Centro Euro-Mediterraneo sui Cambiamenti Climatici

Research Papers
Issue RP0166
November 2012

Scientific Computing and Operations Division (SCO)

OceanVAR software for use with NEMO: documentation and test guide

By **Luisa D'Amore**
Universitá degli Studi di Napoli Federico II
luisa.damore@unina.it

Rossella Arcucci
Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC)
rossella.arcucci@cmcc.it

Luisa Carracciuolo
Istituto di Chimica e Tecnologia di Polimeri (CNR)
luisa.carracciuolo@ictp.cnr.it

and **Almerico Murli**
Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC)
almerico.murli@cmcc.it

SUMMARY In this document is described the OceanVAR software (a three-dimensional variational data assimilation system, 3dvar) developed at the CMCC for use with the NEMO [6] modeling system and SLA, ARGO, XBT and GLIDER observations [1].
OceanVar software was developed using Fortran90 and it uses the NetCDF library to read the input files provided from NEMO software.
This document has the aim to provide a guide for the installation and the use of the OceanVar software. So, here is provided a brief description of the 3dvar model developed, the main steps of the algorithm implemented are shown, and above all how you can use the software is explained: how to install it, what you need to know to install it, where are located the routine, where to find the sources codes, how to edit the Makefile, how to write the input files etc. Regarding the input files we devoted some Sections to show the structure of these file. We provide the schemes you need follow to build the various input files.
A brief tour around the OceanVAR code is provided. Significant efforts have been made to make the code documenting, so this note should be seen as a prelude to looking at the code itself.
The system described here is the release 2006 of the software.

CMCC Research Papers

02

Centro Euro-Mediterraneo sui Cambiamenti Climatici

Contents

Introduction 4

The OceanVAR Algorithm 5

OceanVAR Source Code Organization 7

program oceanvar 9

The Data Files 11

namelist 12

diagnostic 14

firstguess.nc 15

grid1.nc 17

eofs.nc 18

User Guide (compiling and running) 20

Example 24

Bibliography 34

INTRODUCTION

OceanVAR software is used with the NEMO modeling system and the SLA, ARGO, XBT and GLIDER observations.
The data provided from NEMO software are stored in a vector named $x_M = [S, T, \eta, u, v]^T$ (defined on a domain Ω). This vector represent the state variables: salinity S in a tridimensional space, temperature T in a tridimensional space, the free surface elevation η (bidimensional data), and the vectors u and v total horizontal velocity components.

The OceanVar scheme iteratively finds the minimum of the following function:

$$J(\boldsymbol{v}) = \frac{1}{2}\boldsymbol{v}^T\boldsymbol{v} + \frac{1}{2}(\boldsymbol{HVv} - \boldsymbol{d})^T R^{-1}(\boldsymbol{HVv} - \boldsymbol{d}) \tag{1}$$

defined on the three-dimensional space, where R is the observational error covariance matrix, $\boldsymbol{V}$ is obtained assuming the background error covariance matrix B written as $B = VV^T$ and $\boldsymbol{v} = \boldsymbol{V}^+\delta\boldsymbol{x}$ (see [1]), where the superscript "+" indicates the generalized inverse.
The vector

$$\boldsymbol{d} = [\boldsymbol{y} - H\boldsymbol{x_M}] \tag{2}$$

is the **misfit**, y is the vector of observed data defined on a domain Ω' and $H : \Omega \to \Omega'$ is the linearized observational operator [2].

As described in [1], the matrix $\boldsymbol{V}$ is modelled at each minimization iteration as a sequence of linear operators: $\boldsymbol{V} = \boldsymbol{V_D V_{uv} V_\eta V_H V_V}$. The linear operator $\boldsymbol{V_V}$ transforms coefficients which multiply vertical EOFs into vertical profiles of temperature and salinity defined at the model vertical levels, $\boldsymbol{V_H}$ applies horizontal covariances on fields of temperature and salinity, $\boldsymbol{V_\eta}$ calculates the sea surface height error covariance from three dimensional fields of temperature and salinity, $\boldsymbol{V_{uv}}$ calculates velocity from sea surface height, temperature and salinity, and $\boldsymbol{V_D}$ applies a divergence damping filter on the velocity field.
The coefficients of the EOFs are computed by using the singular value decomposition of the covariance matrix obtained for the state vector as described in [7].

THE OCEANVAR ALGORITHM

OceanVAR algorithm is mainly based on two steps: **calc_msft** and **3dvar** (see Figure 1) having the same names of the main modules of the software described in the next section.
These steps implement, respectively, the operations to compute the misfit $d = y - Hx_M$ and the minimun of the function $J(\boldsymbol{v}) = \frac{1}{2}\boldsymbol{v}^T\boldsymbol{v} + \frac{1}{2}(\boldsymbol{HVv} - \boldsymbol{d})^T R^{-1}(\boldsymbol{HVv} - \boldsymbol{d})$.

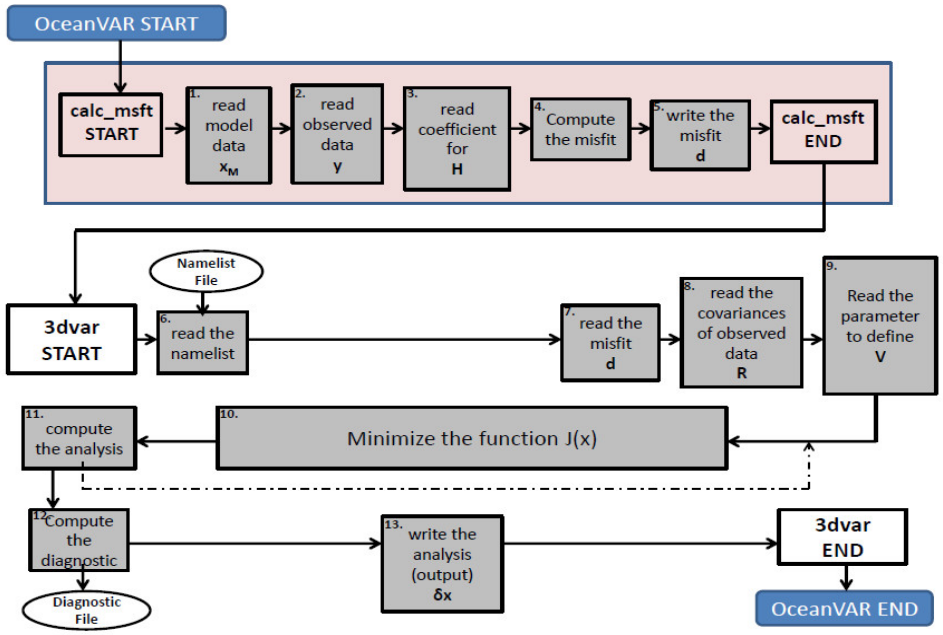

Figure 1:
Illustration of the major steps taken during the OceanVAR analysis precedure

We summarizes the role of each step in the OceanVAR algorithm (for details regarding to the file of data see section ”DATA FILES”):

1. **setup the background**: The background field x_M is provide in NetCDF file, so the software use the NetCDF library to read these values. Variables required by calc_msft are stored in the xb Fortran90 derived data type (e.g. xb%u, xb%v etc) which are output of NEMO software [6].

2. **setup the observations**: the software reads the observations y from the input file `name_obs.dat`. Variables required by calc_msft are stored in the name_obs Fortran90 derived data type (e.g. if name_obs=ARGO, the array arg%lon, arg%lat, etc. store the longitude values, latitude values, etc.).

3. **setup the operator** H: the software reads the parameters to define the operators H from the input file. The value depends on the type of observed data which is used.

4. **compute the misfit**: the software computes: $d = Hx_M - y$.
5. **write the misfit**: the values of the misfit are written in the output file `name_obs_mis.dat`.
6. **read the namelist**: the software run-time options for the 3dvar module are read in from a `namelist` file. These options are described more fully in section "NAMELIST FILE".
7. **read the misfit**: the values of the misfit used by 3dvar module are read in the input file `name_obs_mis.dat` (for details see "DATA FILES" section).
8. **read the covariance matrix** R: covariance matrix of observations are read in the input file `name_obs_mis.dat`.
9. **setup the operator** V: the software reads the parameters necessary for the operators $\boldsymbol{V_D V_{uv} V_\eta V_H V_V}$ from the input files: `grid.nc`, `eof.nc` and `namelist` (more details in "DATA FILES" section).
10. **minimize the cost function J**: to compute the minumum of $J(v)$ the software uses an L-BFGS routine [3].
11. **compute the analysis** δx: To get the value δx named **analysis**, the software convert v: $\delta x = Vv$.
12. **compute the diagnosis**: the software writes in a diagnostic file (`Oceanvar.diagnostic`) all used parameters, the grid dimension, the number of used observation, etc. This file is updated during the execution.
13. **write the output**: the software writes the analysis values in an output file `corr_#griglia.dat`.

OCEANVAR SOURCE CODE ORGANIZATION

This section provides a brief tour around the OceanVAR code. Significant efforts have been made to make the code documenting, so this section should be seen as a prelude to looking at the code itself.

Figure 2 show the content of the directory `oceanvar` which stores all the files needed for the installation and use of the software.

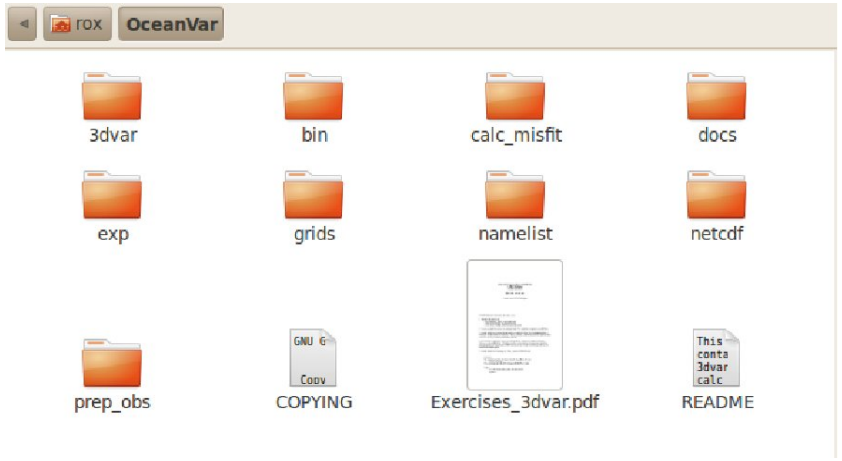

Figure 2:
`oceanvar` source code organization

The software consists of two main modules: `calc_msft` and `3dvar`. The both modules must be compiled.
The source files for the `calc_msft` module are stored in `nemo_MFS` sub.directory (see Figure 3).

The source file for `3dvar` module are stored in `src` sub-directory and they are organized in sub-directories named with the ”operations” which are implemented inside (see Figure 4).

For all routines is used the Fortran90 excepted the `''routine.f''` routine in which is implemented the L-BFGS [3] methods in Fortran77.

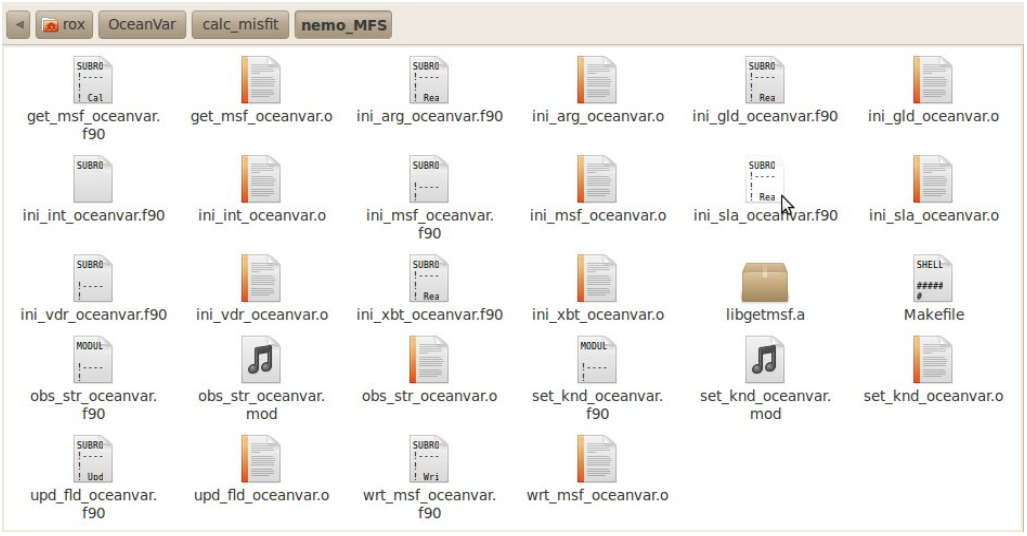

Figure 3:
`calc_msft` source code organization

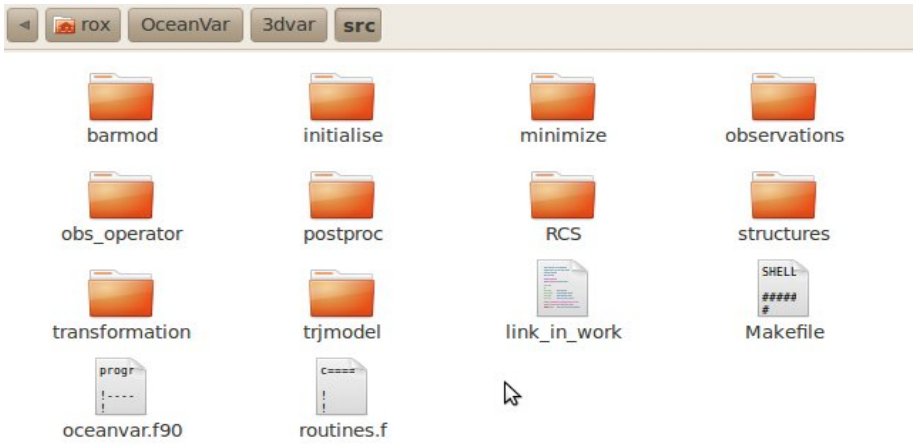

Figure 4:
`src` subdirectory source code organization

PROGRAM OCEANVAR

In this section we describe the routine `oceanvar.f90` which is the main program for `3dvar` module:

```
program oceanvar

! ---

! ---

! ---

!-----------------------------------------------------------------

!!! READ FILES STORED IN namelist DIRECTORY

! Initialize diagnostics and read namelists
      call def_nml

! ---
! Outer loop - multigrid
  do ktr = 1,drv%ntr

   drv%ktr = ktr

!!! READ AND INITIALISE GRID - THE FILES USED ARE STORED IN initialise DIRECTORY

! ---
! Define grid parameters
      if( ktr.eq.1 .or. drv%ratio(ktr).ne.1.0 )then
       call def_grd
       write(drv%dia,*) 'out of def_grd'
      endif

!!! READ OBSERVATIONS AND OPERATOR H - THE FILES USED ARE STORED IN observations DIRECTORY

! ---
! Get observations
      if(ktr.eq.1) call get_obs
       write(drv%dia,*) 'out of get_obs'

! ---
! Define interpolation parameters
      call int_par
       write(drv%dia,*) 'out of def_grd'

! ---
! Define observational vector
      call obs_vec
       write(drv%dia,*) 'out of obs_vec'

!!! INITIALISE OPERATORS V - THE FILES USED ARE STORED IN initalise DIRECTORY

! ---
! Define constants for background covariances
     if( ktr.eq.1 .or. drv%ratio(ktr).ne.1.0 ) then
      call def_cov
       write(drv%dia,*) 'out of def_cov'
     endif

! ---
! Initialise barotropic model
      if(drv%bmd(drv%ktr) .eq. 1) then
         call ini_bmd
          write(drv%dia,*) 'out of ini_bmd'
      endif

!!! INITIALISE FUNCTION J - THE FILES USED ARE STORED IN initalise DIRECTORY

! ---
! Initialise cost function and its gradient
      call ini_cfn
```

```
       write(drv%dia,*) 'out of ini_cfn'

! ---
! Initialise from old iterration
     if( ktr.gt.1 .and. drv%ratio(ktr).ne.1.0 ) then
      call ini_itr
       write(drv%dia,*) 'out of ini_itr'
     endif

!!! MINIMIZE FUNCTION J - THE FILES USED ARE STORED IN minimize DIRECTORY

! ---
! Minimize the cost function (inner loop)
     call min_cfn
       write(drv%dia,*) 'out of min_cfn'

   if(ktr.eq.drv%ntr)then

!!! POST PROCESSING - THE FILES USED ARE STORED IN postproc DIRECTORY

! ---
! Convert to innovations
     call cnv_inn
! ---
! Write outputs and diagnostics
     call wrt_dia
   endif

! ---
! Save old iterration
   if( ktr.ne.drv%ntr)then
    if(drv%ratio(ktr+1).ne.1.0 ) then
     call sav_itr
    endif
   endif

! ---
! End of outer loop
  enddo

!-----------------------------------------------------------------------

end program oceanvar
```

THE DATA FILES

The OceanVAR software use three type of input file: ASCII file, binary and NerCDF file [4] (Network Common Data Form).

ASCII files:

1. `namelist`: input file. In this file are stored all the parameters necessary for configuring the software; for example, the types of data observed used, parameters to configure the minimum BFGS routine, etc. It is used in the `3dvar` module;

2. `diagnostic`: output file. In this file are stored all the parameters used and produced during the execution of the software, for example the size of the grid used, the number of observations used, the number of iterations needed for the calculation of the minimum, the value of the function and its gradient calculated at each iteration; It is used in the `3dvar` module;

binary files:

1. `name_obs.dat`: input file. In this file are storeds the vectors of observed data (vectors y) name_obs and their covariance matrices. For example, if the observed data are of type ARGO, the file containing the values of those data and the respective covariance matrices is ARGO.dat. It is used in the `calc_misfit` module;

2. `name_obs_mis.dat`: input/output file. In this file are read/written respectively the values of the misfit computeded on the observed data type name_obs. For example, if the observed data are of type ARGO, the file containing the values of these data si ARGO_mis.dat. This file is used in the `calc_misfit` module as an output file and in the `3dvar` module as an input file;

3. `corr_#griglia.dat`: output fie. In this file are written to the results of the data assimilation process (the vector δx). It is used in the `3dvar` module;

NetCDF fies:

1. `firstguess.nc`: input file. In this file are stored the vectors x_M. It is used in the `calc_misfit` module;

2. `grid1.nc`: input file. This file stores the variables required to define the grid (examples are the latitude and longitude). It is used in the `3dvar` module;

3. `eof.nc`: input file. In this file are stored the values of the EOFs. It is used in the `3dvar` module;

NAMELIST

In the next block there's the structure of OceanVAR namelist file.

```
!-------------------------------------------------------------
! OceanVar namelist
!-------------------------------------------------------------
!
! Namelist runlst
! ---
!
!             Define the general set-up
!
!  flag_a    - character*12: flag for the analysis
!  sdat_f    - Starting date of the forecast
!  shou_f    - Starting hour of the forecast
!  lhou_f    - Length of the forecast
!
! ---
&runlst
   flag_a    =
   sdat_f    =
   shou_f    =
   lhou_f    =
/
!-------------------------------------------------------------
!
! Namelist obslst
!
!             Observational data sets set-up
!
! ---
!  obs_sla -  1-assimilate sla, 0-do not assimilate
!  obs_arg -  1-assimilate Argo, 0-do not assimilate
!  obs_xbt -  1-assimilate XBT, 0-do not assimilate
!  obs_gld -  1-assimilate glider, 0-do not assimilate
!  obs_tra -  1-assimilate Argo trajectories, 0-do not assimilate
!  obs_trd -  1-assimilate drifter trajectories, 0-do not assimilate
!  obs_vdr -  1-assimilate velocity from drifters,
!             0-do not assimilate
!
! ---
&obslst
   obs_sla =
   obs_arg =
   obs_xbt =
   obs_gld =
   obs_tra =
   obs_trd =
   obs_vdr =
/
!-------------------------------------------------------------
!
! Namelist drvlst
! ---
!
!             Outer loop set-up
!
!  ntr      -  Number of outer iterations over grids
!  grid     -  grid number in each iteration
!  read_grd -  Logical to read grids from files.
!              See subroutine def_grd.f90
!  ratio    -  Resolution ratio between the previous
!              and the current grid
!  mask     -  Mask type:
!               1 - no land
!               2 - 2d mask by coasts
!               3 - 3d mask by bottom topography
!  barmd    -  Run barotropic model on the grid
!  divda    -  Apply divergence damping on the grid
!  divdi    -  Initialise corrections by divergence damping
!
```


```
! ---
&grdlst
   ntr      =
   grid     =
   read_grd =
   ratio    =
   mask     =
   barmd    =
   divda    =
   divdi    =
/
!-----------------------------------------------------------
!
! Namelist ctllst
! ---
!
!             BFGS minimizers set-up
!
!  ctl_m   -  Number of copies saved in the minimizer
!  ctl_tol -  Stopping criteria (absolute)
!  ctl_per -  Stopping criteria (relative)
!
! ---
&ctllst
   ctl_m    =
   ctl_tol  =
   ctl_per  =
/
!-----------------------------------------------------------
!
! Namelist rcflst
! ---
!
!             Covariance constants
!
!  neof     - Number of vertical EOFs
!  nreg     -  Number of regions
!  read_eof - Logical to read EOFs from files.
!             See subroutine def_cov.f90
!  rcf_ntr -  Number of iterations of the recursive filter
!  rcf_L   -  Horizontal correlation radius
!  rcf_efc -  Extension factor for coasts
!
! ---
&covlst
   neof     =
   nreg     =
   read_eof =
   rcf_ntr  =
   rcf_L    =
   rcf_efc  =
/
!-----------------------------------------------------------
!
! Namelist slalst
! ---
!
!          SLA assimilation set-up
!
!  sla_dep -  Maximum depth for assimilation of sla
!
! ---
&slalst
   sla_dep  =
/
!-----------------------------------------------------------
!
! Namelist bmdlst
! ---
!
!          Barotropic model set-up
!
!  bmd_dt  -  Time step
!  bmd_ndy -  Number of days to integrate the model
```

```
!  bmd_ady -  Number of days for averaging
!  bmd_ady -  Weight for the trapezoidal scheme
!  bmd_fc1 -  Horizontal friction for vorticity
!  bmd_fc2 -  Horizontal friction for divergence
!  bmd_ovr -  Overrelaxation
!  bmd_resem - Stopping criteria
!  bmd_ncnt - Maximum number of successive corrections
!
! ---
&bmdlst
  bmd_dt     =
  bmd_ndy    =
  bmd_ady    =
  bmd_alp    =
  bmd_fc1    =
  bmd_fc2    =
  bmd_ovr    =
  bmd_resem  =
  bmd_ncnt   =
/
!-------------------------------------------------------------
```

DIAGNOSTIC

In the next block there's the structure of OceanVAR diagnostic file.

```
 -------------------------------------------------------------

                          NAMELISTS:

 -------------------------------------------------------------
 RUN NAMELIST INPUT:
 Flag for the analysis:           flag_a    =
 Starting day of the forecast:    sdat_f    =
 Starting hour of the forecast:   shou_f    =
 Lenght of the forecast:          lhou_f    =
 -------------------------------------------------------------
 OBSERVATIONS NAMELIST INPUT:
 Use SLA observations:                    obs_sla =
 Use ARGO observations:                   obs_arg =
 Use XBT observations:                    obs_xbt =
 Use glider observations:                 obs_gld =
 Use velocity observations:               obs_vel =
 Use Argo trajectory observations:        obs_tra =
 Use drifter trajectory observations:     obs_trd =
 Use drifter observations of velocity:    obs_vdr =
 -------------------------------------------------------------
 GRID NAMELIST INPUT:
 Multigrid iterrations:                    ntr     =
 Grids:                                   grid     =
 Read grids from a file:                 read_grd  =
 Ratio:                                  ratio     =
 Masks:                                   mask     =
 Run barotropic model:                   barmd     =
 Divergence damping in analysis:         divda     =
 Divergence damping in initialisation:   divdi     =
 -------------------------------------------------------------
 MINIMIZER NAMELIST INPUT:
 Number of saved vectors:         ctl_m     =
 Minimum gradient of J:           ctl_tol   =
 Percentage of initial gradient:  ctl_per   =
 -------------------------------------------------------------
 COVARIANCE NAMELIST INPUT:
 Number of EOFs:                  neof      =
 Number of regions:               nreg      =
```


```
 Read EOFs from a file:            read_eof =
 Half number of iterations:        rcf_ntr  =
 Horizontal correlation radius:    rcf_L    =
 Extension factor for coastlines: rcf_efc   =
-------------------------------------------------------------
 SLA NAMELIST INPUT:
 Minimum depth for observations:  sla_dep   =
-------------------------------------------------------------
 BAROTROPIC MODEL NAMELIST INPUT:
 Time step:              bmd_dt     =
 Simulation days:        bmd_ndy    =
 Averaged days:          bmd_ady    =
 Implicit weight:        bmd_alp    =
 Friction intensity:     bmd_fc1    =
 Friction intensity:     bmd_fc2    =
 Over-relaxation:        bmd_ovr    =
 Minimum residual:       bmd_resem  =
 Maximum iterations      bmd_ncnt   =
-------------------------------------------------------------
Grid dimensions are:
out of def_grd
Number of SLA observations:
Number of ARGO observations:
No xbt:
out of get_obs
Number of SLA observations:
Real number of ARGO observations:
out of def_grd
 Total number of observations:
out of obs_vec
Eof dimensions are:
Uses    ----    eofs.
out of def_cov
out of ini_bmd
Size of the control vector:
out of ini_cfn
out of min_cfn
out of get_obs
Number of SLA observations:
Real number of ARGO observations:
out of def_grd
 Total number of observations:
out of obs_vec
out of ini_bmd
out of ini_cfn
out of min_cfn
writes to corrections.dat !!!!!!!!!!!!!!!!!!!!!!!!!
```

FIRSTGUESS.NC

In the next block there's the structure of the `firstguess.nc` file input for calc_misfit.

```
netcdf first_guess {
dimensions:
        x =   ;
        y =   ;
        deptht =   ;
        time_counter =   ; //
variables:
        float nav_lon(y, x) ;
                nav_lon:units = " " ;
                nav_lon:valid_min =  ;
                nav_lon:valid_max =   ;
                nav_lon:long_name = " " ;
                nav_lon:nav_model = " " ;
```

```
        float nav_lat(y, x) ;
                nav_lat:units = " " ;
                nav_lat:valid_min =  ;
                nav_lat:valid_max =  ;
                nav_lat:long_name = " " ;
                nav_lat:nav_model = " " ;
        float deptht(deptht) ;
                deptht:units = " " ;
                deptht:positive = " " ;
                deptht:valid_min =  ;
                deptht:valid_max =  ;
                deptht:title = " " ;
                deptht:long_name = " " ;
        float time_counter(time_counter) ;
                time_counter:units = " " ;
                time_counter:calendar = " " ;
                time_counter:title = " " ;
                time_counter:long_name = " " ;
                time_counter:time_origin = " " ;
        float vosaline(time_counter, deptht, y, x) ;
                vosaline:units = " " ;
                vosaline:missing_value =  ;
                vosaline:valid_min =  ;
                vosaline:valid_max =  ;
                vosaline:long_name = " " ;
                vosaline:short_name = " " ;
                vosaline:online_operation = "  " ;
                vosaline:axis = " " ;
                vosaline:interval_operation =  ;
                vosaline:interval_write =  ;
                vosaline:associate = " " ;
        float votemper(time_counter, deptht, y, x) ;
                votemper:units = " " ;
                votemper:missing_value =  ;
                votemper:valid_min =  ;
                votemper:valid_max =  ;
                votemper:long_name = " " ;
                votemper:short_name = " " ;
                votemper:online_operation = " " ;
                votemper:axis = " " ;
                votemper:interval_operation =  ;
                votemper:interval_write =  ;
                votemper:associate = " " ;
        float sossheig(time_counter, y, x) ;
                sossheig:units = " " ;
                sossheig:missing_value =  ;
                sossheig:valid_min =  ;
                sossheig:valid_max =  ;
                sossheig:long_name = " " ;
                sossheig:short_name = " " ;
                sossheig:online_operation = " " ;
                sossheig:axis = " " ;
                sossheig:interval_operation =  ;
                sossheig:interval_write =  ;
                sossheig:associate = " " ;
        float sowaflup(time_counter, y, x) ;
                sowaflup:units = " " ;
                sowaflup:missing_value =  ;
                sowaflup:valid_min =  ;
                sowaflup:valid_max =  ;
                sowaflup:long_name = " " ;
                sowaflup:short_name = " " ;
                sowaflup:online_operation = " " ;
                sowaflup:axis = "  ;
                sowaflup:interval_operation =  ;
                sowaflup:interval_write =  ;
                sowaflup:associate = " " ;
        float sohefldo(time_counter, y, x) ;
                sohefldo:units = " " ;
                sohefldo:missing_value =  ;
                sohefldo:valid_min =  ;
                sohefldo:valid_max =  ;
                sohefldo:long_name = " " ;
                sohefldo:short_name = " " ;
```


```
                sohefldo:online_operation = " " ;
                sohefldo:axis = " " ;
                sohefldo:interval_operation =   ;
                sohefldo:interval_write =   ;
                sohefldo:associate = " " ;
        float soshfldo(time_counter, y, x) ;
                soshfldo:units = " " ;
                soshfldo:missing_value =   ;
                soshfldo:valid_min =   ;
                soshfldo:valid_max =   ;
                soshfldo:long_name = " " ;
                soshfldo:short_name = " " ;
                soshfldo:online_operation = " " ;
                soshfldo:axis = " " ;
                soshfldo:interval_operation =   ;
                soshfldo:associate = " " ;

// global attributes:
                :Conventions = " " ;
                :file_name = " " ;
                :production = " " ;
                :TimeStamp = " " ;
                :associate_file = " " ;


data:

nav_lon =
nav_lon_{1,1}  nav_lon_{2,1} ... ... ... ... ... ... nav_lon_{jm,1},
nav_lon_{1,2}  nav_lon_{2,2} ... ... ... ... ... ... nav_lon_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
nav_lon_{1,im} nav_lon_{2,im} ... ... ... ... ... ... nav_lon_{jm,im};


nav_lat =
nav_lat_{1,1}  nav_lat_{2,1} ... ... ... ... ... ... nav_lat_{jm,1},
nav_lat_{1,2}  nav_lat_{2,2} ... ... ... ... ... ... nav_lat_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
nav_lat_{1,im} nav_lat_{2,im} ... ... ... ... ... ... nav_lat_{jm,im};



... Other... ;
```

GRID1.NC

In the next block there's the structure of the `grid1.nc` file which is an input file for `3dvar` module. It contains the parameters to define the grid.

```
netcdf grid1 {
dimensions:
        im =  ;
        jm = ;
        km =  ;
variables:
        float lon(jm, im) ;
                lon:units = " " ;
        float lat(jm, im) ;
                lat:units = " " ;
        float dep(km) ;
                dep:units = " " ;
        float lsm(jm, im) ;
                lsm:long_name = " " ;
        float dx(jm, im) ;
                dx:long_name = " " ;
                dx:units = " " ;
        float dy(jm, im) ;
```

```
                dy:long_name = " " ;
                dy:units = " " ;
        float dz(km) ;
                dz:long_name = " " ;
                dz:units = " " ;
        float topo(jm, im) ;
                topo:long_name = " " ;
                topo:units = " " ;
        float regs(jm, im) ;
                regs:long_name = " " ;
                regs:units = " " ;
        float mdt(jm, im) ;
                mdt:long_name = " " ;
                mdt:units = " " ;
        float tmsk(km, jm, im) ;
                tmsk:long_name = " " ;
                tmsk:units = " " ;

// global attributes:
                :title = " " ;



data:

lon =
lon_{1,1}  lon_{2,1} ... ... ... ... ... ... lon_{jm,1},
lon_{1,2}  lon_{2,2} ... ... ... ... ... ... lon_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
lon_{1,im} lon_{2,im} ... ... ... ... ... ... lon_{jm,im};


lat =
lat_{1,1}  lat_{2,1} ... ... ... ... ... ... lat_{jm,1},
lat_{1,2}  lat_{2,2} ... ... ... ... ... ... lat_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
lat_{1,im} lat_{2,im} ... ... ... ... ... ... lat_{jm,im};



... Other... ;
```

EOFS.NC

In the next block there's the structure of `eofs.nc` file which is an input file for `3dvar` module. It contains the parameters to define the EOFs.

```
netcdf eofs {
dimensions:
        nreg =   ;
        nlev =   ;
        neof =   ;
variables:
        float eva(neof, nreg) ;
                eva:long_name = " " ;
        float evc(neof, nlev, nreg) ;
                evc:long_name = " " ;

// global attributes:
                :title = " " ;



data:

eva =
eva_{1,1}  eva_{2,1} ... ... ... ... ... ... eva_{jm,1},
eva_{1,2}  eva_{2,2} ... ... ... ... ... ... eva_{jm,2},
```

```
... ... ... ... ... ... ... ... ... ... ... ... ... ... ;
eva_{1,im} eva_{2,im} ... ... ... ... ... ... eva_{jm,im};



... Other... ;
```

USER GUIDE (COMPILING AND RUNNING)

OceanVar software consist of two main module: `calc_msft` and `3dvar`.

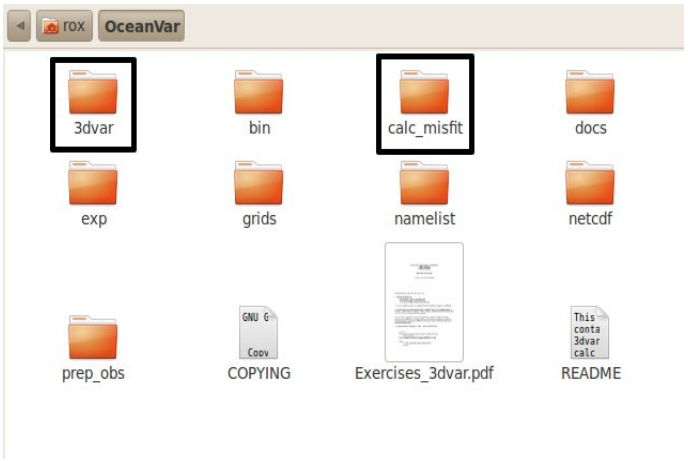

Figure 5:
OceanVAR directory

To install the software you must compile both modules.

calc_msft installation:

The `calc_msft` source file are stored in `nemo_MFS` subdirectory.
To install `calc_msft` module you need to modify the Makefiles and to type ”make” in `nemo_MFS` directory.

```
SHELL = /bin/sh

#############################################################
#
#         Makefile for OceanVar
#
#############################################################

F_C = sxf90 -dw -c
F_O =

LIB = libgetmsf.a

RM      = rm -f
AR      = sxar ruv

OBJ =  \
       set_knd_oceanvar.o \
       obs_str_oceanvar.o \
       ini_arg_oceanvar.o \
       ini_msf_oceanvar.o \
       ini_gld_oceanvar.o \
```


```
        ini_sla_oceanvar.o \
        ini_vdr_oceanvar.o \
        upd_fld_oceanvar.o \
        ini_int_oceanvar.o \
        ini_xbt_oceanvar.o \
        get_msf_oceanvar.o \
        wrt_msf_oceanvar.o


.SUFFIXES: .f90


all:  $(LIB)
        @echo $(LIB) is compiled

$(LIB) :          $(STR) $(OBJ)
        $(AR) $(LIB) $(OBJ) $(STR)

.DEFAULTS:
.f90.o :  $(STR) $(OBJ)
        $(F_C) $(F_O) $*.f90


clean:
        /bin/rm -f *.o *.mod i.* *.L $(LIB)
```

It is necessary in `Makefile` setup:

- `F_C` = fortran compilator

For example, if you choses to use `g95` compiler, you assigns:

- `F_C = g95`

3dvar installation:

The `3dvar` source file are stored in subdirectory named 3dvar as shown in Figure 6.

Figure 6:
directory 3DVar

To install `3dvar` go into `work` directory, modify the `Makefile` adding information about your compiler etc. and after to type ”make”.

```
SHELL = /bin/sh

#############################################################
#
#         Makefile for OceanVar
#
#############################################################

F_C = sxf90 -dw -c
F_O =

LIB = libgetmsf.a

RM      = rm -f
AR      = sxar ruv

OBJ =  \
       set_knd_oceanvar.o \
       obs_str_oceanvar.o \
       ini_arg_oceanvar.o \
       ini_msf_oceanvar.o \
       ini_gld_oceanvar.o \
       ini_sla_oceanvar.o \
       ini_vdr_oceanvar.o \
       upd_fld_oceanvar.o \
       ini_int_oceanvar.o \
       ini_xbt_oceanvar.o \
       get_msf_oceanvar.o \
       wrt_msf_oceanvar.o


.SUFFIXES: .f90


all:  $(LIB)
        @echo $(LIB) is compiled

$(LIB) :          $(STR) $(OBJ)
        $(AR) $(LIB) $(OBJ) $(STR)
```


```
.DEFAULTS:
.f90.o :  $(STR) $(OBJ)
        $(F_C) $(F_O) $*.f90


clean:
        /bin/rm -f *.o *.mod i.* *.L $(LIB)
```

It is necessary in the `Makefile` to setup:

- `F_C`= fortran compilator
- `F_O` ed `F_L` =compiler parameters
- `EXTINC`=path for including NetCDF library
- `EXTLIB`=path for using and compiling NetCDF library.

For example, if you choses to use `g95` compiler, you assigns:

- `F_C = g95`
- `F_O = $(F_C) -dw -c -ftrace=frame -cpp -Dopt_netcdf`
- `F_L = $(F_C) -dw -ftrace=frame -cpp -Dopt_netcdf`

also, the variable `path_netcdf` contain the path to the location of the directory in which you installed the NetCDF library, the variable `EXTINC` and `EXTINC` are:

- `EXTINC = -I/path_netcdf/include`
- `EXTLIB = -L/path_netcdf/lib -lnetcdf`

The executable generated typing ”make” is named `var_3d`. It is located in the directory `bin` of `oceanvar`.

Remark 0.1 *The NetCDF library must be compiled with the same fortran compiler with which you choose to fill OceanVAR.*

EXAMPLE

The files stored in the directory named `OceanVar/exp` (see Figure 7) are an example to run the 3dvar executable file.

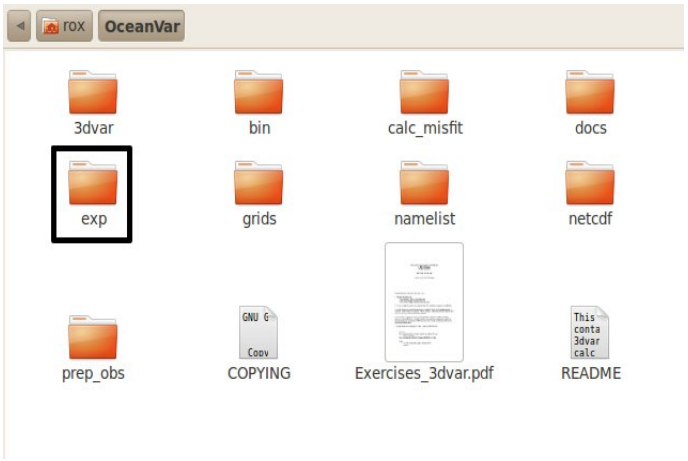

Figure 7:
directory OceanVAR

Into `OceanVar/exp/MFS471/work` subdirectory there is a script named `exercise1.sh` to use the OceanVar software for a MFS configuration of NEMO software [6] using the observed data `SLA, ARGO, XBT` and `GLIDER` [1].

The script:

To use the example provided in this section you need to assign to the variable `work` the path of the position in which the OceanVar softare is installed adding the path of the position in which the exercise is stored.

```
#################

work=path_oceanvar/exp/MFS471/work
cd $work

# 3D-VAR assimilation

# Exercise 1

#################
```


```
!  Link files with input variables

rm SLA.dat ARGO.dat XBT.dat GLIDER.dat drifter_vel.dat
```

! load the input files related to the observed data

```
ln -fs exercise1_input/SLA.dat SLA.dat
ln -fs exercise1_input/ARGO.dat ARGO.dat
ln -fs exercise1_input/XBT.dat XBT.dat
ln -fs exercise1_input/GLIDER.dat GLIDER.dat
ln -fs exercise1_input/drifter_vel.dat drifter_vel.dat
```

! load the netcdf input files relate to the x_M vector.

```
rm first_guess.nc
ln -fs all_input/050225_T.nc first_guess.nc
```

! copy the calc_msft executable in the local directory.

```
rm calc_msft
ln -fs ../bin/calc_msft calc_msft
```

! compute the misfit

```
./calc_msft first_guess.nc
```

! load the netcdf input file containing the information of the grid.

```
rm grid1.nc
ln -fs grids/MFS_16_72.nc grid1.nc
```

! load the netcdf files related to the EOFs.

```
rm eofs.nc
ln -fs all_input/WINTER_Med_eof.nc eofs.nc
```

! load namelist file

```
rm var_3d_nml
ln -fs oceanvar_nml var_3d_nml
```

! copy var_3d executable into local directory.

```
rm var_3d
ln -fs ../../../bin/var_3d var_3d
```

! minimize the function J.

```
./var_3d
```

The `name_obs.dat` file:

The values of the observed data are provided in binary files located in the same directory in which there are data of the grid. These files have the name of the observed data to which they refer: `sla.dat`, `argo.dat`, `xbt.dat`, etc.

The `firstguess.nc` file:

We provide an example of `firstguess.nc` file (an input file for the calc_misfit module). It contain the x_M vector defined on a grid 182×64.

```
netcdf first_guess {
dimensions:
        x = 182 ;
        y = 64 ;
        deptht = 72 ;
        time_counter = UNLIMITED ; // (1 currently)
variables:
        float nav_lon(y, x) ;
                nav_lon:units = "degrees_east" ;
                nav_lon:valid_min = -9.125f ;
                nav_lon:valid_max = 36.125f ;
                nav_lon:long_name = "Longitude" ;
                nav_lon:nav_model = "Default grid" ;
        float nav_lat(y, x) ;
                nav_lat:units = "degrees_north" ;
                nav_lat:valid_min = 30.25f ;
                nav_lat:valid_max = 46.f ;
                nav_lat:long_name = "Latitude" ;
                nav_lat:nav_model = "Default grid" ;
        float deptht(deptht) ;
                deptht:units = "m" ;
                deptht:positive = "unknown" ;
                deptht:valid_min = 1.472102f ;
                deptht:valid_max = 5334.648f ;
                deptht:title = "deptht" ;
                deptht:long_name = "Vertical T levels" ;
        float time_counter(time_counter) ;
```


```
        time_counter:units = "seconds since 0005-02-25 00:00:00" ;
        time_counter:calendar = "gregorian" ;
        time_counter:title = "Time" ;
        time_counter:long_name = "Time axis" ;
        time_counter:time_origin = " 0005-FEB-25 00:00:00" ;
float vosaline(time_counter, deptht, y, x) ;
        vosaline:units = "PSU" ;
        vosaline:missing_value = 1.e+20f ;
        vosaline:valid_min = 1.e+20f ;
        vosaline:valid_max = -1.e+20f ;
        vosaline:long_name = "Salinity" ;
        vosaline:short_name = "vosaline" ;
        vosaline:online_operation = "ave(only(x))" ;
        vosaline:axis = "TZYX" ;
        vosaline:interval_operation = 1200.f ;
        vosaline:interval_write = 86400.f ;
        vosaline:associate = "time_counter deptht nav_lat nav_lon" ;
float votemper(time_counter, deptht, y, x) ;
        votemper:units = "C" ;
        votemper:missing_value = 1.e+20f ;
        votemper:valid_min = 1.e+20f ;
        votemper:valid_max = -1.e+20f ;
        votemper:long_name = "Pot. Temper" ;
        votemper:short_name = "votemper" ;
        votemper:online_operation = "ave(only(x))" ;
        votemper:axis = "TZYX" ;
        votemper:interval_operation = 1200.f ;
        votemper:interval_write = 86400.f ;
        votemper:associate = "time_counter deptht nav_lat nav_lon" ;
float sossheig(time_counter, y, x) ;
        sossheig:units = "M" ;
        sossheig:missing_value = 1.e+20f ;
        sossheig:valid_min = 1.e+20f ;
        sossheig:valid_max = -1.e+20f ;
        sossheig:long_name = "Sea Surface Height" ;
        sossheig:short_name = "sossheig" ;
        sossheig:online_operation = "ave(only(x))" ;
        sossheig:axis = "TYX" ;
        sossheig:interval_operation = 1200.f ;
        sossheig:interval_write = 86400.f ;
        sossheig:associate = "time_counter nav_lat nav_lon" ;
float sowaflup(time_counter, y, x) ;
        sowaflup:units = "Kg/m2/S" ;
        sowaflup:missing_value = 1.e+20f ;
        sowaflup:valid_min = 1.e+20f ;
        sowaflup:valid_max = -1.e+20f ;
        sowaflup:long_name = "Net Upward Water Flux" ;
        sowaflup:short_name = "sowaflup" ;
        sowaflup:online_operation = "ave(only(x))" ;
        sowaflup:axis = "TYX" ;
        sowaflup:interval_operation = 1200.f ;
        sowaflup:interval_write = 86400.f ;
        sowaflup:associate = "time_counter nav_lat nav_lon" ;
float sohefldo(time_counter, y, x) ;
        sohefldo:units = "W/m2" ;
        sohefldo:missing_value = 1.e+20f ;
        sohefldo:valid_min = 1.e+20f ;
        sohefldo:valid_max = -1.e+20f ;
        sohefldo:long_name = "Net Downward Heat Flux" ;
        sohefldo:short_name = "sohefldo" ;
        sohefldo:online_operation = "ave(only(x))" ;
        sohefldo:axis = "TYX" ;
        sohefldo:interval_operation = 1200.f ;
        sohefldo:interval_write = 86400.f ;
        sohefldo:associate = "time_counter nav_lat nav_lon" ;
float soshfldo(time_counter, y, x) ;
        soshfldo:units = "W/m2" ;
        soshfldo:missing_value = 1.e+20f ;
        soshfldo:valid_min = 1.e+20f ;
        soshfldo:valid_max = -1.e+20f ;
        soshfldo:long_name = "Shortwave Radiation" ;
        soshfldo:short_name = "soshfldo" ;
        soshfldo:online_operation = "ave(only(x))" ;
        soshfldo:axis = "TYX" ;
```

```
                soshfldo:interval_operation = 1200.f ;
                soshfldo:interval_write = 86400.f ;
                soshfldo:associate = "time_counter nav_lat nav_lon" ;

// global attributes:
                :Conventions = "GDT 1.3" ;
                :file_name = "PE_1d_050225_050226_grid_T.nc" ;
                :production = "An IPSL model" ;
                :TimeStamp = "2008-JUN-16 01:41:08 GMT-0200" ;
                :associate_file = "PE_1d_050225_050226_grid_T.nc PE_1d_050225_050226_grid_U.nc
                     PE_1d_050225_050226_grid_V.nc" ;



data:

nav_lon =
nav_lon_{1,1}  nav_lon_{2,1} ... ... ... ... ... ... nav_lon_{jm,1},
nav_lon_{1,2}  nav_lon_{2,2} ... ... ... ... ... ... nav_lon_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
nav_lon_{1,im} nav_lon_{2,im} ... ... ... ... ... ... nav_lon_{jm,im};


nav_lat =
nav_lat_{1,1}  nav_lat_{2,1} ... ... ... ... ... ... nav_lat_{jm,1},
nav_lat_{1,2}  nav_lat_{2,2} ... ... ... ... ... ... nav_lat_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
nav_lat_{1,im} nav_lat_{2,im} ... ... ... ... ... ... nav_lat_{jm,im};



... Other... ;
```

`grid1.nc` file:

In the example `exercise1.sh`, the grid used is $871 \times 253 \times 72$.

```
netcdf grid1 {
dimensions:
        im = 871 ;
        jm = 253 ;
        km = 72 ;
variables:
        float lon(jm, im) ;
                lon:units = "degrees_east" ;
        float lat(jm, im) ;
                lat:units = "degrees_north" ;
        float dep(km) ;
                dep:units = "meters" ;
        float lsm(jm, im) ;
                lsm:long_name = "Land-sea mask" ;
        float dx(jm, im) ;
                dx:long_name = "Resolution in X" ;
                dx:units = "meters" ;
        float dy(jm, im) ;
                dy:long_name = "Resolution in Y" ;
                dy:units = "meters" ;
        float dz(km) ;
                dz:long_name = "Resolution in Z" ;
                dz:units = "meters" ;
        float topo(jm, im) ;
                topo:long_name = "Topography" ;
                topo:units = "meters" ;
        float regs(jm, im) ;
                regs:long_name = "Regions" ;
                regs:units = " " ;
        float mdt(jm, im) ;
                mdt:long_name = "Mean dynamic topography" ;
```


```
                mdt:units = "meters" ;
        float tmsk(km, jm, im) ;
                tmsk:long_name = "T points mask" ;
                tmsk:units = " " ;

// global attributes:
                :title = "Mediterranean grid of OPA model: MFS1672ti" ;



data:

lon =
lon_{1,1}  lon_{2,1} ... ... ... ... ... ... lon_{jm,1},
lon_{1,2}  lon_{2,2} ... ... ... ... ... ... lon_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
lon_{1,im} lon_{2,im} ... ... ... ... ... ... lon_{jm,im};


lat =
lat_{1,1}  lat_{2,1} ... ... ... ... ... ... lat_{jm,1},
lat_{1,2}  lat_{2,2} ... ... ... ... ... ... lat_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
lat_{1,im} lat_{2,im} ... ... ... ... ... ... lat_{jm,im};



... Other... ;
```

The `eofs.nc` file:

We provide an example of `eofs.nc` file (an input file for the `3dvar` module). It contain the EOFs.

```
netcdf eofs {
dimensions:
        nreg = 13 ;
        nlev = 145 ;
        neof = 20 ;
variables:
        float eva(neof, nreg) ;
                eva:long_name = "Eigen values" ;
        float evc(neof, nlev, nreg) ;
                evc:long_name = "EOFs" ;

// global attributes:
                :title = "WINTER EOFs for the Mediterranean" ;



data:

eva =
eva_{1,1}  eva_{2,1} ... ... ... ... ... ... eva_{jm,1},
eva_{1,2}  eva_{2,2} ... ... ... ... ... ... eva_{jm,2},
... ... ... ... ... ... ... ... ... ... ... ... ... ... ,
eva_{1,im} eva_{2,im} ... ... ... ... ... ... eva_{jm,im};




... Other... ;
```

The `var_3d_nml` namelist file:

We provide an example of `namelist` file into the `OceanVar/namelist` directory. You need to assign the input parameter for the OceanVar software using this file.

```
!-------------------------------------------------------------
! OceanVar namelist
!-------------------------------------------------------------
!
! Namelist runlst
! ---
!
!             Define the general set-up
!
!  flag_a    - character*12: flag for the analysis
!  sdat_f    - Starting date of the forecast
!  shou_f    - Starting hour of the forecast
!  lhou_f    - Length of the forecast
!
! ---
&runlst
   flag_a    = "MFS_analysis"
   sdat_f    = 20050225
   shou_f    = 12
   lhou_f    = 24
/
!-------------------------------------------------------------
!
! Namelist obslst
!
!             Observational data sets set-up
!
! ---
!  obs_sla -  1-assimilate sla, 0-do not assimilate
!  obs_arg -  1-assimilate Argo, 0-do not assimilate
!  obs_xbt -  1-assimilate XBT, 0-do not assimilate
!  obs_gld -  1-assimilate glider, 0-do not assimilate
!  obs_tra -  1-assimilate Argo trajectories, 0-do not assimilate
!  obs_trd -  1-assimilate drifter trajectories, 0-do not assimilate
!  obs_vdr -  1-assimilate velocity from drifters,
!             0-do not assimilate
!
! ---
&obslst
   obs_sla = 1
   obs_arg = 1
   obs_xbt = 1
   obs_gld = 0
   obs_tra = 0
   obs_trd = 0
   obs_vdr = 0
/
!-------------------------------------------------------------
!
! Namelist drvlst
! ---
!
!             Outer loop set-up
!
!  ntr      -  Number of outer iterations over grids
!  grid     -  grid number in each iteration
!  read_grd - Logical to read grids from files.
!             See subroutine def_grd.f90
!  ratio    -  Resolution ratio between the previous
!             and the current grid
!  mask     -  Mask type:
!              1 - no land
!              2 - 2d mask by coasts
!              3 - 3d mask by bottom topography
!  barmd    -  Run barotropic model on the grid
!  divda    -  Apply divergence damping on the grid
!  divdi    -  Initialise corrections by divergence damping
!
! ---
```


```
&grdlst
   ntr      =  2,
   grid     =  1,  1,   1
   read_grd = .T.  ,.T.,.T.
   ratio    =  1., 1.,  1.
   mask     =  3,  3,   3
   barmd    =  1,  1,   1
   divda    =  0,  1,   1
   divdi    =  0,  1,   1
/
!-----------------------------------------------------------
!
! Namelist ctllst
! ---
!
!             BFGS minimizers set-up
!
!  ctl_m   -  Number of copies saved in the minimizer
!  ctl_tol -  Stopping criteria (absolute)
!  ctl_per -  Stopping criteria (relative)
!
! ---
&ctllst
   ctl_m    = 5
   ctl_tol  = 1.e-2
   ctl_per  = 1.e-2
/
!-----------------------------------------------------------
!
! Namelist rcflst
! ---
!
!             Covariance constants
!
!  neof     - Number of vertical EOFs
!  nreg     -  Number of regions
!  read_eof - Logical to read EOFs from files.
!             See subroutine def_cov.f90
!  rcf_ntr -  Number of iterations of the recursive filter
!  rcf_L   -  Horizontal correlation radius
!  rcf_efc -  Extension factor for coasts
!
! ---
&covlst
   neof     = 20
   nreg     = 13
   read_eof = .true.
   rcf_ntr  = 4
   rcf_L    = 15000.
   rcf_efc  = 2.0
/
!-----------------------------------------------------------
!
! Namelist slalst
! ---
!
!          SLA assimilation set-up
!
!  sla_dep -  Maximum depth for assimilation of sla
!
! ---
&slalst
   sla_dep  = 150.
/
!-----------------------------------------------------------
!
! Namelist bmdlst
! ---
!
!          Barotropic model set-up
!
!  bmd_dt  -  Time step
!  bmd_ndy -  Number of days to integrate the model
!  bmd_ady -  Number of days for averaging
```

```
!  bmd_ady -  Weight for the trapezoidal scheme
!  bmd_fc1 -  Horizontal friction for vorticity
!  bmd_fc2 -  Horizontal friction for divergence
!  bmd_ovr -  Overrelaxation
!  bmd_resem - Stopping criteria
!  bmd_ncnt - Maximum number of successive corrections
!
! ---
&bmdlst
  bmd_dt    = 7200.
  bmd_ndy   =  3.
  bmd_ady   = 0.8
  bmd_alp   = 1.0
  bmd_fc1   = 0.1
  bmd_fc2   = 0.2
  bmd_ovr   = 1.9
  bmd_resem = 5.e-2
  bmd_ncnt  = 201
/
!-------------------------------------------------------------
```

The `corr_1.dat` file:

Using the OceanVar software with the input files described in this section you obtain the valued of δx_i saved in the `corr_1.dat` file which is stored in the current directory.

The `Oceanvar.diagnostics` file:

We provide an example of `diagnostic` file obtained using the OceanVar software with the input files described in this section.

```
 -------------------------------------------------------------

                        NAMELISTS:

 -------------------------------------------------------------
 RUN NAMELIST INPUT:
 Flag for the analysis:           flag_a    = MFS_analysis
 Starting day of the forecast:    sdat_f    =   20050225
 Starting hour of the forecast:   shou_f    =   12
 Lenght of the forecast:          lhou_f    =   24
 -------------------------------------------------------------
 OBSERVATIONS NAMELIST INPUT:
 Use SLA observations:                    obs_sla =   1
 Use ARGO observations:                   obs_arg =   1
 Use XBT observations:                    obs_xbt =   1
 Use glider observations:                 obs_gld =   0
 Use velocity observations:               obs_vel =   0
 Use Argo trajectory observations:        obs_tra =   0
 Use drifter trajectory observations:     obs_trd =   0
 Use drifter observations of velocity:    obs_vdr =   0
 -------------------------------------------------------------
 GRID NAMELIST INPUT:
 Multigrid iterrations:                    ntr     =   2
 Grids:                                   grid     =   1  1
 Read grids from a file:                  read_grd =   T  T  T
 Ratio:                                   ratio    =   1.000000000000000   1.000000000000000
 Masks:                                    mask    =   3  3
 Run barotropic model:                    barmd    =   1  1
 Divergence damping in analysis:          divda    =   0  1
 Divergence damping in initialisation:    divdi    =   0  1
 -------------------------------------------------------------
```


```
 MINIMIZER NAMELIST INPUT:
 Number of saved vectors:          ctl_m     =   5
 Minimum gradient of J:            ctl_tol   =   1.0000000000000000D-02
 Percentage of initial gradient:   ctl_per   =   1.0000000000000000D-02
-------------------------------------------------------------
 COVARIANCE NAMELIST INPUT:
 Number of EOFs:                   neof      =   20
 Number of regions:                nreg      =   13
 Read EOFs from a file:            read_eof  =   T
 Half number of iterations:        rcf_ntr   =   4
 Horizontal correlation radius:    rcf_L     =   15000.00000000000
 Extension factor for coastlines:  rcf_efc   =   2.000000000000000
-------------------------------------------------------------
 SLA NAMELIST INPUT:
 Minimum depth for observations:   sla_dep   =   150.0000000000000
-------------------------------------------------------------
 BAROTROPIC MODEL NAMELIST INPUT:
 Time step:              bmd_dt      =   7200.000000000000
 Simulation days:        bmd_ndy     =   3.000000000000000
 Averaged days:          bmd_ady     =   0.8000000000000000
 Implicit weight:        bmd_alp     =   1.000000000000000
 Friction intensity:     bmd_fc1     =   0.1000000000000000
 Friction intensity:     bmd_fc2     =   0.2000000000000000
 Over-relaxation:        bmd_ovr     =   1.900000000000000
 Minimum residual:       bmd_resem   =   5.0000000000000000D-02
 Maximum iterations      bmd_ncnt    =   201
-------------------------------------------------------------
Grid dimensions are:    871   253   72
out of def_grd
Number of SLA observations:    0
Number of ARGO observations:    568
No xbt:    18869
out of get_obs
Number of SLA observations:  >>>>>>>>>>>>>  0
Real number of ARGO observations:    568
out of def_grd
 Total number of observations:    1784
out of obs_vec
Eof dimensions are:    13   145   20
Uses    20    eofs.
out of def_cov
out of ini_bmd
Size of the control vector:    4407260
out of ini_cfn
out of min_cfn
out of get_obs
Number of SLA observations:  >>>>>>>>>>>>>  0
Real number of ARGO observations:    568
out of def_grd
 Total number of observations:    1784
out of obs_vec
out of ini_bmd
out of ini_cfn
out of min_cfn
writes to corrections.dat !!!!!!!!!!!!!!!!!!!!!!!!!
```

Bibliography

[1] An oceanographic three-dimensional variational data assimilation scheme - S.Dobricic, N.Pinardi - Ocean Modelling 22 (2008) 89-105 - Elsevier

[2] HPC computation issues of the incremental 3D variational data assimilation scheme in OceanVar software - L.D'Amore, R.Arcucci, L.Marcellino, A.Murli - Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM) vol. 7, no. 3-4, 2012, pp. 91-105 ISSN 1790-8140.

[3] J. Nocedal R.H. Byrd, P. Lu and C. Zhu, L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization, ACM Transactions on Mathematical Software, Vol. 23, No. 4, December 1997, Pages 550-560.

[4] http://www.unidata.ucar.edu/software/netcdf/

[5] E.Lorenz Empirical Orthogonal Functions and Statistical Weather Prediction, december 1956, scientific report No.1, Statistical Forecasting Project.

[6] http://www.nemo-ocean.eu/

[7] Quality Assessment of a 1985-2007 Mediterranean Sea Reanalysis - M.Adani, S.Dobricic, N.Pinardi - 2011, Journal of Atmospheric and Oceanic Technology vol.28

[8] Impact of Multi-altimeter Sea Level Assimilation in the Mediterranean Forecasting Model - M.-I. Pujol, S. Dobricic, N. Pinardi, M. Adani - Journal of Atmospheric & Oceanic Technology; Dec2010, Vol. 27 Issue 12, p2065

[9] A Parallel Three-dimensional Variational Data Assimilation Scheme - L.D'Amore, R.Arcucci, L.Marcellino, A.Murli - Numerical Analysis and Applied Mathematics, AIP C.P. 1389, 1829-1831 - ISBN: 978-0-7354-0956-9

Visit **www.cmcc.it** for information on our activities and publications.

The Euro-Mediterranean Centre on Climate Change is a Ltd Company with its registered office and administration in Lecce and local units in Bologna, Venice, Capua, Sassari, Viterbo, Benevento and Milan. The society doesn't pursue profitable ends and aims to realize and manage the Centre, its promotion, and research coordination and different scientific and applied activities in the field of climate change study.