dlon(mx),dlat(1),dlat(my),grid1,mx,my
 120     format(' lonw lone lats latn grid mx my'/,5f7.2,2i4)
         Idepth = 1
C        LOOP TO STORE TIME, PLACE, AND TEMPERATURE
         ANOMALY
C        VALUE.
C        Reads in the data from tbin as formatted in it.
         write(16,*)' input obser. point filename',obsfile
C        zero mean calc arrays
         do level=1,1
           numpts(level) = 0
```

```
      do ix=1,mx
       do iy=1,my
          lmean(ix,iy)= tmean(ix,iy,nk)
        Enddo
      Enddo
10    CALL READER(rlon,rlat,tobs,ILOOP,iyr,im,id,obsfile)
        if (iloop .eq. 0) go to 11
        CALL LMEAN(tm,rlon,rlat,tlmean,crt)
C       DETERMINE JULIAN DAY AND REAL-VALUED TIME
C       OF TEMPERATURE VALUE.
C       Type II = jday //// Type III = jday+(iyr*365)
        call juldat(iyr,im,id,jday)
        tmptim = jday
        write(16,165) rlon,rlat,tmptim
 165       format(' rlon,rlat= ',2f9.3,' tmptim =',f12.2)
C       TAKE THE OBSERVED TEMPERATURE (tobs) AND
        SUBTRACT
C       THE MEAN
C       TEMPERATURE(tmean) OUT TO GET THE ANOMOLY
        (anomly)
C       STORE THE POSITION, TIME, AND VALUE.
        if( tobs(1) .lt. 900. .and. tm .gt. crt) then
        numpts(level) = numpts(level) + 1
          n = numpts(level)
          if ( n .gt. maxpts ) stop ' too many points to store!'
          zlat(n,level) = rlat
          zlon(n,level) = rlon
          time(n,level) = tmptim
          anomly(n,level) = tobs(1) - tm
         write(16,160) tm,iloop,tobs(1),anomly(n,level)
          endif
          go to 10
        enddo
11       continue
160    format('t local mean=',f8.3,' tobs',i7,'=',f8.3,' 2T.Diff=',f9.3)
      print *,'level/n=',level,n,numpts(level)
       write(16,*)'completed input from',obsfile
      print *,'completed input from',obsfile
C     DETERMINE THE COVARIANCE, VARIANCE, AND
C     COUNTER ARRAYS.
C     ANY LAGS OUTSIDE THE BIN BOUNDARIES GO INTO
      A "SPARE"
C     ROW AND COLUMN
      do level =1,1,1
```

```
C INITIALIZE THE ARRAYS TO ZERO.
                ndbnp1 = numdbn + 1
                ptntbnp1 = numtbn + 1
              do lagt=1,ntbnp1
                            do lagd=1,ndbnp1
                            covar(lagd,lagt) = 0.
                            var1(lagd,lagt) = 0.
                  var2(lagd,lagt) = 0.
                              num(lagd,lagt) = 0
            corr(lagd,lagt)= 0.
              end do
                end do
                numcmp = 0
     do ipt=1,numpts(level)-1
           anomi = anomly(ipt,level)
           zlati = zlat(ipt,level)
           zloni = zlon(ipt,level)
           timei = time(ipt,level)
           jpt = ipt + 1
        call tdsep
     do jpt=ipt+1,numpts(level)
          numcmp = numcmp + 1
           lagd = min0 ( lagdd(jpt) , ndbnp1 )
           lagt = min0 ( lagtt(jpt) , ntbnp1 )
           anomj = anomly(jpt,level)
           covar(lagd,lagt) = covar(lagd,lagt) + anomi * anomj
           var1(lagd,lagt) = var1(lagd,lagt) + anomi * anomi
           var2(lagd,lagt) = var2(lagd,lagt) + anomj * anomj
           num(lagd,lagt) = num(lagd,lagt) + 1
      end do
      end do
C     FOR EACH ELEMENT OF THE COVARIANCE AND
      VARIANCE
C     ARRAYS, DIVIDE THROUGH
C     BY THE NUMBER OF VALUES USED IN THE
      SUMMATIONS.
C     THEN CALCULATE THE CORRELATION MATRIX.
         lagtmx = 0
         lagdmx = 0
         nmax = 0
         do lagt=1,numtbn
          itimlg(lagt) = ( lagt - 1 ) * inctlg
```

```
          do lagd=1,numdbn
           n = num(lagd,lagt)
           if ( n .ne. 0 ) then
            covar(lagd,lagt) = covar(lagd,lagt) / n
            var1(lagd,lagt) = var1(lagd,lagt) / n
            var2(lagd,lagt) = var2(lagd,lagt) / n
            corr(lagd,lagt) = covar(lagd,lagt) /
     *      ( sqrt ( var1(lagd,lagt) * var2(lagd,lagt) ) )
             endif
            idstlg(lagd) = ( lagd - 1 ) * incdlg
            end do
           end do
C WRITE OUT THE CORRELATION MATRIX IN A MATRIX
  FORMAT.
          write(17,'(i4,35i6)') itimlg(1),(itimlg(i),i=1,numtbn)
          do lagd=1,numdbn
          write(17,'(i4,35f6.2)') idstlg(lagd),
     *    (corr(lagd,lagt),lagt=1,numtbn)
          end do
         close(17)

C    WRITE OUT THE MATRIX CONTAINING THE NUMBER OF
C    PAIRS COMPARED IN
C    EACH BIN. THE VALUES ARE SCALED TO 10.
          if ( nmax .lt. 9999 ) then
           scale = 1.
           else
           scale = 9999. / nmax
          endif
          do lagt=1,numtbn
          do lagd=1,numdbn
          num(lagd,lagt) = nint ( num(lagd,lagt) * scale )
          end do
         end do
         write(18,'(i4,35i6)') itimlg(1),(itimlg(i),i=1,numtbn)
         do lagd=1,numdbn
         write(18,'(i4,35i6)') idstlg(lagd),
     *    (num(lagd,lagt),lagt=1,numtbn)
         end do
       end do
1000  continue
       print *,'n/nmax/scale=',n,nmax,scale
      close(18)
        stop ' normal end '
      end
```

```
c
*******************************************************
      SUBROUTINE tdsep
         parameter ( maxpts =141330)
         common // zlat(maxpts,1), zlon(maxpts,1), time(maxpts,1),
     1       lagdd(maxpts) , lagtt(maxpts),
     2       xla1,xlo1,timei,jpt,numpts(64),incdlg,inctlg,level
         data pi/3.1415927/
         data rad/.0174533/
         data deg/57.29578/
      data ert/6371.2/
      A=XLA1*RAD
        COSA=COS(A)
        SINA=SIN(A)
      do 10 kpt=jpt,numpts(level)
       B=zlat(kpt,level)*RAD
        CC=ABS(zlon(kpt,level)-XLO1)*RAD
        AMB=ABS(A-B)
        IF(AMB.GT.RAD.AND.CC.GT.RAD) GO TO 15
        XMLAT=(A+B)/2.
        DELY=AMB
        DELX=CC*COS(XMLAT)
        C=SQRT(DELX**2.+DELY**2.)
        GO TO 20
   15   IF(CC.GT.PI) CC=2.*PI-CC
        COSB=COS(B)
        SINB=SIN(B)
        COSCC=COS(CC)
        COSC=sina*sinb+cosa*cosb*COSCC
        SINC=SQRT(1.-COSC**2.)
        C=ASIN(SINC)
        IF(COSC.LT.0.0) C=PI-C
   20  CDEG=C*ert
        distnc=CDEG
        lagdd(kpt) = nint ( distnc / incdlg ) + 1
       lagtt(kpt) = nint(abs (timei - time(kpt,level)) /inctlg) +1
   10  continue
        RETURN
        END
```

```
C*************************************************************
*
      SUBROUTINE LMEAN(tm,x,y,tlmean,crv)
      parameter(mx= 59,my=71)
c          tm =mean used for tobs
c          x = rlon y = rlat
       dimension tlmean(mx,my),ff(4),xx(4),yy(4)
       common /LL/dlat(my),dlon(mx),grid
c   Find the mean temp value corresponding to x,y
       ii=(x-dlon(1))/grid+1
       jj=(y-dlat(1))/grid+1
       if(ii.gt.mx.or.jj.gt.my) then
       print*,'II,JJ EXCEED LIMIT!!! ===>RETURN'
       return
       endif
       xx(1)=dlon(ii)
       yy(1)=dlat(jj)
       ff(1)=tlmean(ii,jj)
       xx(2)=xx(1)
       yy(2)=dlat(jj+1)
       ff(2)=tlmean(ii,jj+1)
       xx(3)=dlon(ii+1)
       yy(3)=yy(2)
       ff(3)=tlmean(ii+1,jj+1)
       xx(4)=xx(3)
       yy(4)=yy(1)
       ff(4)=tlmean(ii+1,jj)
       call BLINT(xx,yy,ff,4,x,y,tm,crv)
       return
       end
C*************************************************************
      subroutine blint(xx,yy,ff,n,x,y,f,Z)
      dimension ff(n),xx(n),yy(n)
C   Bilinear interpolation subroutine.
C   (Xi,Yi,fi) = data grid & values surounding model point (x,y)
C   f = interpolated value at the model grid point.
```

```
  if(ff(1).ge.Z.and.ff(2).ge.Z.and.ff(3).ge.Z.and.ff(4).ge.Z)then
    a1=xx(1)-xx(2)+xx(3)-xx(4)
    a2=-xx(1)+xx(4)
    a3=-xx(1)+xx(2)
    a4=xx(1)-x
    b1=yy(1)-yy(2)+yy(3)-yy(4)
    b2=-yy(1)+yy(4)
    b3=-yy(1)+yy(2)
    b4=yy(1)-y
    A=a3*b1-a1*b3
    B=b2*a3+b1*a4-a1*b4-a2*b3
    C=-a2*b4+a4*b2
    if(ABS(A*C).gt.0.002*B**2) then
      t=(-B-sqrt(B*B-4.*A*C))/(2.*A)
      else
         t=C/ABS(B)
    endif
10   continue
    A=a2*b1-a1*b2
    B=b3*a2+b1*a4-a1*b4-a3*b2
    C=-a3*b4+a4*b3
    if(ABS(A*C).gt.0.002*B**2) then
       s=(-B+sqrt(B*B-4.*A*C))/(2.*A)
       else
          s=-C/abs(B)
    endif
20   continue
    f=ff(1)*(1.-t)*(1.-s)+ff(2)*t*(1.-s)+ff(3)*s*t+ff(4)*(1.-t)*s
    else
     rmn=0.0
     fmn=0.0
     do i=1,n
      if(ff(i).ge.Z) then
       rrr=1./(1.0e-20+(x-xx(i))**2+(y-yy(i))**2)
       rmn=rmn+rrr
       fmn=fmn+ff(i)*rrr
      endif
    enddo
    if(rmn.gt.0.0) then
     f=fmn/rmn
    else
     f=-99.99
    endif
   endif
```

```
C   .....................................................
    return
    end
C ------------------------------------------------------
     SUBROUTINE JULDAT(IYR,IMO,IDAY,JULD)
C SUBR TO CALCULATE THE JULIAN DATE
C ACCOUNTS FOR LEAP YEARS
C INPUTS: IYR - YEAR (INTEGER 0 TO 99)
C         IMO - MONTH (INTEGER 1 TO 12)
C        IDAY - DAY (INTEGER (1 TO LAST DAY OF MONTH)
C OUTPUTS: JULD - JULIAN DATE (1 TO 365/366 FOR LEAP
   YEAR)
C VARIABLES:
C        MON (12) - DAYS IN EACH MONTH
C.....................................................
    DIMENSION MON(12)
    DATA MON /31,28,31,30,31,30,31,31,30,31,30,31/
    MON(2) = 28
C CHECK FOR LEAP YEAR
    IF (MOD(IYR,4) .EQ. 0 .AND. IYR .NE. 0) MON(2) = 29
    JULD = 0
    IF (IMO .EQ. 1) GOTO 120
    DO 100 K = 1, IMO-1
     JULD = JULD + MON(K)
100 CONTINUE
120 CONTINUE
     JULD = JULD + IDAY
     ym=iyr-1930
     ly=ym/4.0
    ily=ifix(ly)
    juld=juld+ym*365+ily
    RETURN
    END
    SUBROUTINE READER(X,Y,TEMP,ILOOP,iy,im,id,INFILE)
C *******************************************************
C *            SUBROUTINE READER2                       *
C *      READS TEMPS AT SPECIFIED PCT OF DBDB5   *
C *              FOR CHRTR                    *
C *******************************************************
```

```
C   INTERFACES TO CHRTR
C   THIS SUBROUTINE READS TEMPERATURE (OR SALINITY)
C   PROFILES.
C   THE TEMP AT A USER SPECIFIED PCT DEPTH, AND PASSES
    THE
C   LAT, LON, AND TEMP BACK TO PROGRAM CHRTR FOR
C   SUBSEQUENT GRIDDING.
C   VARIABLE WDEP2 IS DBDB5 DEPTH. XDEPTH IS DEPTH
    THAT TEMP
C   IS PICKED OFF (VERTICALLY INTERPOLATED) FROM EACH
    MOODS
C   PROFILE.
C   XDEPTH WILL BE DIFFERENT FOR EACH PROFILE,
    DEPENDING ON
C   WATER DEPTH (AS DETERMINED FROM DBDB5).
    NOTE THAT
C   DBDB5 WAS NEVER MEANT TO BE USED IN
    SHALLOW WATER,
C   THUS THE USER IS RESPONSIBLE TO UNDERSTAND ITS
C   LIMITATIONS IN THE SPECIFIC ANALYSIS REGION.
C
    CHARACTER infile*80
    DIMENSION TEMP(200)
C   FIRST TIME READER9 IS CALLED, SPECIFY AND OPEN
    INPUT FILE
     if(iloop.eq.0) then
           write(16,*)'opening ',infile
     close(3)
     open(3,FILE=infile,form='formatted',STATUS='old')
     read(3,*)lev
     write(6,600)lev
     write(16,600)lev
  600 format(1x,'what level (depth of water column)?',i4)
     endif
     iloop = iloop+1
C        READ PROFILE FROM MASTER FILE -
    5 continue
      READ(3,*,END=99) XLAT,XLON,t,iy,im,id
       Y = XLAT
       X = XLON
       TEMP(1) = T
       RETURN
   99 continue
       iloop = 0
       return
       END
```

```
**************************************************
      SUBROUTINE LINR(DELX,N,XBGN,XEND,ICT,X,Y,AX,AY)
C     LINEAR INTERPOLATOR ROUTINE FOR USE WHEN SPLINE
C     IS NOT APPROPRIATE
C     IE.LARGE GAPS IN INPUT DATA NEAR SHARP CHANGES IN
C     GRADIENT WITH LOCALLY INADEQUATE SAMPLING.
C     DELX=DESIRED INTERPOLATION INTERVAL,X AND Y ARE
      INPUT
C     WITH X=INDEP.
C     VARIABLE,AX AND AY ARE OUTPUT,N=LENGTH OF X,
      XBGN AND
C     XEND=DESIRED
C     BEGINNING AND ENDING VALUES OF AX,ICT=LENGTH OF AX
      DIMENSION X(*),Y(*),AX(*),AY(*)
      NN=N-1
      ICT=((XEND-XBGN)/DELX)+1.0001
      DO 1 J=1,ICT
      A=J-1
    1 AX(J)=A*DELX+XBGN
      KK=1
      DO 2 J=1,NN
      JJ=J+1
      SLOPE=(Y(JJ)-Y(J))/(X(JJ)-X(J))
      DO 3 K=KK,ICT
      I=K
      IF(AX(K).GT.X(JJ).AND.AX(K).LE.X(N)) GO TO 2
    3 AY(K)=SLOPE*(AX(K)-X(J))+Y(J)
      GO TO 4
    2 KK=I
    4 RETURN
      END
```

References

Aagaard K (1989) A synthesis of the Arctic Ocean circulation. Rapports. et Proces-verbaux des Réunion Conseil International pour. l'Exploration de la Mer, 188, 11–22.

Aagaard K, Coachman LK (1975) Toward an ice-free Arctic Ocean. EOS Transaction, American Geophysical Union, 56, 484–486.

Aagaard K, Coachman LK, Carmack EC (1981) On the halocline of the Arctic Ocean. Deep Sea Research, 28, 529–545.

Aagaard K, Swift JH, Carmack EC (1985) Thermohaline circulation in the Arctic Mediterranean seas. Journal of Geophysical Research, 90, 4833–4846.

Aagaard K, Foldvik A, Hillman SR (1987) The West Spitzbergen Current: Disposition and water mass transformation. Journal of Geophysical Research, 92, 3778–3784.

Arctic Climatology Project (1997) Environmental Working Group joint U.S.-Russian atlas of the Arctic Ocean – winter period, edited by L. Timokhov, F. Tanis. Environmental Research Institute of Michigan in association with the National Snow and Ice Data Center, Ann Arbor, MI. CD-ROM.

Arctic Climatology Project (1998) Environmental Working Group joint U.S.-Russian atlas of the Arctic Ocean – summer period, edited by L. Timokhov, F. Tanis. Environmental Research Institute of Michigan in association with the National Snow and Ice Data Center, Ann Arbor, MI. CD-ROM.

Arief D (1998) Outer southeast Asia: a region of deep straits, including the Benda Sea coastal segment (13, S). In: The Sea, Vol. 11, edited by Robinson and Brink, pp. 507–522.

ARGO Science Team (2001) Argo: the global array of profiling floats. In: Observing the Oceans in the 21st Century, edited by CJ. Koblinsky, and NR. Smith, GODAE Project Office, Bureau of Meteorology, Melbourne, pp. 248–258.

ASEAN Subcommittee on Climatology (1982) The ASEAN Climatic Atlas. Directorate of National Mapping. Malaysia, Kuala Lumpur, 104 pp.

Assireu AT, Stevenson MR, Stech JL (2003) Surface circulation and kinetic energy in the SW Atlantic obtained by drifters. Continental Shelf Research, 23, 145–157.

Bagriantsev NV, Gordon AL, Huber BA (1989) Weddell Gyre: Temperature and maximum stratum. Journal of Geophysical Research, 94, 8331–8334.

Bang I, Choi JK, Kantha L, Horton C, Cliford M, Suk MS, Chang KI, Nam SY, Lie HJ (1996) A hindcast experiment in the East Sea (Sea of Japan), La mer, 34, 108–130.

Bannon PR, Chu PC (1988) Anelastic semi-geostrophic flow over a mountain ridge. Journal of the Atmospheric Sciences, 45, 1025–1029.

Bathen KH (1972) On the seasonal changes in the depth of the mixed layer in the North Pacific Ocean. Journal of Geophysical Research, 77, 7138–7150.

Beckmann A, Hartmut H, Timmermann R (1999) A numerical model of the Weddell Sea: Large-scale circulation and water mass distribution. Journal of Geophysical Research, 104, 23375–23391.

Behringer DW, Stommel H (1980) The beta spiral in the North Atlantic subtropical gyre. Deep Sea Research, 27A, 225–238.

Bennett A (1992) Inverse methods in physical oceanography. Cambridge University Press, Cambridge, UK, 346 pp.

Bingham FM, Lukas R (1994) The Southward intrusion of North Pacific Intermediate Water along the Mindanao coast. Journal of Physical Oceanography, 24, 141–154.

Bingham, FM, Lukas R (1995) The distribution of intermediate water in the western equatorial Pacific during January–February 1986. Deep Sea Research, 42, 1545–1573.

Bingham FM, Talley LD (1991) Estimates of Kuroshio transport using an inverse technique. Deep Sea Research, 38, S21-S43.

Bleck R, Smith L (1990) A wind-driven isopycnal coordinate model of the north and equatorial Atlantic Ocean. I. Model development and supporting experiments. Journal of Geophysical Research, 95, 3273–3285.

Blumberg A, Mellor G (1987) A description of a three dimensional coastal ocean circulation model, Three-Dimensional Coastal Ocean Models, edited by NS. Heaps, American Geophysics Union, Washington, DC, pp. 1–16.

Boebel O, Schmid C, Podesta G, Zenk W (1999) Intermediate water in the Brazil–Malvinas Confluence Zone: A Lagrangian view. Journal of Geophysical Research, 104, 21063–21082.

Bower AS, LeCann B, Rossby T, Zenh W, Gould J, Speer K, Richardson PL, Prater MD, Zhang HM (2002) Directly measured mid-depth circulation in the northeastern North Atlantic Ocean. Nature, 419, 603–607.

Broecker WS (1991) The great ocean conveyor. Oceanography, 4, 79–89.

Bryan K (1987) Parameter sensitivity of primitive equation ocean general circulation models. Journal of Physical Oceanography, 17, 970–985.

Bryden HL (1983) The Southern Ocean. In: Eddies in the Ocean, edited by AR. Robinson, Springer, New York Berlin Heidelberg, 265–277.

Bryden HL, Kinder TH (1980) Heat transport by currents across 25°N latitude in the Atlantic Ocean. Science, 207, 884–886.

Bryden HL, Phillsbury RD (1977) variability of deep flow in the Drake Passage from year-long current measurements. Journal of Physical Oceanography, 7, 803–810.

Burgers G, Balmaseda MA, Vossepoel FC, van Oldenborgh GJ, van Leeuwen PJ (2002) Balanced ocean-data assimilation near the equator. Journal of Physical Oceanography, 32, 2509–2519.

Campos EJD, Miller JL, Muller TJ, Peterson RG (1995) Physical oceanography of the Southwest Atlantic Ocean. Oceanography, 8, 87–91.

Carmack EC, Foster TD (1975) Circulation and distribution of oceanographic properties near Filchner Ice Shelf. Deep Sea Research, 22, 77–90.

Carsey FD (1982) Arctic sea ice distribution at the end of summer 1973–1976 from satellite microwave data. Journal of Geophysical Research, 87, 5809–5835.

Cattle H (1985) Diverting Soviet rivers: Some possible repercussions for the Arctic Ocean. Polar Record, 22, 485–498.

Chao SY, Shaw PT, Wang J (1996) Wind relaxation as a possible cause of the South China Sea Warm Current. Journal of Oceanography, 51, 111–132.

Chapman D (1985) Numerical treatment of cross-shelf open boundaries in a barotropic ocean model. Journal of Physical Oceanography, 15, 1060–1075.

Cheang BK (1980) Some aspects of winter monsoon and its characteristics in Malaysia. Research Publication, No. 2, Malaysian Meteorological Service, Kuala Lumpur.

Chen CS, Beardsley RC, Limeburner R (1992) The structure of the Kuroshio southwest of Kyushu: Velocity, transport and potential vorticity fields. Deep Sea Research, 39, 245–268.

Chen CS, Beardsley RC, Limeburner R, Kim K (1994) Comparison of winter and summer hydrographic observations in the Yellow and East China Seas and adjacent Kuroshio during 1986. Continental Shelf Research, 14, 909–929.

Chu PC (1986) An instability theory of ice–air interaction for the migration of the marginal ice zone. Geophysical Journal of Royal Astronomical Society, 86, 863–883.

Chu PC (1987a) Generation of unstable modes of the iceward attenuating swell by icebreeze. Journal of Physical Oceanography, 17, 828–832.

Chu PC (1987b) An instability theory of ice–air interaction for the formation of ice-edge bands. Journal of Geophysical Research, 92, 6966–6970.

Chu PC (1987c) An icebreeze mechanism for an ice divergence–convergence criterion in the marginal ice zone. Journal of Physical Oceanography, 17, 1627–1632.

Chu PC (1988) An instability theory of air–sea interaction for coastal upwelling. Advances in Atmospheric Sciences, 5, 277–285

Chu PC (1989) Relationship between sea surface temperature gradient and thermally forced planetary boundary layer air flow. Pure and Applied Geophysics, 130, 31–45.

Chu PC (1993) Generation of low frequency unstable modes in a coupled equatorial troposphere and ocean mixed layer. Journal of the Atmospheric Sciences, 50, 731–749.

Chu PC (1995a) P-vector method for determining absolute velocity from hydrographic data. Marine Technological Society Journal, 29(2), 3–14.

Chu PC (1995b) A feature model for Arctic upper ocean thermal structure. Proceedings on the Fourth Conference on Polar Meteorology and Oceanography, American Meteorological Society, 224–227.

Chu PC (1999a) Two kinds of predictability in Lorenz system, Journal of the Atmospheric Sciences, 56, 1427–1432.

Chu PC (1999b) Fundamental circulation functions for the determination of open boundary conditions. Proceedings on the Third Conference of Coastal Oceanic and Atmospheric Prediction, American Meteorological Society, 389–394.

Chu PC (2000) P-vector spirals and determination of absolute velocities. Journal of Oceanography, 56, 591–599.

Chu PC (2002) C-vector for identification of oceanic secondary circulation across Arctic fronts in Fram Strait. Geophysical Research Letters, 29, 10.1029/2002GLO15978.

Chu PC, Chang CP (1997) South China Sea warm pool. Advances in Atmospheric Sciences, 14, 195–206.

Chu PC, Fan CW (2001) A low salinity cool-core cyclonic eddy detected northwest of Luzon during the South China Sea Monsoon Experiment (SCSMEX) in July 1998. Journal of Oceanography, 57, 549–563.

Chu PC, Fan CW (2006) An inverse model for calculation of global volume transport from wind and hydrographic data. Journal of Marine Systems, in press.

Chu PC, Garwood RW Jr (1990) Thermodynamic feedback between cloud and ocean mixed layer. Advances in Atmospheric Sciences, 7, 1–10.

Chu PC, Garwood RW Jr (1991) On the two – phase thermodynamics of the coupled cloud – ocean mixed layer. Journal of Geophysical Research, 96, 3425–3436.

Chu PC, Lan J (2003) Extremely strong thermohaline source/sinks generated by diagnostic initialization. Geophysical Research Letters, 30(6), 10.1029/2002GL016525.

Chu PC, Li RF (2000) South China Sea isopycnal surface circulations. Journal of Physical Oceanography, 30, 2419–2438.

Chu PC, Liu H (1999) Seasonal and interannual variability of the world ocean thermal structure. Proceedings on the 10th Symposium on Global Change Studies, American Meteorological Society, 151–154.

Chu PC, Wang GH (2003) Seasonal variability of thermohaline front in the central South China Sea. Journal of Oceanography, 59, 65–78.

Chu PC, Garwood RW Jr, Muller P (1990) Unstable and damped modes in coupled ocean mixed layer and cloud models. Journal of Marine Systems, 1, 1–11, 1990.

Chu PC, Wells SK, Haeger SD, Szczechowski C, Carron MJ (1997a) Temporal and spatial scales of the Yellow Sea thermal variability. Journal of Geophysical Research, 102, 5655–5668.

Chu PC, Fralick CR, Haeger SD, Carron MJ (1997b) A parametric model for Yellow Sea thermal variability. Journal of Geophysical Research, 102, 10499–10508.

Chu PC, Tseng HC, Chang CP, Chen JM (1997c) South China Sea warm pool detected from the Navy's Master Oceanographic Observational Data Set (MOODS). Journal of Geophysical Research, 102, 15761–15771.

Chu PC, Lu SH, Chen YC (1997d) Temporal and spatial variabilities of the South China Sea surface temperature anomaly. Journal of Geophysical Research, 102, 20937–20955.

Chu PC, Fan CW, Ehret LL (1997e) Determination of open boundary conditions from interior observational data. Journal of Atmospheric and Oceanic Technology, 14, 723–734.

Chu PC, Fan CW, Cai WJ (1998a) Evaluation of P-vector method using modular ocean model (MOM). Journal of Oceanography, 54, 185–198.

Chu PC, Chen YC, Lu SH (1998b) Temporal and spatial variabilities of Japan Sea surface temperature and atmospheric forcings. Journal of Oceanography, 54, 273–384.

Chu PC, Lu SH, Chen YC (1998c) Wind-driven South China Sea deep basin warm-core/cool-core eddies. Journal of Oceanography, 54, 347–360.

Chu PC, Fan CW, Lozano CJ, Kerling J (1998d) An airborne expandable bathythermograph (AXBT) survey of the South China Sea, May 1995. Journal of Geophysical Research, 103, 21637–21652.

Chu PC, Wang QQ, Bourke RH (1999a) A geometric model for Beaufort/Chukchi Sea thermohaline structure. Journal of Atmospheric and Oceanic Technology, 16, 613–632.

Chu PC, Lu SH, Liu WT (1999b) Uncertainty of the South China Sea prediction using NSCAT and NCEP winds during tropical storm Ernie 1996. Journal of Geophysical Research, 104, 11273–11289.

Chu, PC, Lu SH, Chen YC (1999c) A coastal atmosphere-ocean coupled system (CAOCS) evaluated by an airborne expandable bathythermograph survey in the South China Sea, May 1995. Journal of Oceanography, 55, 543–558.

Chu PC, Edmons NL, Fan CW (1999d) Dynamical mechanisms for the South China Sea seasonal circulation and thermohaline variabilities. Journal of Physical Oceanography, 29, 2971–2989.

Chu PC, Fan CW, Liu WT (2000a) Determination of sub-surface thermal structure from sea surface temperature. Journal of Atmospheric and Oceanic Technology, 17, 971–979.

Chu PC, Veneziano JM, Fan CW (2000b) Response of the South China Sea to tropical cyclone Ernie 1996. Journal of Geophysical Research, 105, 13991–14009.

Chu PC, Lan J, Strauhs H (2000c) A numerical simulation of the Japan/East Sea (JES) seasonal circulation. Estuarine and Coastal Modeling, 6, American Society of Civil Engineering, 94–113.

Chu PC, Lan J, Fan CW (2001a) Japan Sea circulation and thermohaline structure. Part 1. Climatology. Journal of Physical Oceanography, 31**,** 244–271.

Chu PC, Lan J, Fan CW (2001b) Japan Sea circulation and thermohaline structure. Part 2. A variational P-vector method. Journal of Physical Oceanography, 31, 2886–2902.

Chu PC, Lu SH, Fan CW (2001c) An air-ocean coupled nowcast/forecast system for the east Asian marginal seas. Advances in Mathematical Modeling of Atmosphere and Ocean Dynamics, edited by PE. Hodnett, Kluwer Scientific Publishing Co., pp. 137–142.

Chu PC, Liu QY, Jia YL, Fan CW (2001d) Evidence of barrier layer in the Sulu and Celebes Seas. Journal of Physical Oceanography, 32, 3596–3615.

Chu PC, Li RF, You XB (2002a) Northwest Pacific subtropical countercurrent on isopycnal surface in Summer. Geophysical Research Letters, 29, 10.1029/2002GLO14831.

Chu PC, Wang GH, Chen YC (2002b) Japan Sea circulation and thermohaline structure. Part 3. Autocorrelation Functions. Journal of Physical Oceanography, 32, 3596–3615.

Chu PC, Ma BB, Chen YC (2002c) South China Sea thermohaline structure and circulation. Acta Oceanologia Sinica, 21, 227–261.

Chu PC, Ivanov LM, Korzhova TP, Margolina TM, Melnichenko OM (2003a) Analysis of sparse and noisy ocean current data using flow decomposition. Part 1. Theory. Journal of Atmospheric and Oceanic Technology, 20, 478–491.

Chu PC, Ivanov LM, Korzhova TP, Margolina TM, Melnichenko OM (2003b) Analysis of sparse and noisy ocean current data using flow decomposition. Part 2. Application to Eulerian and Lagrangian data. Journal of Atmospheric and Oceanic Technology, 20, 492–512.

Chu PC, Lu SH, Fan CW, Kim CS (2003c) A numerical simulation of Japan/East Sea (JES) thermohaline structure and circulation. Advances in Coastal Modeling, edited by VC. Lakhan, Elsevier Oceanographic Series, 67, 431–466.

Chu PC, Li RF, Fan CW (2003d) Determination of the current system on isopycnal surface between Mindanao and New Guinea from GDEM. Chinese Journal of Oceanology and Limnology, 21, 193–213.

Chu PC, Wang GH, Fan CW (2004a) Evaluation of US Navy's Modular Ocean Data Assimilation System Using SCSMEX data. Journal of Oceanography, 60, 1007–1021.

Chu PC, Ivanov LM, Margolina TM (2004b) Rotation method for reconstructing process and field from imperfect data. International Journal of Bifurcation and Chaos, 14(8), 2991–2997.

Chu PC, Ivanov LM, Margolina TM (2005a) Seasonal variability of the Black Sea chlorophyll-a concentration. Journal of Marine Systems, 56, 243–261.

Chu PC, Ivanov LM, Melnichenko OM (2005b) Fall-winter current reversals on the Texas–Louisiana continental shelf. Journal of Physical Oceanography, 35, 902–910.

Chu PC, Fang CL, Kim CS (2005c) Japan/East Sea model predictability. Continental Shelf Research, 25, 2107–2121.

Chu PC, Chen YC, Kuninaka A (2005d) Seasonal variability of the East China/Yellow Sea surface buoyancy flux and thermohaline structure. Advances in Atmospheric Sciences, 22, 1–20.

Chu PC, Ivanov LM, Melnichenko OM (2006) ARGO floats revealing bimodality of large-scale mid-depth circulation in the North Atlantic. Journal of Oceanography, submitted.

Clancy RM, Pollak KD (1983) A real-time synoptic ocean thermal analysis/forecast system. Progress in Oceanography, 12, 383–424.

Clark RA, Hill HW, Reiniger RF, Warren BA (1980) Current system south and east of the Grand Bank of Newfoundland. Journal of Physical Oceanography, 10, 25–65.

Coachman LK, Aaggard K, Tripp RB (1975) Bering Strait: the Regional Oceanography. University of Washington Press, Seattle, 172 pp.

Coats DA (1981) An estimate of absolute geostrophic velocity from the density field in the northeastern Pacific Ocean. Journal of Geophysical Research, 86, 8031–8036.

Cox MD (1987): GFDL Ocean Model Circular No. 7. GFDL/Princeton University, Princeton, NJ, 1 pp.

Cushman-Roisin B (1984) On the maintenance of the Subtropical Front and its associated countercurrent. Journal of Physical Oceanography, 14, 1179–1190.

Dale WL (1956) Winds and drift currents in the South China Sea. Malayan Journal of Tropical Geography, 8, 1–31.

Danilov AI, Ivanov LM, Kulakov MY, Margolina TM, Pavlov VK (1996) Modern radioactive climate of the Kara Sea. Geophysical Report, Russian Academy of Sciences, 346(4), 545–548 (in Russian).

da Silva AM, Young CC, Levitus S (1994) Atlas of Surface Marine Data 1994. Technical Report of Geosciences, 94, University of Wisconsin-Milwaukee, 83 pp.

Davis R (1978) On estimating velocity from hydrographic data. Journal of Geophysical Research, 83**,** 5507–5509.

Deacon GER (1979) The Weddell gyre. Deep Sea Research, 26A, 981–995.

Deacon GER (1982) Physical and biological zonation in the Southern Ocean. Deep Sea Research, 29, 1–16.

Deacon GER, Foster TD (1977) The boundary region between the Weddell Sea and Drake Passage currents. Deep Sea Research, 24, 505–510.

Defant A (1941) Quantitative Untersuchngen zur Statik und Dynamik des Atlantischen Ozeans. Die absolute Topographie des physikalischen Meeresniveaus und der Druckflachen sowie die Wasserbewegungen im Raum des Atlantischen Ozeans. In Wissenschaftliche Ergebnisse der Deutschen Atlantischen Expedition auf dem Forschungs-und Vermessungsschiff "Meteor" 1925–27, 6, Part 1, 191–260.

Defant A (1961) Physical Oceanography, Vol 1, Pergamon, New York, 729 pp.

DeMaster DJ, Dunbar RB, Gordon LI, Leventer AR, Morrison JM, Nelson DM, Nittrouer CA, Smith WO Jr (1992) Cycling and accumulation of biogenic and organic matter in high-latitude environments. The Ross Sea, Oceanography, 5, 146–153.

Dietrich G (1969) Atlas of the hydrography of the northern North Atlantic Ocean. International Council for the Expedition of the Sea, Copenhagen, 140 pp.

Egawa T, Nagata Y, Sato S (1993) Seasonal variation of the current in the Tsushima Strait deduced from ADCP data of Ship-of-Opportunity. Journal of Oceanography, 49, 39–50.

Emery KO, Csanady GT (1973) Surface circulation of lakes and nearly landlocked seas. Proceedings of National Academy of Sciences, USA, 70, 93–97.

Eremeev VN, Ivanov LM, Kirwan AD Jr, Melnichenko OV, Kochergin SV, Stanichnaya RR (1992) Reconstruction of oceanic flow characteristics from quasi-Lagrangian data. Part 2. Characteristics of the large-scale circulation in the Black Sea. Journal of Geophysical Research, 97, 9743–9753.

Eremeev VN, Ivanov LM, Kirwan AD Jr, Margolina TM (1994) Amount of Cs-137 and Cs-134 radionuclides in the Black Sea produced by the Chernobyl disaster. Journal of Environmental Radiology, 26, 49–63.

Eremeev VN, Ivanov LM, Kirwan AD Jr, Margolina TM (1995) Analysis of cesium pollution in the Black Sea by regularization method. Marine Pollution Bulletin, 30(7), 460–462.

Evans DL, Signorini SS, Miranda LB (1983) A note on the transport of the Brazil Current. Journal of Physical Oceanography, 13, 1732–1738.

Eykhoff P (1973) System Identification: Parameter and State Estimation. Elsevier, Amsterdam, 555 pp.

Fahrbach E, Meincke J, Osterhus S, Rohardt G, Schauer U, Tverberg V, Verduin J (2001) Direct measurements of volume transports through Fram Strait. Polar Research, 20, 217–224.

Fandry CG, Pillsbury RD (1979) On the estimation of absolute geostrophic volume transport applied to the Antarctic Circumpolar Current. Journal of Physical Oceanography, 9, 449–455.

Ffield A, Gordon AL (1992) Vertical mixing in the Indonesian thermocline. Journal of Physical Oceanography, 22, 184–195.

Fieux M, Andrie C, Delecluse P, Ilahude AG, Kartavtseff A, Mantisi F, Molcard R, Swallow JC (1994) Measurements within the Pacific-Indian ocean throughflow region. Deep Sea Research, 41, 1091–1130.

Fine RA, Lukas R, Bingham FM, Warner WJ, Gammon RH (1994) The western equatorial Pacific – a water mass crossroads. Journal of Geophysical Research, 99, 25063–25080.

Foldvik A, Kvinge T, Torresen T (1985) Bottom currents near the continental shelf break in the Weddell Sea. Antarctic Research Series, 43, 21–34.

Foldvik A, Aagaard K, Torresen T (1988) On the velocity field of the East Greenland Current. Deep Sea Research, 35, 1335–1354.

Foster TD, Middleton JH (1979) Variability in the bottom water of the Weddell Sea. Deep Sea Research, 26, 743–762.

Fox DN, Teague WJ, Barron CN, Carnes MR, Lee CM (2002) The modular ocean data assimilation system (MODAS). Journal of Atmospheric and Oceanic Technology, 19, 240–252.

Fuglister FC (1960) Atlantic Ocean Atlas of Temperature and Salinity Profiles and Data from the International Geophysical Year of 1957–1958. Woods Hole Oceanographic Institution Atlas Series, 1, 209 pp.

Fuglister FC (1963) Gulf Stream '60. Progress in Oceanography, 1, 265–373.

Gandin LS (1965) Objective Analysis of Meteorological Fields. Israel Program for Scientific Translation, Jerusalem, 242 pp.

Garzoli SL, Bianchi A (1987) Time-space variability of the local dynamics of the Malvinas–Brazil confluence as revealed by inverted echo sounders. Journal of Geophysical Research, 92, 1914–1922.

Gascard JC, Kergomard C, Jeannin PF, Fily M (1988) Diagnostic study in Fram Strait marginal ice zone during summer from 1983 and 1984 marginal ice zone experiment Lagrangian observations. Journal of Geophysical Research, 93, 3613–3641.

Gascard JC, Richez C, Rouault C (1995) New insights on large-scale oceanography in Fram strait: the West Spitzbergen Current. In Arctic Oceanography: Marginal Ice Zones and Continental Shelves. Coastal and Estuarine studies, 49, 131–182.

Gerdes R, Schauer U (1997) Large-scale circulation and water mass distribution in the Arctic Ocean from model results and observations. Journal of Geophysical Research, 102, 8467–8484.

Godfrey JS (1989) A Sverdrup model of the depth-integrated flow for the world ocean allowing for island circulations. Geophysical and Astrophysical Fluid Dynamics, 45, 89–112.

Godfrey JS (1996) The effect of the Indonesian Throughflow on ocean circulation and heat exchange with the atmosphere: A review. Journal of Geophysical Research, 101, 12217–12237.

Goni G, Wainer I (2001) Investigation of the Brazil Current front variability from altimeter data. Journal of Geophysical Research, 106, 31117–31128.

Gordon AL, Greengrove CL (1986) Geostrophic circulation of the Brazil–Falkland Confluence. Deep Sea Research, 33, 573–585.

Gordon AL, Georgi DT, Taylor WH (1977) Antarctic Polar Front zone in the western Scotia Sea, summer 1975. Journal of Physical Oceanography, 7, 309–328.

Gordon AL, Martinson DG, Taylor WH (1981) The wind-driven circulation in the Weddell-Enderby basin. Deep Sea Research, 28, 151–163.

Haney RL (1971) Surface boundary condition for ocean circulation models. Journal of Physical Oceanography, 1, 241–248.

Hase H, Yoon JH, Koterayama, W (1999) The current structure of the Tsushima Warm Current along the Japanese Coast. Journal of Oceanography, 55, 217–235.

Hidaka K (1940a) Absolute evaluation of ocean currents in dynamic calculations. Proceedings of Imperial Academy of Tokyo, 16, 391–393.

Hidaka K (1940b) Practical evaluation of ocean currents. Proceedings of Imperial Academy of Tokyo, 16, 394–397.

Hogg NG (1992) On the transport of the Gulf Stream between Cape Hatteras and the Grand Banks. Deep Sea Research, 39, 1231–1246.

Holland WR (1973) Baroclinic and topographic influences on the transport in western boundary currents. Geophysical Fluid Dynamics, 4, 187–210.

Hopkins TS (1991) The GIN Sea, Review of physical oceanography and literature from 1972. Earth Science Review, 30, 175–318.

Hoskins BJ, Draghici L, Davies HC (1978) A new look at the ω-equation. Quarterly Journal of Royal Meteorological Society, 104, 31–38, 1978.

Hu D, Cui M (1989) The western boundary current in the far-western Pacific Ocean. Proceedings of the Western Pacific International Meeting and Workshop on TOGA COARE, ORSTOM, Noumea, New Caledonia, 24–30 May, edited by J. Picaut, R. Lukas, T. Delcroix, pp. 135–143.

Hu Y, Guan C, Gao H (1992) Water temperature and circulation structure in the upper ocean of the northern South China Sea. Oceanography in China, Vol. 6, Ocean Press, Beijing, pp. 60–69 (in Chinese with English abstract).

Huang QZ, Wang WZ, Li YS, Li CW (1994) Current characteristics of the South China Sea. Oceanology of China Seas, Vol 1, edited by D. Zhou, YB. Liang, CK. Tseng, Kluwer Academic Press, Norwell, Massechussetts, pp. 113–122.

Inoue N, Mitta T, Tawara S (1985) Tsushima Strait, II Physics. Coastal Oceanography of Japanese Island, Tokai University Press, pp. 914–933 (in Japanese).

Iselin C (1936) A study of the circulation of the western North Atlantic. Papers in Physical Oceanography and Meteorology, 4(4), 101.

Isobe A (1994) Seasonal variability of the barotropic and baroclinic motion in the Tsushima–Korea Strait. Journal of Oceanography, 50, 223–238.

Isoda Y, Saitoh S (1988) Variability of the sea surface temperature obtained by the statistical analysis of AVHRR imagery – A case study of the south Japan Sea. Journal of Oceanographic Society of Japan, 44, 52–59.

Isoda Y, Saitoh S (1993) The northward intruding eddy along the east coast of Korea. Journal of Oceanography, 49, 443–458.

Ivanov LM, Kirwan AD Jr, Margolina TM (2001) Filtering noise from oceanographic data with some applications for the Kara and Black Seas. Journal of Marine Systems, 28(1–2), 113–139.

Ivanov LM, Margolina TM, Danilov AI (2004) Application of inverse technique to study radioactive pollution and mixing processes in the Arctic Seas. Journal of Marine Systems, 48(1–4), 117–131.

Johnson CM (1980) Wintertime sea–ice extremes and the simultaneous atmospheric circulation. Monthly Weather Review, 108, 1782–1791.

Johnson GC, McPhaden MJ (1999) Interior pycnocline flow from the subtropical to the equatorial Pacific Ocean. Journal of Physical Oceanography, 29, 3073–3089.

Jones EP, Anderson LG (1986) On the origin of the chemical properties of the Arctic Ocean halocline. Journal of Geophysical Research, 91, 10759–10767.

Joyce TM, Patterson SL, Millard RC Jr (1981) Anatomy of a cyclonic ring in the Drake Passage. Deep Sea Research, 28, 1265–1287.

Kajiura K, Tsuchiya M, Hidaka K (1958) The analysis of oceanographic condition in the Japan Sea. Report on Development of Fishery Resource in the Tsushima Warm Current, 1, 158–170 (in Japanese).

Kano Y (1980) The annual variation of the temperature, salinity and oxygen contents in the Japan Sea. The Oceanographic Magazine, 31, 15–26.

Kara AB, Rochford PA, Hurlburt HE (2000) Mixed layer depth variability and barrier layer formation over the North Pacific Ocean. Journal of Geophysical Research, 105, 16783–16801.

Kashino Y, Aoyama M, Kawano T, Hendiarti N, Syaefudin Y, Anantasena Y, Muneyama K, Watanabe H (1996) The water masses between Mindanao and New Guinea. Journal of Geophysical Research, 101, 12391–12400.

Kashino Y, Watanabe H, Herunadi B, Aoyama M, Hartoyo D (1999) Current variability at the Pacific entrance of the Indonesian Through flow. Journal of Geophysical Research, 104, 11021–11035.

Kawabe M (1982a) Branching of the Tsushima Current in the Japan Sea. Part I. Data analysis. Journal of Oceanographic Society of Japan, 38, 95–107.

Kawabe M (1982b) Branching of the Tsushima Current in the Japan Sea. Part II. Numerical experiment. Journal of Oceanographic Society of Japan, 38, 183–192.

Keffer T (1985) The ventilation of the world's oceans: maps of the potential vorticity field. Journal of Physical Oceanography, 15, 509–523.

Killworth P (1986) A Bernoulli inverse method for determining the ocean circulation. Journal of Physical Oceanography, 16, 2031–2051.

Kim K, Chung JY (1984) On the salinity-minimum and dissolved oxygen-maximum layer in the East Sea (Sea of Japan). Ocean Hydrodynamics of the Japan and East China Seas, edited by T. Ichiye, Elsevier, Amsterdam, pp. 55–65.

Kim YG, Kim K (1999) Intermediate Waters in the East/Japan Sea. Journal of Oceanography, 55, 123–132.

Kim CH, Yoon JH (1996) Modeling of the wind-driven circulation in the Japan Sea using a reduced gravity model. Journal of Oceanography, 52, 359–373.

Klinck JM, Nowland WD Jr (2001) Antarctic Circumpolar Current. Encyclopedia of Ocean Science, San Diego, 1st Edition, Academic Press, pp. 151–59.

Kubokawa A (1997) A two-level model of subtropical gyre and subtropical countercurrent. Journal of Oceanography, 53, 231–244, 1997.

Lavender KL, Davis RE, Owens WB (2000) Direct velocity measurements described a new circulation regime in the Labrador and Irminger seas. Nature, 407, 66–69.

Lavender KL, Owens B, Davis RE (2005) Mid-depth circulation of the subpolar North Atlantic as measured by surface floats. Deep Sea Research I, 52, 767–785.

Legutki S (1991) A numerical investigation of the circulation in the Greenland and Norwegian seas. Journal of Physical Oceanography, 21, 118–148.

Lentini CAD, Podesta GG, Campos EJD, Olson DB (2001) Sea surface temperature anomalies on the Western South Atlantic from 1982–1994. Continental Shelf Research, 21, 89–112.

Levitus S (1984) Annual cycle of temperature and heat storage in the world ocean. Journal of Physical Oceanography, 14, 727–746.

Levitus S, Boyer T (1994) World Ocean Atlas, Vol 4: Temperature. NOAA Atlas NESDIS, 4, U.S. Government Printing Office, Washington, DC, 117 pp.

Levitus S, Burgett R, Boyer T (1994) World Ocean Atlas, Vol 3: Salinity. NOAA Atlas NESDIS, 3, U.S. Government Printing Office, Washington, DC, 99 pp.

Li H, Yuan Y (1992) On the formation and maintenance mechanisms of the cold water mass of the Yellow Sea. Chinese Journal of Oceanology and Limnology, 10(2) 97–106.

Li L, Nowlin WD Jr, Su J (1998) Anticyclonic rings from the Kuroshio in the South China Sea. Deep Sea Research, 45, 1469–1482.

Li R, Zeng Q, Ji Z, Gun D (1992) Numerical simulation for a northeastward flowing current from area off the eastern Hainan Island to Tsugaru/Soya Strait. La Mer, 30, 229–238.

Li RF, You XB, Chu PC (2005) The eastward subtropical countercurrent on isopycnal surface in the Western North Pacific. Science in China Ser D, Earth Sciences, 48(7), 1065–1073.

Lindstrom E, Lukas R, Fine R, Firing E, Godfrey S, Meyeyers G, Tsuchiya M (1987) The western Equatorial Pacific ocean circulation study. Nature, 330, 533–537.

Liu Q, Jia Y, Liu P, Wang Q, Chu PC (2001) Seasonal and intraseasonal thermocline variability in the central South China Sea. Geophysical Research Letters, 28, 4467–4470.

Lorenz EN (1963) Deterministic nonperiodic flow. Journal of the Atmospheric Sciences, 20, 130–141.

Lozano CJ, Robinson AR, Arango HG, Gangopadhyay A, Sloan Q, Haley PJ, Anderson L, Leslie W (1996) An interdisciplinary ocean prediction system: assimilation strategies and structure data model. Modern Approaches to Data Assimilation in Ocean Modeling, edited by P. Malanotte-Rizzoli, Elsevier, Amsterdam, pp. 413–452.

Lozier MS, McCartney MS, Owens WB (1994) Anomalous anomalies in averaged hydrographic data. Journal of Physical Oceanography, 24, 2624–2638.

Lozier MS, Owens WB, Curry RG (1995) The climatology of the North Atlantic. Progress in Oceanography, 36, 1–44.

Lukas R (1988) Interannual fluctuations of the Mindanao current inferred from sea-level. Journal of Geophysical Research, 93, 6744–6748.

Lukas R, Lindstrom E (1991) The mixed layer of the western equatorial Pacific Ocean. Journal of Geophysical Research, 96, 3343–3357.

Lukas R, Firing E, Hacker P, Richardson PL, Collins CA, Fine R, Gammon R (1991) Observations of the Mindanao current during the western equatorial Pacific-Ocean circulation study. Journal of Geophysical Research, 96, 7089–7104.

Lukas R, Yamagata T, McCreary JP (1996) Pacific low-latitude western boundary currents and the Indonesian throughflow. Journal of Geophysical Research, 101, 12209–12216.

Maamaatuaiahutapu K, Provost C, Andrie C, Vigan X (1999) Origin and ages of mode waters in the Brazil–Malvinas Confluence region during austral winter 1994. Journal of Geophysical Research, 104, 21051–21061.

Maizuru Marine Observatory (1997) Climate Chart of the Japan Sea. Maizuru, Japan, 1 pp.

Martin S, Kawase M (1998) The southern flux of sea ice in the Tatarskiy Strait, Japan Sea and the generation of the Liman Current. Journal of Marine Research, 56, 141–155.

Masumoto Y, Yamagata T (1991) Response of the western tropical Pacific to the Asian winter monsoon – the generation of the Mindanao dome. Journal of Physical Oceanography, 21, 1386–1398.

Masuzawa J (1969) The Mindanao Current. Bulletin of Japanese Society of Fishery and Oceanography, pp. 99–104.

Maykut GA, McPhee MG (1995) Solar heating of the Arctic mixed layer. Journal of Geophysical Research, 100, 24691–24703.

McCartney MS (1982) The subtropical recirculation of Mode Waters. Journal of Marine Research, 40(Suppl.), 427–464.

McCreary JP, Anderson DLT (1984) A simple model of El Nino and the Southern Oscillation. Monthly Weather Review, 112, 934–946.

McDougall TJ (1988) Neutral-surface potential vorticity. Progress in Oceanography, 20, 185–221.

McWilliams JC (1977) A note on a consistent quasigeostrophic model in a multiply connected domain. Dynamics of Atmosphere and Oceans, 1, 427–441.

Melling H, Lewis EL (1982) Shelf drainage flows in the Beaufort Sea and their effect on the Arctic Ocean pycnocline. Deep Sea Research, 29, 967–985.

Mellor GL (2003) Users guide for a three-dimensional, primitive equation, numerical ocean model. Program in Atmospheric and Oceanic Sciences, Princeton University, 53 pp.

Memery L, Arhan M, Alvarez-Salgado XA, Messias MJ, Mercier H, Castro CG, Rios AF (2000) The water masses along the western boundary of the south and equatorial Atlantic. Progress in Oceanography, 47, 69–98.

Menke W (1984) Geophysical Data Analysis: Discrete Inverse Theory. Academic Press, San Diego, 451 pp.

Metzger EJ, Hurlburt HE (1996) Coupled dynamics of the South China Sea, the Sulu Sea, and the Pacific Ocean. Journal of Geophysical Research, 101, 12331–12352.

Miller JR (1976) The salinity effect in a mixed layer ocean model. Journal of Physical Oceanography, 6, 29–35.

Mitta T, Ogawa Y (1984) Tsushima currents measured with current meters and drifters. In: Ocean Hydrodynamics of the Japan and East China Seas, edited by T. Ichiye, Elsevier Oceanography Series, 39, Amsterdam, pp. 67–76.

Miyazaki M (1952) The heat budget of the Japan Sea. Bulletin of Hokkaido Regional Fishery Research Laboratory, 4, 1–54 (in Japanese with English abstract).

Miyazaki M (1953) On the water masses of the Japan Sea. Bulletin of Hokkaido Regional Fishery Research Laboratory, 7, 1–65 (in Japanese with English abstract).

Miyazaki M, Abe S (1960) On the water masses in the Tsushima Current area. Journal of Oceanographic Society of Japan, 16, 19–28 (in Japanese with English abstract).

Monterey G, Levitus S (1997) Seasonal variability of mixed layer depth for the world ocean. NOAA Atlas NESDIS 14, U.S. Government Printing Office, Washington, DC, 100 pp.

Moore RM, Wallace DWR (1988) A relationship between heat transfer to sea ice and temperature-salinity properties of Arctic Ocean waters. Journal of Geophysical Research, 93, 565–571.

Morey SL, Shriver JF, O'Brien JJ (1999) The effects of Halmahera on the Indonesian throughflow. Journal of Geophysical Research, 104, 23281–23296.

Moriyasu S (1972) The Tsushima current. In: Kuroshio, Its Physical Aspects, edited by Stommel, Yoshida, University of Tokyo Press, Tokyo, pp. 353–369.

Mosby H (1934) The waters of the Atlantic Antarctic Ocean. Scientific Results of the Norwegian Antarctic Expeditions, 1927–1928, 11, Oslo, 1–131.

Mountain DG, Coachman LK, Aagaard K (1976) On the flow through Barrow Canyon. Journal of Physical Oceanography, 6, 461–470.

Muench RD, Gordon AL (1995) Circulation and transport of water along the western Weddell Sea margin. Journal of Geophysical Research, 100, 18503–18515.

Munk WH (1950) On the wind-driven ocean circulation. Journal of Meteorology, 7, 79–93.
Needler GT (1982) On determining the velocity from the density field-including a closed form. Ocean Modelling, 46, unpublished manuscript.
Nelson DM, Smith WO Jr, Gordon LI, Huber BA (1987) Spring distributions of density, nutrients and phytoplankton biomasses in the ice-edge zone of the Weddell/Scotia Sea. Journal of Geophysical Research, 92, 7181–7190.
Nitani H (1970) Oceanographic conditions in the sea east of Philippines and Luzon Strait in summer of 1965 and 1966. In: The Kuroshio-A Symposium on Japan Current, edited by JD. Marr, East–West Press, Honolulu, Hawaii, pp. 213–232.
Nowlin WD Jr, Klinck JM (1986) The physics of Circumpolar Current. Reviews of Geophysics and Space Physics, 24, 469–491.
Nowlin WD Jr, Whitworth T, Pillsbury RD (1977) Structure and transport of the Antarctic Circumpolar Current at Drake Passage from short term measurements. Journal of Physical Oceanography, 7, 788–802.
Ohlmann JC, Siegel DA, Gautier C (1996) Ocean mixed layer radiant heating and solar penetration: A global analysis. Journal of Climate, 9, 2265–2280.
Olbers DJ, Wenzel M, Willbrand J (1985) The inference of North Atlantic circulation patterns from climatological hydrographic data. Review of Geophysics, 23, 313–356.
Olson DB, Podesta GP, Evans RH, Brown OB (1988) Temporal variations in the separation of Brazil and Malvinas currents. Deep Sea Research, 35, 1971–1990.
Oppenheim AV, Schafer RW (1975) Digital Signal Processing. Prentice-Hall, Englewood Cliffs, NJ, pp. 211–212.
Ozsoy E, Hecht A, Unluata U (1989) Circulation and hydrology of the Levantine Basin. Progress in Oceanography, 22, 125–170.
Pacanowski RC, Dixon KW, Rosati A (1991) GFDL Modular Ocean Model, Users Guide Version 1.0, GFDL Technical Report, 2, 46 pp.
Park S, Chu PC (2006a) Interannual SST variability in the Japan/East Sea and relationship with environmental variables. Journal of Oceanography, 62(2), 115.
Park S, Chu PC (2006b) Thermal and haline fronts in the Yellow/East China Sea: Surface and subsurface seasonality comparison. Journal of Oceanography, in press.
Partos P, Piccolo MC (1988) Hydrography of the Argentine continental shelf between 38°S and 42°S. Continental Shelf Research, 8, 1043–1056.
Pedlosky J (1986) Thermocline theories. In: General Circulation of the Ocean, edited by HDL. Abardanel, WR. Young, Springer, New York Berlin Heidelberg, 55–101.
Pedlosky J (1987) Geophysical Fluid Dynamics. Springer, New York Berlin Heidelberg, 710 pp.
Perkin RG, Lewis EL (1984) Mixing in the West Spitzbergen Current. Journal of Physical Oceanography, 14, 1315–1325.

Peterson RG, Stramma L (1991) Upper-level circulation in the South Atlantic Ocean. Progress in Oceanography, 26, 1–73.

Peterson RG, Whitworth T III (1989) The subantarctic and Polar Fronts in relation to deep water masses through the southwestern Atlantic. Journal of Geophysical Research, 94, 10817–10838.

Phoebus PA (1988) Improvements to the data selection algorithms in the Optimum Thermal Interpolation System (OTIS). Naval Ocean Research and Development Activity Report, No. 239.

Pickard GL, Emery WJ (1990) Descriptive Physical Oceanography, An Introduction, 5th Edition. Pergamon Press, Oxford, pp. 173–76.

Pillsbury RD, Jacobs SS (1985) Preliminary observations from long-term current meter moorings near the Ross Ice Shelf. Oceanography of the Antarctic Continental Shelf, edited by SS. Jacobs, Antarctic Research Series, Vol. 43, American Geophysical Union Washington, DC, pp. 87–107.

Podesta GP, Brown OB, Evans RH (1991) The annual cycle of satellite-derived sea surface temperature in the Southwestern Atlantic Ocean. Journal of Climate, 4, 457–467.

Preller RH, Hogan PJ (1998) Oceanography of the Sea of Okhotsk and the Japan/East Seas. In: The Sea, Vol. 11, edited by AR. Robinson, KK. Brink, Wiley, New York, pp. 429–481.

Press WH, Flannery BP, Teukolsky SA, Vetterling WT (1986) Numerical Recipes – the Art of Scientific Computing. Cambridge University Press, Cambridge, UK, pp. 660–667.

Price JF, Baringer MO (1994) Outflows and deep water production by marginal seas. Progress in Oceanography, 33, 161–200.

Price JF, Weller RA, Pinkel R (1986) Diurnal cycling: Observations and models of the upper ocean response to diurnal heating, cooling and wind mixing. Journal of Geophysical Research, 91, 8411–8427.

Qiu B (1999) Seasonal eddy field modulation of the North Pacific Subtropical Countercurrent: TOPEX/Poseidon observations and theory. Journal of Physical Oceanography, 29, 2471–2486.

Qiu DZ, Huang YT, Chen LM, Guo ZX (1985) Circulation structures in the studied waters. Comprehensive Investigations and Studies of the South China Sea, Vol. 2, Science Press, Beijing, pp. 204–230 (in Chinese).

Qu TD (2000) Upper layer circulation in the South China Sea. Journal of Physical Oceanography, 30, 1450–1460.

Qu TD, Mitsudera H, Yamagata T (2000) Intrusion of the North Pacific waters into the South China Sea. Journal of Geophysical Research, 105, 6415–6424.

Quadfasel D, Meincke J (1987) Note on the thermal structure of the Greenland Sea gyres. Deep Sea Research, 35, 1143–1150.

Quadfasel D, Ungewiß M (1988) MIZEX 87 – RV VALDIVIA cruise 54. CTD-observations in the Greenland Sea. Technical Reports, Institut für Meereskunde, Hamburg, Band-Nr. 88–5.

Quadfasel D, Gascard JC, Koltermann KP (1987) Large-scale oceanography in Fram Strait during the 1984 marginal ice zone experiment. Journal of Geophysical Research, 92, 6719–6728.

Reid JL (1989) On the total geostrophic circulation of the South Atlantic Ocean: flow patterns, tracers, and transports. Progress in Oceanography, 23, 149–244.

Reid JL (1994) On the total geostrophic circulation of the North Atlantic Ocean: flow patterns, tracers, and transports. Progress in Oceanography, 33, 1–92.

Reid JL (1997) On the total geostrophic circulation of the Pacific Ocean: flow patterns, tracers, and transports. Progress in Oceanography, 39, 263–352.

Roden GL (1980) On the variability of surface temperature fronts in the western Pacific, as detected by satellite. Journal of Geophysical Research, 85, 2704–2710.

Rosati A, Gudgel R, Miyakoda K (1996) Global ocean data assimilation system. Modern Approaches to Data Assimilation in Ocean Modeling, edited by P. Malanotte-Rizzoli, Elsevier, Amsterdam, pp. 181–203.

Rudels B (1987) On the mass balance of the polar ocean with special emphasis on the Fram Strait. Skr Nor Polarinst, 188, 1–53.

Saraceno M, Provost C, Piola AR, Bava J, Gagliardini A (2004) Brazil Malvinas Frontal System as seen from 9 years of advanced very high resolution radiometer data. Journal of Geophysical Research, 109, doi 10.1029/2003JC002122.

Schmitz W Jr (1996a) On the world ocean circulation, Vol. 1, Some global features/North Atlantic circulation. Woods Hole Oceanographic Institution Technical Report, WHOI-96–03, 141 pp.

Schmitz W Jr (1996b) On the world ocean circulation, Vol. 2, The Pacific and Indian Oceans/global update. Woods Hole Oceanographic Institution Technical Report, WHOI-96–08, 237 pp.

Schmitz W Jr, McCartney MS (1993) On the North Atlantic circulation. Review of Geophysics, 31, 29–49.

Schmitz W Jr, Richardson PL (1991) On the sources of the Florida Current. Deep Sea Research, 38(Suppl. 1), S389–S409.

Schott F, Stommel H (1978) Beta spirals and absolute velocities in different oceans. Deep Sea Research, 25, 961–1010.

SCSMEX Science Working Group (1995) The South China Sea Monsoon Experiment (SCSMEX) Science Plan. NASA/Goddard Space Flight Center, Greenbelt, Maryland, 65 pp.

Sekine Y (1986) Wind-driven circulation in the Japan Sea and its influence on the branching of the Tsushima Current. Progress in Oceanography, 17, 297–312.

Semtner AJ, Chervin RM (1992) Ocean general circulation from a global eddy-resolving model. Journal of Geophysical Research, 97, 5493–5550.

Senjyu T (1999) The Japan Sea intermediate water: its characteristics and circulation. Journal of Oceanography, 55, 111–122.
Seung YH (1992) A simple model for separation of East Korean Warm Current and formation of the North Korean Cold Current. Journal of Oceanological Society of Korea, 27, 189–196.
Seung YH, Yoon JH (1995) Some features of winter convection in the Japan Sea. Journal of Oceanography, 51, 61–73.
Shaw PT (1989) The intrusion of water masses into the sea southwest of Taiwan. Journal of Geophysical Research, 94, 18213–18226.
Shaw PT (1991) The seasonal variation of the intrusion of the Philippine Sea water into the South China Sea. Journal of Geophysical Research, 96, 821–827.
Shaw PT, Chao SY (1994) Surface circulation in the South China Sea. Deep Sea Research, Part 1, 41, 1663–1683.
Shriver JF, Hurlburt HE (1997) The contribution of the global thermohaline circulation to the Pacific to Indian Ocean Throughflow via Indonesia. Journal of Geophysical Research, 102, 5491–5511.
Soong YS, Hu JH, Ho CR, Niiler PP (1995) Cold-core eddy detected in South China Sea. EOS Transaction, American Geophysical Union, 76, 345–347.
South China Sea Institute of Oceanology, 1985: Integrated Investigation Report on Sea Area of the South China Sea, Vol. 2, Science Press, Beijing, pp. 183–231 (in Chinese).
Spall MA (1991) A diagnostic study of wind- and buoyancy-driven North Atlantic circulation. Journal of Geophysical Research, 96, 18509–18518.
Sprintall J, Meyers G (1991) An optimal XBT sampling network for the eastern Pacific Ocean. Journal of Geophysical Research, 96, 10539–10552.
Sprintall J, Tomczak M (1992) Evidence of barrier layer in the surface layer of tropics. Journal of Geophysical Research, 97, 7305–7316.
Stommel H (1957) A survey of ocean current theory. Deep Sea Research, 4, 149–184.
Stommel H, Schott F (1977) The beta spiral and the determination of the absolute velocity field from hydrographic station data. Deep Sea Research, 24, 325–329.
Strakhov VN (1991) Method for filtration of the linear algebraic systems as basis for the solutions of linear problem in the gravimetry and magnitometry. Dok Akad Nauk SSSR, 320, 590–599 (in Russian).
Stramma L, Ikeda Y, Peterson RG (1990) Geostrophic transport in the Brazil Current region north of 20°S. Deep Sea Research, 37, 1875–1886.
Suda K, Hidaka K (1932) The results of the oceanographic observations on board R.M.S. 'Syunpu Maru' in the southern part of the Japan Sea in the summer of 1930. Journal of Oceanography of Imperial Marine Observatory, 4, 1–174 (in Japanese).
Suda K, Hidaka K, Matsudaira Y, Kurashige E, Kawasaki H, Kubo T (1932) The results of the oceanographic observations on board R.M.S. 'Syunpu Maru' in the southern part of the Japan Sea in the summer of 1929, Part

1. Journal of Oceanography of Imperial Marine Observatory, 3, 291–375 (in Japanese).
Sverdrup HU (1947) Wind driven currents in a baroclinic ocean with applications to the equatorial currents of the eastern Pacific. Proceedings of National Academy pf Sciences USA, 33, 318–326.
Sverdrup HU (1953) The currents of the coast of Queen Maud Land. Norsk Geografisk Tidsskrift, 14, 239–249.
Sverdrup HU, Johnson MW, Fleming RH (1942) The Oceans: Their Physics, Chemistry, and General Biology, Prentice-Hall, Englewood, NJ, 1087 pp.
Swenson MS, Hansen DV (1999) Tropical Pacific Ocean mixed layer heat budget: the Pacific cold tongue. Journal of Physical Oceanography, 29, 69–81.
Swift JH, Aagaard K (1981) Seasonal transitions and water mass formation in the Iceland and Greenland seas. Deep Sea Research, 28, 1107–1129.
Swift JH, Takahashi T, Livingston (1983) The contribution of the Greenland and Barents seas to the deep water of the Arctic Ocean. Journal of Geophysical Research, 88, 5981–5986.
Talley LD (1988) Potential vorticity distribution in the North Pacific. Journal of Physical Oceanography, 18, 89–106.
Talley LD (1993) Distribution and formation of North Pacific Intermediate Water. Journal of Physical Oceanography, 23, 517–537.
Tao SY, Chen LX (1987) A review of recent research on the east Asian summer monsoon in China. In: Monsoon Meteorology, edited by C-P. Chang, TN. Krishnamurti, Oxford University Press, pp. 60–92.
Teague WJ, Carron MJ, Hogan PJ (1990) A comparison between the Generalized Digital Environmental Model and Levitus climatology, Journal of Geophysical Research, 95, 7167–7183.
Tikhonov AN, Arsenin VL (1979) Methods for Solving Ill-posed Problems. Nauka, Moscow, pp. 285.
Tikhonov AN, Goncharsky AV, Stepanov VV, Yagola AG (1990) Numerical Methods for Ill-Posed Problems. Nauka, Moscow, 229 pp.
Timofeyev VT (1962) The movement of Atlantic water and heat into the Arctic sea basin. Deep Sea Research, 9, 263–269.
Toba Y, Tomizawa K, Kurasawa Y, Hanawa K (1982) Seasonal and year-to-year variability of the Tsushima–Tsugaru Warm Current system with its possible cause. La Mer, 20, 41–51.
Toole JM (1987) WOCE, interbasin exchanges, and marginal sea overflows. EOS Transactions, American Geophysical Union, 68(1), 2–3, 11.
Tschiya M (1989) Circulation of the Antarctic intermediate water in the North Atlantic Ocean. Journal of Marine Research, 47, 747–755.
Tschiya M (1991) Flow path of the Antarctic intermediate water in the western equatorial south Pacific Ocean. Deep Sea Research, 38(Suppl. 1), S273–S279.
Tully JP, Giovando LF (1963) Seasonal temperature structure in the eastern subarctic Pacific Ocean. In: Marine Distributions, edited by MJ. Dun-

bar, Royal Society of Canadian Special Publications, No. 5, University of Toronto Press, Toronto, pp. 10–36.
Uda M (1934) The results of simultaneous oceanographic investigations in the Japan Sea and its adjacent waters in May and June. Journal of Imperial Fishery Experiment Stations, 5, 57–190 (in Japanese).
Uda M (1955) On the Subtropical Convergence and the currents in the Northwestern Pacific. Records of Oceanographic Works in Japan, 2, 141–150.
Uda M, Hasunuma K (1969) The eastward Subtropical Countercurrent in the western North Pacific Ocean. Journal of Oceanographic Society of Japan, 25, 201–210.
Uda M, Nakao T (1972) Water masses and currents in the South China Sea and their seasonal changes. Paper presented at the 3rd Cooperative Study of the Kuroshio and Adjacent Regions (CSK) Symposium, UNESCO, Bangkok, Thailand.
Unidata (2004) NetCDF Documentation. This document can be downloaded from the website: http://my.unidata.ucar.edu/content/software/netcdf/docs.html.
Untersteiner N (1988) On the ice and heat balance in Fram Strait. Journal of Geophysical Research, 92, 527–532.
van Aken HM, Quadfasel D, Warpakowski A (1991) The Arctic front in the Greenland Sea during February 1989: hydrographic and biological observations. Journal of Geophysical Research, 96, 4739–4750.
van Loon H (1984) Climates of the Oceans, World Survey of Climatology, 15, 453–458.
Vapnik VN, Chervonenkis AY (1974a) On the method of ordered risk minimization, Part-1. Avtomatika i Telemekhanika, 8, 21–30, (in Russian).
Vapnik VN, Chervonenkis AY (1974b) On the method of ordered risk minimization Part-2. Avtomatika i Telemekhanika, 9, 29–39 (in Russian).
Vinje TK, Finnekasa O (1986) The ice transport through the Fram Strait. Skr Nor Polarinst, 186, 1–39.
Wadhams P (1983) Sea-ice thickness distribution in Fram Strait. Nature, 305, 108–111.
Wadhams P, Gill AE, Linden PF (1979) Transect by submarine of the East Greenland Polar Front. Deep Sea Research, 26, 1311–1327.
Walsh JE, Johnson CM (1979) Interannual atmospheric variability and associated fluctuations in Arctic Sea ice extent. Journal of geophysical Research, 84, 6915–6928.
Wang L, Koblinsky C, Howden S, Huang N (1999) Interannual variability in the South China Sea from expandable bathythermograph data. Journal of Geophysical Research, 104, 23509–23523.
Wang GH, Su JL, Chu PC (2003) Mesoscale eddies in the South China Sea observed with altimeter data. Geophysical Research Letters, 30(21), doi: 10.1029/2003GL018532.

White W, Hasunuma K, Solomon H (1978) Large-scale season and secular variability of the Subtropical Front in the western North Pacific from 1954 to 1974. Journal of Geophysical Research, 83, 4531–4544.

White WB, Meyers G, Hasunuma K (1982) Space/time statistics of short-term climatic variability in the western North Pacific. Journal of Geophysical Research, 87, 1979–1989.

Whitworth T III (1983) Monitroing the transport of the Antarctic Circumpolar Current at Drake Passage. Journal of Physical Oceanography, 13, 2045–2057.

Whitworth T III, Peterson RG (1985) Volume transport of the Antarctic Circumpolar Current from bottom pressure measurements. Journal of Physical Oceanography, 15, 810–816.

Wijffels SE, Firing E, Toole J (1995) The mean structure and variability of the Mindanao current at 8°N. Journal of Geophysical Research, 100, 18421–18435.

Wilson HR, Rees NW (2000) Classification of mesoscale features in the Brazil–Falkland Current confluence zone. Progress in Oceanography, 45, 415–426.

Worthington LV (1976) On the North Atlantic circulation. The Johns Hopkins Oceanographic Studies, 6, The Johns Hopkins University Press, 110 pp.

Worthington LV, Wright WR (1970) North Atlantic Atlas of Potential Temperature and Salinity in the Deep Water, Including Temperature, Salinity, and Oxygen Profiles from Erika Dan Cruise of 1962. Woods Hole Oceanographic Institution Atlas Series 2, 58 plates.

Wright WR, Worthington LV (1970) The water masses of the North Atlantic Ocean: a volumetric census of temperature and salinity. Series Atlas in Maine Environment, 19, 8 pp, 7 plates.

Wunsch C (1978) The general circulation of the North Atlantic west of 50°W determined from inverse method. Reviews of Geophysics, 16, 583–620.

Wunsch C (1996) The ocean circulation inverse problem. Cambridge University Press, Cambridge, UK, 442 pp.

Wunsch C, Grant B (1982) Towards the general circulation of the North Atlantic Ocean. Progress in Oceanography, 11, 1–59.

Wylie CR Jr (1975) Advanced Engineering Mathematics. McGraw-Hill, New York, 156–160.

Wyrtki K (1961a) Scientific results of marine investigations of the South China Sea and Gulf of Thailand 1959–1961. Naga Report, 2, Scripps Institution of Oceanography, University of California, San Diego, pp. 164–169.

Wyrtki K (1961b) Physical oceanography of the South east Asian Waters. Naga Report, 2, Scripps Institution of Oceanography, University of California, San Diego, pp. 195.

Wyrtki K (1964) The thermal structure of the eastern Pacific Ocean. Deustche Hydrogr. Zeit., Suppl. Ser. A, 8, 6–84.

Xu Q (1992) Ageostrophic pseudovorticity and geostrophic c-vector forcing – a new look at the Q vector in three dimensions. Journal of the Atmospheric Sciences, 49, 981–990.

Xu XZ, Qiu Z, Chen HC (1982) The general description of the horizontal circulation in the South China Sea. Proceedings of the Symposium of the Chinese Society of Marine Hydrology and Meteorology, Chinese Society of Oceanology and Limnology. Science Press, Beijing, pp. 119–127 (in Chinese with English abstract).

Yi SU (1966) Seasonal and secular variations of the water volume transport across the Korea Strait. Journal of Oceanological Society of Korea, 1, 7–13.

Yoon JH (1982) Numerical experiment on the circulation in the Japan Sea Part, III. Mechanism of the Nearshore Branch of the Tsushima Current. Journal of Oceanographic Society of Japan, 38, 125–130.

Yoshida K, Kidokoro T (1967a) A subtropical countercurrent in the North Pacific – an eastward flow near the Subtropical Convergence. Journal of Oceanographic Society of Japan, 23, 88–91.

Yoshida K, Kidokoro T (1967b) A subtropical countercurrent (II) – a prediction of eastward flows at lower subtropical latitudes. Journal of Oceanographic Society of Japan, 23, 231–246.

You Y (1995) Salinity variability and its role in the barrier layer formation during TOGA-COARE. Journal of Physical Oceanography, 25, 2778–2807.

You Y (1998) Rain-formed barrier layer of the western equatorial Pacific warm pool: A case study. Journal of Geophysical Research, 103, 5361–5378.

Zavilov PO, Wainer I, Absy JM (1999) Sea surface temperature variability off southern Brazil and Uruguay as revealed from historical data since 1854. Journal of Geophysical Research, 105, 21021–21032.

Zhou FX, Shen JJ, Berestov AL, Marushkevich AD (1995) Seasonal features of large-scale geostrophic circulations in the South China Sea. Tropical Oceanology, 14(4), 9–14 (in Chinese with English abstract).

Index

advection type T profiles, 27, 28, 29, 30, 32, 75
ageostrophic velocity, 229, 238
Alaskan Coastal Water, 28
Alpha-Mendeleyev Ridge, 389
American Continent, 221
Antarctic Circumpolar Current, 123, 124, 223, 376, 377, 379, 380, 381, 383, 385
Antarctic Coastal Current, 379, 382
Antarctic Intermediate Water, 255, 258, 260, 261
Antarctic Peninsula, 385
Antarctica, 208, 221, 379
anticyclonic eddy, 191, 193, 215, 217, 255, 260, 278, 292, 294, 305, 326, 327, 328, 329, 343, 345, 357, 359, 362, 370
Antilles Current, 123, 124, 367
Arctic front, 232, 233, 238, 393
Arctic Intermediate Water, 255, 258, 260, 261, 393, 394, 404
Arctic Mediterranean seas, 389, 391, 392, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404
Arctic Ocean, 77, 78, 232, 389, 390, 392, 393, 395, 397, 401, 403, 404, 405, 433
Arctic Surface Water, 391, 393
Arctic Water, 232, 376, 391, 392, 395
Asian monsoon, 17, 83
Atlantic Water, 78, 232, 391, 393, 395, 397, 401, 402, 403, 404, 405
attribute, 242, 243, 244, 245, 374
Australia, 83, 208, 221, 254, 273, 274, 359
Australian Gyre, 359, 361, 363
Australian Mediterranean, 359, 361
autocorrelation function, 63, 64, 65, 66, 67, 68, 69, 76, 82, 527
AXBT data, 316, 318, 319, 334, 348, 349
AXCTD data, 13, 333, 334

balanced data assimilation, 407, 410, 413, 414
baroclinic instability, 279, 316, 363
barrier layer, 40, 42, 43, 45, 46, 47, 48, 49, 50, 51
basin-wide cyclonic gyre, 139, 140, 175, 176, 294
basis functions, 88, 93, 100
Beaufort Gyre, 403
Beaufort Sea, 23, 25, 26, 28, 29, 404
Beaufort Sea Deep Water, 30
Beaufort Sea Shelf Water, 29
Beaufort Undercurrent, 404, 405
Bering Sea Water, 28, 29, 30
Bering Strait, 28, 222, 390, 392
Bernoulli method, 4
beta-spiral method, 4, 7, 8, 109, 368, 370
bi-modality, 107
bin method, 63, 76
Black Sea, 88, 92, 93, 95
Borneo-Palawan islands, 287, 288

bottom topography, 45, 54, 132, 141, 142, 165, 189, 203, 204, 255, 277, 281, 330, 348, 401, 418, 419, 436
Boussinesq approximation, 1
box model, 4, 5, 7, 8, 109, 187, 188
Brazil Current, 373, 374, 375, 376, 377
Brazil-Malvinas Confluence, 365, 373, 374, 375, 376, 377
Brazil-Malvinas confluence zone, 365, 374
Brunt-Vaisala frequency, 230
buoyancy forcing, 237

Celebes Sea, 43, 44, 45, 48, 49, 50, 261, 268, 269, 271, 272
Chukchi Sea, 16, 23, 24, 25, 26, 389
Circulation Research of the East Asian Marginal Seas, 153
coefficient matrix, 89, 90, 92, 94, 107
cold-core eddy, 167, 282, 316, 336, 337, 342, 344, 351, 352, 359
complex empirical orthogonal function, 144
composite analysis, 143, 167, 282, 284, 316
Comprehensive Ocean-Atmosphere Data Set, 48, 204
condition number, 89, 92, 94, 98
continental shelf, 16, 17, 23, 56, 65, 289, 292, 374, 378, 383, 389, 390, 429
continuity equation, 3, 110, 111, 114
coordinate variables, 243, 244
Coriolis force, 1, 2, 383
Coriolis parameter, 2, 3, 204, 408, 409
correlation coefficient, 58, 59, 62, 63
cost function, 88, 107, 185, 188
Cross-Basin Current, 291, 294, 301, 302, 305, 306, 316
cubic spline, 113, 284
curve-fitting model, 33, 35, 296
C-vector, 229, 230, 231, 233, 235, 237
cyclonic eddy, 143, 176, 178, 215, 217, 255, 260, 261, 267, 292, 295, 327, 329, 330, 332, 335, 343, 357, 359, 370, 383

data assimilation, 81, 407, 408, 410, 412, 415
data nudging, 407
data variables, 243
deep layer, 19, 21, 25, 52, 53, 58, 87, 181, 261, 265, 267, 271, 371, 423
deep layer jet-core, 261
deep sub model, 33, 35
deep-mixing type T-profiles, 29
deep-mixing type S-profiles, 28
density jump, 44
detrainment regime, 43, 44, 49, 50
diagnostic initialization, 415, 416, 417, 418, 421, 423, 424
difference criterion, 35, 76
dimension ratio, 89, 94
divergence equation, 409
Drake Passage, 208, 381, 382, 385

East Australia Current, 359
East Greenland Current, 232, 391, 395, 397, 399, 400, 401, 403
East Korean Bay, 143, 350
East Korean Warm Current, 142, 143, 160, 189, 191
eddy kinetic energy, 277, 346, 347, 348
eigenfunctions, 88, 93, 107
Ekman convergence, 278
Ekman drift, 267
Ekman flow, 6, 238
Ekman Number, 2
entrainment regime, 44, 49
entrainment zone, 15, 19, 21, 26, 31, 52, 87, 230
equatorial Rossby waves, 39
Eurasian basin, 389, 390, 395, 403
EWG atlas, 391, 392, 394, 395
extraction, 245
extra-equatorial region, 2, 3, 203, 204, 205
extremely strong 'sources/sinks', 418, 421, 422

first guess, 21, 22, 81, 82, 105, 211
first necessary condition, 9, 119, 128, 136
Fourier series, 87, 91, 93
f-plane, 119, 408, 409
Fram Strait, 232, 233, 234, 235, 389, 393, 394, 397, 401, 402, 403
F-Test, 73

Gaussian model, 71, 72, 73, 76
Gaussian-type random variable, 223
GDEM climatology, 144, 171
GDEM data, 81, 83, 84, 153, 167, 246, 247, 296, 425
geostrophic advection, 229
geostrophic balance, 7, 109, 111, 114, 126, 201, 263, 410, 412
geostrophic convergence, 278
geostrophic forcing, 230, 233, 237, 239
geostrophic shear, 3, 4, 127, 128, 332
geostrophic velocity, 2, 3, 8, 81, 109, 174, 201, 277
GIN Sea, 232, 389, 393, 395, 396, 397, 398, 399
global conveyor belt, 369
global heat storage, 102, 103, 104, 246
gradient criterion, 35, 36
gradient space, 19, 20, 540
Greenland-Scotland Ridge, 389, 393, 396
Greenland Fracture Zone, 397
Gulf Stream, 36, 123, 124, 365, 366, 367, 368, 371, 372, 373, 377

Hainan Island, 178, 331
Halmahera Eddy, 249, 253, 255, 258, 260, 265, 267, 273, 362
Halmahera Sea, 255, 256, 270, 273
halocline, 15, 25, 26, 27, 29, 30, 31, 32, 87, 156, 157, 307, 338, 392, 547
High Salinity Intermediate Water, 156, 157, 160
histogram, 79, 86
horizontal diffusivity, 2, 131
horizontal Laplace operator, 88
Hovgaard cell, 236
Hovgaard Fracture Zone, 233, 234, 236
hydrostatic balance, 1, 3, 7, 9, 109, 126, 229
hydrostatic pressure, 113

ice breeze, 394
ice drift, 394
ice freezing, 24, 25, 26
ice melting, 25, 26, 392
ill-posed algebraic equation, 98, 107
Indian Ocean, 253, 261, 263, 271, 272, 381
Indonesian Throughflow, 253, 271, 272
inertial-gravity mode, 410
isopycnal coordinate system, 109, 110, 111, 114, 119, 126, 127, 165, 173, 247
isothermal/isohaline structure, 25
iteration, 21, 23, 55, 82, 98, 99, 211, 212, 221, 479

Jacobian, 117, 119, 123
Japan Basin, 141, 142, 143, 154, 155, 156, 157, 160, 191, 193
Japan Nearshore Branch, 143, 161, 162, 189, 191, 193
Japan Sea Intermediate Water, 143, 156, 419
Japan Sea Proper Water, 142, 143
Japan/East Sea, 140, 141, 142, 144, 183, 193, 197, 198, 348, 355, 418, 419, 421, 426, 565

Kara Sea, 389
Kelvin waves, 39
Knipovich cell, 235, 236
Knipovich Ridge, 232, 234, 235, 236, 393
Kuroshio Current, 249, 253, 278, 332
Kuroshio intrusion, 166, 167, 172, 175, 178, 180, 181, 275, 285, 287, 289, 292, 294, 295, 343
Kuroshio water, 17, 153, 169, 172, 175, 178, 181, 274, 293, 295, 331, 332, 343

Labrador Basin cyclonic gyre, 365
Labrador and Irminger Currents, 366
Lagrangian parameter, 186, 198
lateral boundary transport, 427, 428, 429
lateral mixing, 112
least square difference, 82, 88, 105
least square error, 129
level of no motion, 4, 9, 235, 326, 333
Liyue Bank, 168, 171, 331
Lomok Strait, 272
Lomonosov Ridge, 389, 390
Lorenz system, 94, 95, 98, 415

Luzon Strait, 56, 165, 166, 167, 169, 172, 174, 175, 178, 180, 181, 182, 183, 274, 275, 284, 287, 289, 292, 293, 294, 295, 301, 332, 336, 337, 343

Mackenzie Canyon, 23
Mackenzie River, 390
Madagascar, 221
Malvinas Current, 375, 376, 377
mass conservation, 3, 6, 7, 9, 109, 185, 186, 188
MCSST data, 296
mean kinetic energy, 345, 347, 348
mid-depth of North Atlantic, 99
middepth sub model, 33, 34, 35
Mindanao Current, 253, 257, 258, 260, 261, 263, 265, 267, 269, 272
Mindanao Eddy, 249, 253, 255, 258, 260, 261, 263, 267, 272, 273
Mindanao Island, 255, 258, 260, 261, 262, 267
mixed layer, 14, 15, 17, 19, 21, 24, 25, 26, 29, 30, 31, 32, 35, 39, 40, 43, 44, 52, 56, 58, 59, 65, 66, 73, 74, 81, 87, 149, 167, 230, 231, 238, 310, 318, 319, 321, 322, 334, 336, 337, 349, 352, 544
mixed layer depth, 17, 21, 30, 35, 81, 149, 231, 544
MODAS, 81, 82, 83, 84, 85, 86, 87, 106, 413, 415
model uncertainty, 428
modular ocean model, 131, 132, 133, 134, 135, 136, 137, 138, 139
Molucca Sea, 269
Montgomery potential, 109, 178
MOODS data, 11, 24, 36, 43, 54, 55, 56, 60, 66, 144, 147, 167, 282, 283, 284, 323, 324, 355, 356
multi-eddy structure, 191, 292, 305, 316, 318, 348, 349
multi-layer structure, 19, 69, 70, 74
multi-time scale, 59, 60, 62

Nansen-Gakkel Ridge, 389
NCEP data, 167, 287
Needler's formula, 8, 9, 116, 120
net fresh water flux, 48, 49, 426
net heat flux, 48, 49, 50, 426, 427
netCDF, vi, 241, 242, 243, 244, 245, 246, 432, 433, 440, 442
Neumann boundary condition, 88
neutral tangent plane, 112
New Guinea, 221, 249, 253, 260
New Guinea Coastal Current, 253, 255, 257, 272
New Guinea Coastal Undercurrent, 253, 255, 257, 258, 260, 263, 265, 267
New Zealand, 259, 365
Newtonian nudging, 408, 410, 412
noise level, 102, 217, 219, 220, 221
noise-to-signal ratio, 89, 90, 94, 98, 101, 107
North Atlantic Gyre, 123, 124
North Atlantic Ocean, 36, 39, 116, 365, 366, 367, 369, 370, 377, 431, 433
North Atlantic Water, 232
North Equatorial Counter Current, 253, 255, 256, 257, 258, 260, 261, 263, 265, 266, 267, 268, 270, 272
North Equatorial Current, 166, 249, 253, 272, 278, 280, 281, 301
North Korean Cold Current, 143
North Pacific Intermediate Water, 255, 258, 260
North Pacific Tropical Water, 256
Norwegian Atlantic Current, 233, 391, 397, 399
null hypothesis, 33, 59, 73

Oki Gunto eddy, 351, 355
optimal interpolation, 37, 77, 81, 283, 407
optimal mode truncation, 88
optimal spectral decomposition, 77, 87, 88, 89, 90, 91, 92, 93, 94
out-of-phase variation, 162, 191, 193, 265

PacificOcean, 8, 36, 39, 42, 59, 78, 141, 142, 165, 208, 249, 251, 255, 259, 271, 272, 275, 281, 285, 295, 301, 348, 370, 385
pair number, 6, 64, 66, 67
parameter analysis, 15 ,77

parametric model, 16, 17, 19, 20, 21, 23, 25, 27, 29, 31, 32, 33, 35, 36, 52, 53, 54, 55, 80, 86, 527
perfect vector, 115
Perturbed Lorenz Attractor, 94
Philippine Sea, 169, 172, 178, 180, 275, 295
Philippines, 17, 165, 249, 253, 272, 273
Point Barrow, 28
Poisson equation, 213, 223
Polar Front, 191, 350, 351, 352, 355, 379, 382, 393
Polar Front Current, 181, 191, 193
Polar Intermediate Water, 393
Polar Water, 232, 391, 392
potential density, 113, 116, 122, 123, 133, 134, 233, 234, 436, 442, 447, 472
potential vorticity, 246, 247, 280, 281, 292, 369, 506, 518
profile data analysis, 15
pseudo potential vorticity, 112
pseudovorticity, 229, 230, 234, 235, 236
P-spiral, 117, 118, 119, 120, 121, 123, 124, 128, 138
P-vector inverse method, 77, 116, 127, 137, 140, 165, 185, 201, 204, 291, 331, 333, 349, 401, 407, 413, 415, 428

q-isoline, 119, 173, 175

rain-formed mechanism, 44, 49, 50
recirculation, 117, 143, 260, 267, 278, 280, 366, 367, 368, 371, 372, 374, 376, 377, 393, 401, 402, 403, 404, 405
reconstruction, 91, 92, 93, 94, 99
reference-level velocity, 3, 4, 5, 6, 7, 8
regression model, 60, 62
Reynolds stress, 1, 2, 204
river run-off, 17, 26, 65, 285
root mean square error, 215, 219, 428, 539
Ross Gyre, 385
Ross Sea, 379, 382, 383, 385, 386
Rossby mode, 410
rotation matrix method, 89, 90, 91, 92, 93, 94

salinity maximum, 157, 160, 256, 306, 310, 341, 393
salinity minimum, 143, 144, 157, 160
satellite remote-sensing, 52
Savu Sea, 272
scatter diagrams, 47, 49, 51, 58, 84, 85
SCSMEX, 82, 83, 84, 85, 333, 345
seasonal adjustment, 26
second necessary condition, 9, 119, 128, 137
secondary circulation, 229, 236
semi-isopycnal coordinate system, 109, 111
sensibility matrix, 98
shallow-mixing type, 26, 32
signal to noise ratio, 64, 82
significance level, 33, 59, 66
source/sink terms, 3, 415, 417, 418, 420, 421, 422, 423
South Atlantic Current, 376
South Atlantic Ocean, 208, 370, 373, 374, 375
South China Sea, 11, 12, 13, 14, 15, 16, 54, 55, 56, 57, 59, 60, 82, 83, 84, 165, 166, 167, 169, 170, 171, 172, 173, 174, 175, 178, 181, 247, 249, 253, 255, 272, 274, 275, 276, 281, 282, 284, 285, 287, 289, 291, 298, 301, 302, 305, 315, 316, 317, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334
South China Sea Deep Water, 84
South China Sea Subsurface Water, 83
South China Sea Surface Water, 83
South Equatorial Current, 123, 124, 255, 373, 374
South Pacific Tropical Water, 256, 257
Southern Ocean, 379, 380, 385
spatial decorrelation scale, 71, 72, 73, 74, 75, 283, 319, 334
spectral coefficient, 88, 89, 100
speed parameter, 116, 127, 128, 129, 130
standard deviation, 14, 33, 36, 60, 62, 85, 140, 223, 226, 318, 319, 349, 350
static instability, 78, 79, 131
Stokes Theorem, 208

stratification-formed mechanism, 44, 49
strong 'sources/sinks', 418, 424
Subpolar Front, 141, 144, 149, 153, 157, 160, 162, 419, 420, 421, 422
Subtropical Countercurrent, 276–281, 363
Sulu Sea, 47, 48, 49, 51, 54, 166, 255, 272
summer monsoon, 19, 55, 56, 141, 142, 149, 152, 153, 157, 160, 162, 166, 167, 169, 175, 180, 193, 282, 292, 294, 296, 298, 316, 329, 330, 348
surface cooling, 25, 26, 44, 48, 49, 397, 401
surface salt flux, 25
surface wind stress, 48, 49, 132, 202, 204, 223, 226, 230, 231, 238, 277, 302, 310, 329, 425, 426, 427
Sverdrup transport, 277, 301

Tasman Front, 359
Tasman Sea, 362
t-distribution, 33, 59, 64, 256
temporal decorrelation scale, 63, 72, 74, 82, 283, 319, 334, 350
terrain-following coordinate, 66
thermal expansion coefficient, 36, 43
thermal structure, 13, 17, 19, 22, 52, 53, 62, 73, 348
thermal wind relation, 3, 6, 7, 110, 120, 122, 127, 129, 189, 202, 229
thermocline, 153, 318, 319, 321, 322, 334, 336, 337, 349, 352, 368, 419, 423, 547, 549, 551
thermodynamic equation, 3, 415
thermohaline circulation, 236, 261, 283, 293, 396
thermohaline front, 296, 299, 301, 302, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 375, 393
thermohaline variability, 63, 282, 305, 307, 310
top shallow sub model, 34
total kinetic energy, 132, 421
trajectory, 119, 120, 122
T-*S* diagram, 83, 84, 306, 308, 309, 323, 324, 333, 334, 355
Tsushima cold-core eddy, 352, 359
Tsushima Strait, 142, 143, 426, 427
Tsushima Warm Current, 143, 160, 162, 185, 189, 191, 193, 426
t-test, 32, 33
two scalar functions, 236, 238
two-step determination, 116

Ulleung/Tsushima Basin, 143, 144, 157, 193
Ulleung eddy, 351, 355
unbalanced data assimiliation, 407, 408, 412, 413
unconstrained optimization, 186

variational P-vector method, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198
velocity potential, 236, 237, 408
vertical eddy diffusivity, 416
Vietnamese Bight, 168, 169, 170, 171, 172, 320, 326, 331
volume transport, 162, 163, 165, 166, 181, 183, 185, 187, 188, 201, 203, 204, 206, 207, 209, 211, 216, 221, 224, 226, 241, 243, 246, 247, 249, 250, 251, 253, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 280, 361, 365, 366, 367, 372, 373, 374, 380, 381, 382, 383, 384, 385, 386, 418, 426, 427
volume transport streamfunction, 274, 275, 276, 366
volume transport vorticity, 203, 204, 205, 206, 207, 241
vorticity equation, 3, 114, 203, 204, 409

warm-core eddy, 167, 282, 316, 329, 330, 352, 359
Weddell Gyre, 383, 385
Weddell Sea, 383, 384, 385, 387
Weddell-Scotia Confluence, 385
West Spitzbergen cell, 236
West Spitzbergen Current, 232, 233, 293, 394, 397, 399, 401, 402, 403, 404
western boundary current, 133, 137, 139, 140, 166, 176, 177, 178, 253, 276, 277, 292, 301, 359, 369, 371, 372, 373, 383

white noise, 90, 91, 92, 96, 97, 217, 221
wind forcing, 26, 237, 425, 427
wind-driven circulation, 236, 259, 301
winter monsoon, 17, 141, 149, 153, 157, 160, 162, 166, 169, 175, 193, 292, 294, 295, 296, 348
WOA dataset, 391
WOCE, 256

Yamato eddy, 351
Yellow Sea, 16, 17, 18, 19, 22, 63, 65, 66, 68, 70, 73, 74, 75, 80, 351, 532

z-level analysis, 15, 77, 78, 79, 104, 283, 296