

CDFTOOLS: a fortran 90 package of programs and libraries for diagnostic of the DRAKKAR OPA9 output.

J.M. Molines *

Last update: (cdftools-2.0.9) September 13, 2006

Introduction

This document describes a set of programs written in Fortran 90, used as diagnostic tools for NEMO-OPA9 model output. This work has been initialized in the frame of the DRAKKAR project, where large model configuration are run. For this reason, a special care has been taken to minimize the required amount of memory. Also, most of the programs assume that the data base is DRAKKAR-like, which means with one file per snap-shot. Shell scripts are also indicated as demo for how to use the programs. The user must carefully check all the variables in the scripts in order to ensure the compatibility with its own settings.

The programs are sorted by category: Statistical (means, variance, RMS, EKE, etc ...), transport (mass, heat, salt), derived quantities (densities, Brunt Vaisala frequency, potential vorticity) and extracting/information tools (vertical profiles, position on the horizontal grid etc ...).

This package is open, and in order to help for new developments, all the netcdf IO (at the IOIPSL standard) are collected in a unique module (cdfio.f90) which is used by the programs. In the same way the routines or functions concerning the equation of state of sea water are also collected into the module eos.f90. The functions and subroutines included in these modules are described in the appendix.

Recently added tools

cdffmax : Display min/max for a variable in a file with location.

cdffmxlheatc : Compute the Heat Content of the mixed layer (J/m2).

cdffsigtrp : Compute density class transport across a section.

cdffzoom : Show an ASCII representation of a 2D (x-y, x-z or y-z) slab from any variable of an output file.

cdffcensus : Compute the water mass census (volume of (T,S) bins).

cdffmocsig : Compute the MOC as a function of potential density (σ_1)

cdffcediags : Compute sea ice area, extent and volume from an icemod output

cdffzonalsum : Compute the zonal sum of the variables in a file, may uses sub basins. (useful for tracer inventory, for instance).

*Laboratoire des Ecoulements Géophysiques et Industriels, CNRS UMR 5519, Grenoble, France

1 Statistics

cdfmoy:

Purpose: Compute the mean fields for the file list given in argument.

Usage: *cdfmoy nc_files*

Input: A list of homogeneous model output files (*e.g.*: xxx_gridT.nc or xxx_gridU.nc).

For instance: *cdfmoy ORCA025-G32_y0010m10d??_gridT.nc* will compute the mean 'gridT' file for month 10 of year 10.

Required mesh_mask files or other files: *att.txt*. This file is used every time a new netcdf variables is build from a program. It contains informations on the various attributes for a given variable name. The standard version of this file is given in the CDFTOOLS main directory. (See appendix for details).

Output: 2 files are produced : *cdfmoy.nc* and *cdfmoy2.nc*. *cdfmoy.nc* holds the mean fields for all the variables in the input files. *cdfmoy2.nc* holds the quadratic mean of some input variables (not all). In the current version, the quadratic mean for *sossheig*, *vozocrtx*, *vomecrtx*, *vovecrtz* are saved.

Remark: Assumes that land value have are set to 0.

Associated script: *cdfmoy.ll*: This script is used in the DRAKKAR project to calculate monthly means, quarterly means and annual means. This is a good example of how to use *cdfmoy*.
cdfmoy-inter.ll: This is a variant of the first script to compute inter-annual means.

cdfmoy_sp:

Purpose: Compute the mean fields for the file list given in argument, just as in `cdfmoy`. The only difference is that land value (or missing values) are not necessarily 0. Useful when 0 have a physical meaning.

Usage: `cdfmoy_sp nc_files`

Input: A list of homogeneous model output files (*e.g.*: `xxx_gridT.nc` or `xxx_gridU.nc`).

For instance: `cdfmoy_sp ORCA025-G32_y0010m10d??_gridT.nc` will compute the mean 'gridT' file for month 10 of year 10.

Required mesh_mask files or other files: `att.txt`. This file is used every-time a new netcdf variables is build from a program. It contains informations on the various attributes for a given variable name. The standard version of this file is given in the CDFTOOLS main directory. (See appendix for details).

Output: 2 files are produced : `cdfmoy.nc` and `cdfmoy2.nc`. `cdfmoy.nc` holds the mean fields for all the variables in the input files. `cdfmoy2.nc` holds the quadratic mean of some input variables (not all). In the current version, the quadratic mean for `sossheig`, `vozocrtx`, `vomecrtx`, `vovecrtz` are saved.

Remark:

Associated scripts: `cdfmoy_sp.ll`: This is an example where it is necessary to use `cdfmoy_sp` instead of `cdfmoy`.

cdfmean:

Purpose: Compute the mean value of a field, weighted by the local metric. If the variable is a 3D variable, the mean value is given for each level, then the global mean is printed. The mean value can be computed on a limited domain, specified on the command line.

Usage: *cdfmean nc_files nc_var T|U|V|F|W [imin imax jmin jmax kmin kmax]*

Input: nc_file is the name of the netcdf file which hold the variable.

nc_var is the netcdf variable name for the mean computation. If a wrong or dummy variable is given, the program presents the list of available variables in the file, and ask for a choice.

T|U|V|F|W : specify the point on the C-grid, corresponding to the variable nc_var.

imin imax jmin jmax kmin kmax : optional parameters. If used, all 6 must be specified. They indicate the limited area (in i,j,k coordinates) where the mean value will be computed. The user can specify 0 as input, which means that the corresponding coordinate will be considered for the whole extent; in this case the pair of coordinates must be set to 0.

For instance: *cdfmean ORCA025-G42_y0010_ANNUAL_gridT.nc votemper T* will compute the mean temperature over the whole domain.

cdfmean ORCA025-G42_y0010_ANNUAL_gridT.nc xxx T will ask a variable name from the list of variables contained in the file. Careful, the type of point (T U V or F) is not asked interactively; this is not really a problem as in most of the OPA9 output, files are build for each type (gridT, gridU etc...).

cdfmean ORCA025-G42_y0010_ANNUAL_gridU.nc vozocrtx U 300 320 400 653 0 0 will compute the mean U-component of the velocity on a horizontally limited area, for the whole water column.

Other valid specifications for the limited area can be, for example : *0 0 400 600 1 15* : the mean will be computed for the upper 15 levels, for a whole zonal band starting at j=400 and ending at j=600.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc, mask.nc must be in the current directory.

Output: Results are printed on the standard output. For 3D fields, intermediate mean values for each level are also displayed.

Remark/bugs : In this version, no special care has been taken to handle neither the periodic grids, nor the north folding conditions. This will be done in a future release.

Associated scripts: None. This program is more typically an interactive program. As it is written, it can handle grids as big as ORCA025 on a small Linux machine (with only 512 Mb of core memory).

cdfheatc:

Purpose: Compute the heat content for the ocean in a given 3D domain (or the whole domain). The heat content (Joules) is computed and given for each levels, then the global heat content (J) is printed, as well as the heat content per unit of volume (J/m³). A sub-domain can be specified on the command line.

Usage: *cdfheatc gridTfiles [imin imax jmin jmax kmin kmax]*

Input: gridTfile is the name of the netcdf file which holds *votemper*.

imin imax jmin jmax kmin kmax : optional parameters. If used, all 6 must be specified. They indicate the limited area (in i,j,k coordinates) where the mean value will be computed. The user can specify 0 as input, which means that the corresponding coordinate will be considered for the whole extent; in this case the pair of coordinates must be set to 0.

For instance: *cdfheatc ORCA025-G42-y0010-ANNUAL-gridT.nc* will compute the heat content over the whole domain.

cdfheatc ORCA025-G42-y0010-ANNUAL-gridT.nc 300 320 400 653 0 0 will compute the heat content on a horizontally limited area, for the whole water column.

Other valid specifications for the limited area can be, for example : *0 0 400 600 1 15* : the heat content will be computed for the upper 15 levels, for a whole zonal band starting at j=400 and ending at j=600.

Required mesh_mask files or other files: *mesh_hgr.nc*, *mesh_zgr.nc*, *mask.nc* must be in the current directory.

Output: Results are printed on the standard output. For 3D fields, intermediate values for each level are also displayed.

Remark/bugs : In this version, no special care has been taken to handle neither the periodic grids, nor the north folding conditions. This will be done in a future release.

Associated scripts: None. This program is more typically an interactive program. As it is written, it can handle grids as big as ORCA025 on a small Linux machine (with only 512 Mb of core memory).

cdfmxlheatc:

Purpose: Compute the heat content for the ocean in the mixed layer read in the gridT file.

Usage: *cdfmxlheatc gridTfile*

Input: gridTfile is the name of the netcdf file which holds *votemper* and *somxl010*

For instance: `cdfmxlheatc ORCA025-G42_y0010_m03d15_gridT.nc` will compute the heat content in the mixed layer for this file.

Required mesh_mask files or other files: *att.txt*, *mesh_zgr.nc*, *mask.nc* must be in the current directory.

Output: Results are written in the netcdf file *mxlheatc.nc*, in the variable *somxlheatc*. Units are Joules/m²

Associated scripts: None.

cdfzonalmean:

Purpose: Compute the zonal mean value for all the variables in the file given as argument.

Usage: *cdfzonalmean nc_files T|U|V|F|W [sub_basin_mask]*

Input: nc.file is the name of the netcdf file which hold the variables.

T|U|V|F|W : specify the point on the C-grid, corresponding to the variables in the file.

sub_basin_mask: If given the program read this file to set a *sub_basin_mask* (for global configurations). If this argument is not given, then the zonal mean is assumed to be global, which is OK for a basin configuration, such as NATL4, for instance. For instance: **cdfzonalmean ORCA025-G42_y0010_ANNUAL_gridT.nc T** will compute the zonal mean temperature over the whole domain; the resulting variable is a 2D variable (latitude,depth).

cdfzonalmean ORCA025-G42_y0010_ANNUAL_gridT.nc T new_maskglo will compute the zonal mean of all the variables contained in the file. A zonal mean for each sub basin will be output.

Required mesh_mask files or other files: *att.txt*, *mesh_hgr.nc*, *mesh_zgr.nc*, *mask.nc* must be in the current directory.

Output: The program outputs as many variables as there are in the input file times the number of sub-basin (5) if the sub-basin mask is given. Variables name starts with 'zo' which replaces the 'vo' or 'so' of the input variable. The name of the sub basin is then appended to the variable name: For instance, zonal mean for *votemper* gives (in case of sub-basins) : *zotemper_glo, zotemper_atl, zotemper_inp, zotemper_ind, zotemper_pac*. ALL these new variables must be defined in *att.txt* ...

Associated scripts: None

cdfzonalsum:

Purpose: Compute the zonal sum value for all the variables in the file given as argument.

Usage: *cdfzonalsum nc_files T|U|V|F|W [sub_basin_mask]*

Input: nc.file is the name of the netcdf file which hold the variables.

T|U|V|F|W : specify the point on the C-grid, corresponding to the variables in the file.

sub_basin_mask: If given the program read this file to set a *sub_basin_mask* (for global configurations). If this argument is not given, then the zonal sum is assumed to be global, which is OK for a basin configuration, such as NATL4, for instance. For instance: **cdfzonalsum ORCA025-G50_y1958_ANNUAL_ptrcT.nc T** will compute the zonal sum of the variables over the whole domain; the resulting variable is a 2D variable (latitude,depth), or just a vector (latitude) if the input variable is already a 2D horizontal variable (inventory, for instance).

Required mesh_mask files or other files: *att.txt*, *mesh_hgr.nc*, *mesh_zgr.nc*, *mask.nc* must be in the current directory.

Output: The program outputs as many variables as there are in the input file times the number of sub-basin (5) if the sub-basin mask is given. Variables name starts with 'zo' which replaces the 'vo' or 'so' of the input variable. The name of the sub basin is then appended to the variable name: For instance, zonal sum for *votemper* gives (in case of sub-basins) : *zotemper_glo, zotemper_atl, zotemper_inp, zotemper_ind, zotemper_pac*. ALL these new variables must be defined in *att.txt* (which is definitively very tedious...)

Associated scripts: see *cdftrc.ll*

See also: *cdfzonalout* : this a formatting tool to print ASCII results in a good shape for the 1D results produced as ncdf by *cdfzonalsum*.

cdfzonalout:

Purpose: Produce a nice ASCII output for 1D variables resulting from cdfzonalsum, or cdfzonalmean

Usage: *cdfzonalout zonalsum.nc*

Input: zonalsum.nc is a netcdf file output from cdfzonalsum or cdfzonalmean

Output: Output is done on standard output (can be re-directed to a file via >)

Associated scripts: see cdftrc.ll

Number of 1D variables : 6

zoinvcfc_glo
zoinvc14_glo
zoqtrcfc_glo
zoqtrc14_glo
zoqintcfc_glo
zoqintc14_glo

npiglo= 1

npjglo= 1021

npk = 46

J	LAT	zoinvcfc_glo	zoinvc14_glo
1021	89.8876	0.114485E-06	1639.13867
1020	89.9478	0.114504E-06	1660.38854
1019	89.8876	0.114485E-06	1639.13867
1018	89.7937	0.112609E-06	1521.98022
1017	89.6954	0.111228E-06	1462.95922
1016	89.5956	0.110859E-06	1355.69262
1015	89.4949	0.109885E-06	1315.38806
1014	89.3935	0.109691E-06	1265.77246
1013	89.2915	0.109644E-06	1211.48840
1012	89.1890	0.108149E-06	1163.96777
1011	89.0860	0.105885E-06	1132.33557
1010	88.9825	0.103872E-06	1096.84130
.....			

cdfvT:

Purpose: Compute the mean UT, US, VT, VS for transport computation.

Usage: *cdfvT CONFIG 'list_of_tags'*

Input: CONFIG is the valid config name (e.g. ORCA025-G32, NATL4-B01, ORCA05-K18 ...).

In general model output files are build as \$CONFIG_\$tag_grid?.nc. The tag part of the name is usually something like y0008m09d10 for instance, but virtually, it is the part of the name between \$CONFIG_ and _grid.

list_of_tags is just the succession of the tags that are to be used in the mean.

When using cdfvT, we assume that all the data files (i.e. gridT, gridU and gridV files for the given CONFIG and tags) are in the current directory.

For instance: cdfvT ORCA025-G32 y0010m10d01 y0010m10d06 y0010m10d11 will compute the mean UT etc fields for the 3 given tags.

Required mesh_mask files or other files: att.txt

Output: *vt.nc*. This file contains the four 3-D variables vozout, vozous, vomevt, vomevs.

Remark: For the sake of simplicity, only one file is used as output, but one should remember that the U-transport (UT and US) are computed at the C-grid U-point, and that the V-transport (VT and VS) are computed at the C-grid V-point. In fact, temperature and salinity are interpolated on the corresponding velocity point, in order to respect mass conservation.

Associated script: *cdfvT.ll*: This script is used in the DRAKKAR project to compute the monthly, quarterly and annual means for the UT US VT VS terms.

cdfvT-inter: This is a variant of the first script for inter-annual means.

cdfeke:

Purpose: Compute the Eddy Kinetic Energy (EKE).

Usage: *cdfeke gridU gridU2 gridV gridV2 gridT*

Input: gridU and gridU2 hold respectively the mean and quadratic mean for U-points. The same for gridV and gridV2. These files are produced by cdfmoy. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means). The EKE is computed with respect to this period of time.

An extra gridT type file is given in argument, just for reading the T-grid, for the header of the output file. In fact, EKE is computed on the T-points.

For instance: *cdfeke ORCA035-G32_y0008-0010_gridU.nc ORCA035-G32_y0008-0010_gridU2.nc ORCA035-G32_y0008-0010_gridV.nc ORCA035-G32_y0008-0010_gridV2.nc ORCA035-G32_y0008-0010_gridT2.nc*

will compute the EKE for the period y0008-0010.

Required mesh_mask files or other files: att.txt

Output: *eke.nc*. This file hold the variable voeke.

Remark: EKE is computed at T-points.

Associated script: *cdfeke.ll*: This script can be used to compute EKE. It is a good example on how to use cdfefe. Note that this script must be used after cdfmoy.ll, because it requires the mean and mean quadratic files to be already computed.

cdfrmsssh:

Purpose: Compute the RMS of the SSH.

Usage: *cdfrmsssh gridT gridT2*

Input: gridT and gridT2 hold respectively the mean and quadratic mean for T-points. These files are produced by cdfmoy. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means). The RMS is computed with respect to this period of time.
For instance: *cdfrmsssh ORCA035-G32_y0008-0010_gridT.nc ORCA035-G32_y0008-0010_gridT2.nc* will compute the RMS SSH for the period y0008-0010.

Required mesh_mask files or other files: att.txt

Output: *rmsssh.nc*. This file hold the variable sossheig_rms.

Remark:

Associated script: *cdfrms.ll*: This scripts is use to compute both RMS ssh and the Stdev of W.

cdfstdevw:

Purpose: Compute the standard deviation for W.

Usage: *cdfstdevw gridW gridW2*

Input: gridW and gridW2 hold respectively the mean and quadratic mean for W-points. These files are produced by cdfmoy. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means). The standard deviation is computed with respect to this period of time.

For instance: *cdfstdevw ORCA035-G32_y0008-0010_gridW.nc ORCA035-G32_y0008-0010_gridW2.nc* will compute the standard deviation of W for the period y0008-0010.

Required mesh_mask files or other files: att.txt

Output: *rmsw.nc*. This file hold the variable vovecrtz_rms.

Remark:

Associated script: *cdfrms.ll*: This scripts is use to compute both RMS ssh and the Stdev of W.

2 Transports

cdfmhst:

Purpose: Compute the Meridional Heat and Salt Transport (Partial Step case).

Usage: *cdfmhst VTfile*

Input: VTfiles are the files produced by the cdfvT program. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means). For instance: cdfmhst ORCA025-G32_y0010m01_VT.nc will compute the meridional heat and salt transport for month 1 of year 0010 for the ORCA025-G32 experiment.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc ,mask.nc, new_maskglo.nc
This latter file holds sub basin 2D masks; if it doesn't exist, only the global mask is taken into account, which is usefull for regional configs such as NATL4.

Output: zonal_heat_trp.dat and zonal_salt_trp.dat which are ASCII files
Example for zonal_heat_trp.dat

Zonal heat transport (integrated along I-model coordinate) (in Pw)												
J	Global		Atlantic		Pacific		Indian		Mediterranean		Austral	
...												
580	19.959	1.9821	19.959	1.0471	19.959	0.8080	19.959	0.1460	999.000	0.0000	999.000	0.0000
579	19.723	1.9769	19.724	1.0465	19.724	0.8108	19.724	0.1528	999.000	0.0000	999.000	0.0000
578	19.488	1.9650	19.488	1.0429	19.488	0.8054	19.488	0.1575	999.000	0.0000	999.000	0.0000
577	19.252	1.9512	19.252	1.0388	19.252	0.7975	19.252	0.1608	999.000	0.0000	999.000	0.0000
576	19.016	1.9334	19.016	1.0334	19.016	0.7840	19.016	0.1650	999.000	0.0000	999.000	0.0000
575	18.779	1.9103	18.779	1.0252	18.779	0.7615	18.779	0.1712	999.000	0.0000	999.000	0.0000
574	18.543	1.8792	18.543	1.0173	18.543	0.7235	18.543	0.1778	999.000	0.0000	999.000	0.0000
573	18.305	1.8406	18.305	1.0007	18.305	0.6837	18.305	0.1818	999.000	0.0000	999.000	0.0000
572	18.068	1.8064	18.068	0.9856	18.068	0.6455	18.068	0.1849	999.000	0.0000	999.000	0.0000
571	17.830	1.7768	17.830	0.9721	17.830	0.6171	17.830	0.1876	999.000	0.0000	999.000	0.0000
570	17.592	1.7494	17.592	0.9582	17.592	0.6000	17.592	0.1911	999.000	0.0000	999.000	0.0000
...												

First column indicates the corresponding J coordinate. Then pairs of column indicates the mean latitude and the transport. Heat transports are in Pw. Salt transports are in KT/s

Remark: missing values are indicated by 999.000

Associated script: *cdfmhst.ll*.

cdfmhst-full:

Purpose: Compute the Meridional Heat and Salt Transport (Full Step case).

Usage: *cdfmhst-full VTfile*

Input: VTfiles are the files produced by the cdfvT program. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means). For instance: cdfmhst-full ORCA025-G04_y0010m01_VT.nc will compute the meridional heat and salt transport for month 1 of year 0010 for the ORCA025-G04 experiment.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc ,mask.nc, new_maskglo.nc
This latter file holds sub basin 2D masks.

Output: zonal_heat_trp.dat and zonal_salt_trp.dat which are ASCII files
Example for zonal_heat_trp.dat

Zonal heat transport (integrated along I-model coordinate) (in Pw)												
J	Global		Atlantic		Pacific		Indian		Mediterranean		Austral	
...												
580	19.959	1.9821	19.959	1.0471	19.959	0.8080	19.959	0.1460	999.000	0.0000	999.000	0.0000
579	19.723	1.9769	19.724	1.0465	19.724	0.8108	19.724	0.1528	999.000	0.0000	999.000	0.0000
578	19.488	1.9650	19.488	1.0429	19.488	0.8054	19.488	0.1575	999.000	0.0000	999.000	0.0000
577	19.252	1.9512	19.252	1.0388	19.252	0.7975	19.252	0.1608	999.000	0.0000	999.000	0.0000
576	19.016	1.9334	19.016	1.0334	19.016	0.7840	19.016	0.1650	999.000	0.0000	999.000	0.0000
575	18.779	1.9103	18.779	1.0252	18.779	0.7615	18.779	0.1712	999.000	0.0000	999.000	0.0000
574	18.543	1.8792	18.543	1.0173	18.543	0.7235	18.543	0.1778	999.000	0.0000	999.000	0.0000
573	18.305	1.8406	18.305	1.0007	18.305	0.6837	18.305	0.1818	999.000	0.0000	999.000	0.0000
572	18.068	1.8064	18.068	0.9856	18.068	0.6455	18.068	0.1849	999.000	0.0000	999.000	0.0000
571	17.830	1.7768	17.830	0.9721	17.830	0.6171	17.830	0.1876	999.000	0.0000	999.000	0.0000
570	17.592	1.7494	17.592	0.9582	17.592	0.6000	17.592	0.1911	999.000	0.0000	999.000	0.0000
...												

First column indicates the corresponding J coordinate. Then pairs of column indicates the mean latitude and the transport. Heat transports are in Pw. Salt transports are in KT/s

Remark: missing values are indicated by 999.000

Associated script: *cdfmhst-full.ll*

cdfhflx:

Purpose: Compute the Meridional Heat Transport from the forcing fields

Usage: *cdfhflx gridTfile*

Input: gridTfile is a file which hold the flux variable *sohefldo*.

Required mesh_mask files or other files: mesh_hgr.nc, new_maskglo.nc

This latter file holds sub basin 2D masks; if it doesn't exist, only the global mask is taken into account, which is useful for regional configs such as NATL4.

Output: Results are displayed on the standard output, with columns corresponding to sub basins.

Remark: This computation is relative to the starting point of the integration. Transports are assumed to vanish at the northern point of the domain.

Associated script: none for the moment.

cdfvhst:

Purpose: Compute the Vertically Integrated Heat and Salt Transport (Partial Step case).

Usage: *cdfvhst VTfile*

Input: VTfiles are the files produced by the cdfvT program. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means). For instance: cdfvhst ORCA025-G32_y0010m01_VT.nc will compute the vertically integrated heat and salt transport for month 1 of year 0010 for the ORCA025-G32 experiment.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc, att.txt

Output: trp.nc, variables somevt somevs sozout sozous

Remark: for example:

$$somevt(i, j) = \int_{-h}^0 vt(i, j) e_{1v}(i, j) e_3(i, j, z) dz \quad (1)$$

$$someut(i, j) = \int_{-h}^0 ut(i, j) e_{2u}(i, j) e_3(i, j, z) dz \quad (2)$$

Associated script: *cdfvhst.ll*.

cdfvhst-full:

Purpose: Compute the Vertically Integrated Heat and Salt Transport (Full Step case).

Usage: *cdfvhst-full VTfile*

Input: VTfiles are the files produced by the cdfvT program. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means). For instance: cdfvhst-full ORCA025-G04_y0010m01_VT.nc will compute the vertically integrated heat and salt transport for month 1 of year 0010 for the ORCA025-G04 experiment.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc, att.txt

Output: trp.nc, variables somevt somevs sozout sozous

Remark: for example:

$$somevt(i, j) = \int_{-h}^0 vt(i, j) e_{1v}(i, j) e_3(z) dz \quad (3)$$

$$someut(i, j) = \int_{-h}^0 ut(i, j) e_{2u}(i, j) e_3(z) dz \quad (4)$$

Associated script: *cdfvhst-full.ll*.

cdfpsi:

Purpose: Compute the Barotropic Stream Function (Partial Step case).

Usage: *cdfpsi Ufile Vfile [V]*

Input: Ufile and Vfile are the files holding vozocrtx and vomecrtx. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means).

The optional [V] parameter, is used to indicate that we want to save the v-computed psi, instead of the default u-computed. This last option is usefull for basin such as NATL4.

For instance: *cdfpsi ORCA025-G32_y0010m01_U.nc ORCA025-G32_y0010m01_V.nc* will compute the BSF for month 1 of year 0010 for the ORCA025-G32 experiment.

Required mesh_mask files or other files: *mesh_hgr.nc, mesh_zgr.nc, mask.nc, att.txt*

Output: *psi.nc*, variables *sobarstf*, on the C-grid f-points, masked.

Remark: This program is prepared to compute BSF either from the U field or the V field. To be in agreement with previous matlab programs, we choose to save only the result from the U field.

The integration constant is set so that the BSF on Asia is 0. (For Orca type simulations, point (jpglo, jpglo) is supposed to be in Asia). Discussion is open if it is better to save the mean value of the BSF derived from U field and V field.

Associated script: *cdfpsi.ll*.

cdfpsi-full:

Purpose: Compute the Barotropic Stream Function (Full Step case).

Usage: *cdfpsi-full Ufile Vfile*

Input: Ufile and Vfile are the files holding vozocrtx and vomecrtx. They correspond to a certain period of time (monthly, quarterly, annual or pluri annual means). For instance: cdfpsi-full ORCA025-G04_y0010m01_U.nc ORCA025-G04_y0010m01_V.nc will compute the BSF for month 1 of year 0010 for the ORCA025-G04 experiment.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc, mask.nc, att.txt

Output: psi.nc, variables sobarstf, on the C-grid f-points, masked.

Remark: This program is prepared to compute BSF either from the U field or the V field. To be in agreement with previous matlab programs, we choose to save only the result from the U field. The integration constant is set so that the BSF on Asia is 0. (For Orca type simulations, point (jpglo, jpglo) is supposed to be in Asia). Discussion is open if it is better to save the mean value of the BSF derived from U field and V field.

Associated script: *cdfpsi-full.ll*.

cdfmoc:

Purpose: Compute the Meridional Overturning Circulation (partial step case), from the meridional velocity and a basin mask file.

Usage: *cdfmoc Vfile*

Input: Vfile is the files holding vomecrty. For instance: *cdfmoc ORCA025-G32.y0010_ANNUAL_gridV.nc* will compute the MOC for with ORCA025-G32 annual mean V field.

Required mesh_mask files or other files: *mesh_hgr.nc*, *mesh_zgr.nc*, *mask.nc*, *new_maskglo.nc* and *att.txt* (if *new_maskglo.nc* doesn't exist, it will work for the global basin; usefull for NATL4 like configs).

Output: The results are output on the file *moc.nc*. There are 5 variables concerning the MOC, one for each sub-basin. They are degenerated 3D variables with the i-dimension reduced to 1 : *zomsfglo* for the GLObal ocean, *zomsfatl* for the ATLantic ocean, *zomsfinp* for the INdoPacific ocean, *zomsfind* for the INDian ocean and finally *zomsfpac* for the PACific ocean.

Notice that the depth associated to the file corresponds to the W depth (gdepw). *nav_lon* is arbitrarily set to 0 (*nav_lon*(1,1 : *npjglo*)), and *nav_lat* is set to the latitude of the i-line going through the North Pole.

Remark: The name Meridional Overturning is a facility of language, because in fact what is computed is the along-I integral of the V component (which is not *stricto sensus* meridional — it follows the J-coordinate —)

Associated script: *cdfmoc.ll*.

cdfmocsig:

Purpose: Compute the Meridional Overturning Circulation (partial step case) in function of σ_1 from the meridional velocity, a TS file and a basin mask file.

Usage: *cdfmocsig Vfile TSfile*

Input: Vfile is the files holding vomecrty, TSfile hold votemper, vosaline for density computation.

For instance: *cdfmocsig ORCA025-G32_y0010_ANNUAL_gridV.nc ORCA025-G32_y0010_ANNUAL_gridT.nc* will compute the MOC for with ORCA025-G32 annual mean V field.

Required mesh_mask files or other files: *mesh_hgr.nc, mesh_zgr.nc, mask.nc, new_maskglo.nc* and *att.txt*

Output: The results are output on the file *mocsig.nc*. There are 5 variables concerning the MOC, one for each sub-basin. They are degenerated 3D variables with the i-dimension reduced to 1 : *zomsfglo* for the GLObal ocean, *zomsfatl* for the ATLantic ocean, *zomsfinp* for the INdoPacific ocean, *zomsfind* for the INDian ocean and finally *zomsfpac* for the PACific ocean.

Notice that the depth associated to the file corresponds to the W depth (gdepw). *nav_lon* is arbitrarily set to 0 (*nav_lon(1,1 : npjglo)*), and *nav_lat* is set to the latitude of the i-line going through the North Pole.

Remark: The name Meridional Overturning is a facility of language, because in fact what is computed is the along-I integral of the V component (which is not *stricto sensus* meridional — it follows the J-coordinate —)

Associated script: *cdfmoc.ll*.

cdfmoc-full:

Purpose: Compute the Meridional Overturning Circulation (partial step case), from the meridional velocity and a basin mask file.

Usage: *cdfmoc-full Vfile*

Input: Vfile is the files holding vomecrty. For instance: `cdfmoc-full ORCA025-G04_y0010_ANNUAL_gridV.nc` will compute the MOC for with ORCA025-G04 annual mean V field.

Required mesh_mask files or other files: `mesh_hgr.nc`, `mesh_zgr.nc`, `mask.nc`, `new_maskglo.nc` and `att.txt`

Output: The results are output on the file *moc.nc*. There are 5 variables concerning the MOC, one for each sub-basin. They are degenerated 3D variables with the i-dimension reduced to 1 : *zomsfglo* for the GLObal ocean, *zomsfatl* for the ATLantic ocean, *zomsfinp* for the INdoPacific ocean, *zomsfind* for the INDian ocean and finally *zomsfpac* for the PACific ocean.

Notice that the depth associated to the file corresponds to the W depth (`gdepw`). *nav_lon* is arbitrarily set to 0 (*nav_lon*(1,1 : *npjglo*)), and *nav_lat* is set to the latitude of the i-line going through the North Pole.

Remark: The name Meridional Overturning is a facility of language, because in fact what is computed is the along-I integral of the V component (which is not *stricto sensus* meridional — it follows the J-coordinate —)

Associated script: *cdfmoc-full.ll*.

cdfmocat1:

Purpose: Compute the Meridional Overturning Circulation (partial step case), from the meridional velocity and a basin mask file. This program computes the MOC for one basin only. Useful for NATL4. It is now obsolete as cdfmoc does the same, if no new_maskglo.nc are available.

Usage: *cdfmoc Vfile*

Input: Vfile is the files holding vomecrty. For instance: `cdfmocat1 NATL4-G07_y0010_ANNUAL_gridV.nc` will compute the MOC for with NATL4-G07 annual mean V field.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc, mask.nc, and att.txt

Output: The results are output on the file *moc.nc*. The variable concerning the MOC is a degenerated 3D variables with the i-dimension reduced to 1 : *zomsfatl* for the ATLantic ocean

Notice that the depth associated to the file corresponds to the W depth (gdepw). *nav_lon* is arbitrarily set to 0 (*nav_lon*(1,1 : *npjglo*)), and *nav_lat* is set to the latitude of the i-line going through the North Pole.

Remark: The name Meridional Overturning is a facility of language, because in fact what is computed is the along-I integral of the V component (which is not *stricto sensus* meridional — it follows the J-coordinate —)

Associated script: *cdfmocat1.ll.* (*to be written*)

cdftransportiz:

Purpose: Compute volume, heat and salt transport across a section, for depth classes. (Partial Step case)

Usage: *cdftransportiz [-test u v] VTfile gridUfile gridVfile 'limit of level'*

Input: VTfiles are the files produced by the cdfvT program.

gridV U and gridV files are equivalent files (same period) for U and V.

'limit of levels' are depth in meters where to set limits for depth classes. If no limits are given, the transport is computed for the whole water column.

Once the data files are read, the user is asked to give a section name (or 'EOF' for ending the program), followed by the geographical limits of a section (imin imax jmin jmax, in model coordinates). This user interaction can be done with an ascii file given as standard input. (cdftransportiz < section.dat, for instance).

For instance: cdftransportiz ORCA025-G32_y0010m01_VT.nc ORCA025-G32_y0010m01_gridU.nc ORCA025-G32_y0010m01_gridV.nc 1500 3000 < section .dat

will compute the transports across a section (given interactively) in three depth classes : from top to 1500 m, from 1500 m to 3000 m and from 3000 m to the bottom, for section described into section.dat file.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc

Output: section_htrp.dat, section_strp.dat, section_vtrp.dat, 3 ASCII files.

Remark: The sign of the transport is somewhat tricky. It depends on the inclination of the section.

As a rule of the thumb, the transport is > 0 when going to the right hand side of the section, cruising the section from (imin,jmin) (imax, jmax). When the optional '-test u v ' arguments of the command line are given, the velocity field is assumed to be a constant field with both U and V taken as the arguments. This gives an easy way to check the sign of the transports for a given section.

Associated script: *cdftransportiz.ll*.

cdftransportiz-full:

Purpose: Compute volume, heat and salt transport across a section, for depth classes. (Partial Step case)

Usage: *cdftransportiz-full [-test u v] VTfile gridUfile gridVfile 'limit of level'*

Input: VTfiles are the files produced by the cdfvT program.

gridV U and gridV files are equivalent files (same period) for U and V.

'limit of levels' are depth in meters where to set limits for depth classes. If no limits are given, the transport is computed for the whole water column.

Once the data files are read, the user is asked to give a section name (or 'EOF' for ending the program), followed by the geographical limits of a section (imin imax jmin jmax, in model coordinates). This user interaction can be done with an ascii file given as standard input. (cdftransportiz-full < section.dat, for instance).

For instance: cdftransportiz-full ORCA025-G04_y0010m01_VT.nc ORCA025-G04_y0010m01_gridU.nc ORCA025-G04_y0010m01_gridV.nc 1500 3000 < section .dat

will compute the transports across a section (given interactively) in three depth classes : from top to 1500 m, from 1500 m to 3000 m and from 3000 m to the bottom, for section described into section.dat file.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc

Output: section_htrp.dat, section_strp.dat, section_vtrp.dat, 3 ASCII files.

Remark: The sign of the transport is somewhat tricky. It depends on the inclination of the section.

As a rule of the thumb, the transport is > 0 when going to the right hand side of the section, cruising the section from (imin,jmin) (imax, jmax). When the optional '-test u v ' arguments of the command line are given, the velocity field is assumed to be a constant field with both U and V taken as the arguments. This gives an easy way to check the sign of the transports for a given section.

Associated script: *cdftransportiz-full.ll*.

cdfmasstrp:

Purpose: Compute volume transport across a section, for depth classes. (Partial Step case)

Usage: *cdfmass [-test u v] gridUfile gridVfile 'limit of level'*

Input: gridV U and gridV files are simultaneous velocity component files.

'limit of levels' are depth in meters where to set limits for depth classes. If no limits are given, the transport is computed for the whole water column.

Once the data files are read, the user is asked to give a section name (or 'EOF' for ending the program), followed by the geographical limits of a section (imin imax jmin jmax, in model coordinates). This user interaction can be done with an ascii file given as standard input. (cdftransportiz < section.dat, for instance).

For instance: cdfmasstrp ORCA025-G32_y0010m01_gridU.nc ORCA025-G32_y0010m01_gridV.nc
1500 3000 < section .dat

will compute the transports across a section (given interactively) in three depth classes : from top to 1500 m, from 1500 m to 3000 m and from 3000 m to the bottom, for section described into section.dat file.

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc

Output: section_htrp.dat, section_strp.dat, section_vtrp.dat, 3 ASCII files.

Remark: The sign of the transport is somewhat tricky. It depends on the inclination of the section.

As a rule of the thumb, the transport is > 0 when going to the right hand side of the section, cruising the section from (imin,jmin) (imax, jmax). When the optional '-test u v ' arguments of the command line are given, the velocity field is assumed to be a constant field with both U and V taken as the arguments. This gives an easy way to check the sign of the transports for a given section.

This program is a simplification of cdftransportiz where the heat and salt transport are not computed anymore.

cdfsigtrp:

Purpose: Compute density class transport for either zonal or meridional section, (partial steps).

Usage: *cdfsigtrp gridTfile gridUfile gridVfile sigma_min sigma_max nbins [options]*

Input: T, U and V files are output file from the model.

sigma_min sigma_max gives the lower and upper limit for the density classes σ_0 ,

nbins is the number of bins (density classes) to explore between σ_{min} and σ_{max} .

The program needs an ascii file called **dens_section.dat** where the sections are described.

An example of such a file is given below:

```
01_Denmark_strait
1013 1056 832 832
02_Faoes_Bank_Channel
1106 1106 800 808
03_Gibraltar
1126 1126 651 655
EOF
```

Note the 'EOF' keyword in the file, which indicates the end of the file for the program.

For instance:

```
cdfsigtrp ORCA025-G50_y1949_ANNUAL_gridT.nc ORCA025-G50_y1949_ANNUAL_gridU.nc
ORCA025-G50_y1949_ANNUAL_gridV.nc 26 30 80
```

will compute the density class tranport for 80 classes, between $\sigma_0 = 26$ and $\sigma_0 = 30$ for the sections described in the file **dens_section.dat**

Options: 2 options are available :

- print : give extra informations on the standard output for the sections. These are 2D arrays for each section, giving a printed 'map' of density, depth, e3 in the (distance,vertical) plane. There are also other 2D output for printed map of isopycnal depth, cumulated transports, and bined transports in the (distance, density) plane. (Useful for short sections !!)
- bing : Two bing files are produced per section. (1) (x/y,depth) for T,S, σ_0 ,U (2) (x/y, σ) for hiso and class transport. It can be used whith sections of any size. It allows a more detailed description of the flow than the standard output of the program (integrated transport along the section).

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc, dens_section.dat

Output: The standard output of this program is an ASCII file called trpsig.txt which is multicolumns file with the first column giving the density, and other columns the density class transport (in sverdrup) for each section (1 column per section). For example (3 sections):

```
.../
27.0000 -0.3498706E+05 0.0000000E+00 0.9302920E+04
27.0500 -0.2951592E+05 0.0000000E+00 0.2953659E+05
27.1000 -0.1460002E+05 0.0000000E+00 0.6516903E+04
27.1500 -0.1678582E+05 0.0000000E+00 0.6516903E+04
27.2000 -0.5445088E+04 0.0000000E+00 0.6516903E+04
27.2500 0.7251206E+04 -0.1140499E+06 0.6062048E+04
27.3000 -0.1079158E+03 -0.7521843E+05 0.5776318E+04
27.3500 0.2931429E+03 -0.7162286E+05 0.5776318E+04
27.4000 0.7215664E+03 -0.1958259E+06 0.5776318E+04
27.4500 0.1075893E+05 -0.2733888E+06 0.6497963E+05
.../
```

First line of this example, indicates the transport for the density class [27.00,27.05[. (line k is the transport for the density class $[\sigma_k, \sigma_{k+1}[$).

Remark: The sign of the transport is > 0 for northward and eastward transports.

3 Derived quantities

cdfsig0:

Purpose: Compute the potential density σ_0 .

Usage: *cdfsig0 gridT*

Input: gridT is a file holding variables votemper and vosaline. For instance: cdfsig0 ORCA035-G32_y0008m01d10_gridT.nc will compute σ_0 for the given tag.

Required mesh_mask files or other files: att.txt

Output: *sig0.nc*. This file hold the variable vosigma0.

Remark: The non-linear equation of state of NEMO OPA9 is used. Therefore, one should be aware that σ_0 computed from mean temperature and salinity fields is not the same than the mean σ_0 computed from instantaneous temperature and salinity fields.

Associated script: *cdfsigma0.ll*. This script can be used to compute σ_0 for a run. It scan all the gridT file for a given year and CONFIG, and produce the corresponding sigma0 file.

cdfbottomsig0:

Purpose: Compute the bottom potential density σ_{0bot} (2D variable).

Usage: *cdfbottomsig0 gridT*

Input: gridT is a file holding variables votemper and vosaline.

For instance: *cdfbottomsig0 ORCA035-G32_y0008m01d10_gridT.nc* will compute σ_{0bot} for the given tag.

Required mesh_mask files or other files: att.txt

Output: *botsig0.nc*. This file hold the variable sobotsig0.

Remark: The non-linear equation of state of NEMO OPA9 is used. Therefore, one should be aware that σ_0 computed from mean temperature and salinity fields is not the same than the mean σ_0 computed from instantaneous temperature and salinity fields. A companion of this program is cdfbottom.

Associated script: *cdfbotsig0.ll*. This script can be used to compute σ_{0bot} for a run. It scan all the gridT file for a given year and CONFIG, and produce the corresponding botsig0 file.

cdfbottom:

Purpose: Extract the bottom values for the 3D variables contained in the file given as argument.
Bottom values are the values for the last point of the water column before land.

Usage: *cdfbottom nc_file [T | U | V | F]* If the point type is specified, look for the corresponding mask in mask.nc file.

Input: Any netcdf file from NEMO output (or diagnosed from NEMO) , containing 3D fields. For instance: *cdfbottom ORCA035-G32_y0008m01d10_gridT.nc* will produce a file with the 2D variables *votemper* and *vosaline* corresponding the their bottom value.

cdfbottom ORCA035-G32_y0008m01d10_gridU.nc U will produce a file with the 2D variable *vozocrtx*, corresponding the the bottom value, the umask being read on the mask.nc file.

Required mesh_mask files or other files: *att.txt* and eventually *mask.nc* file if the type point of the C grid is specified.

Output: *bottom.nc*. This file hold the variable(s) having the same name than the 3D variables of the input file.

Remark: For the sake of simplicity, and for compatibility with other cdftools, we keep the same variable name in the output file than in the input file, though the output variables are 2D and should 'logically' start with 'so'...

Associated script: *cdfbottom.ll*. This script can be used to compute bottom value for a run.

cdfrhoproj:

Purpose: Project a variable of a netcdf file on an isopycnal surface either specified by a σ_0 on the command line, or taken from an input file.

Usage: `cdfrhoproj [-s0 sigma0] cvar nc_rhofile nc_file(*) [T | U | V | F]` If the value of σ_0 for the iso surface is given on the command line, then only this surface is calculated. In other cases, the values for σ_0 are taken from the local *rho_lev* file, which is a very simple ascii file, with the number of surfaces given on the first line, followed by lines with the required σ_0 values. CAUTION: these values must increase in the file.

Input: nc_rhofile already contains the density, cvar is the name of the variable that will be projected on the isopycnal, taken from file nc_file. Various files can be specified. In order to handle the NEMO C-Grid, and as far as σ_0 is computed on a T-point, it is possible to specify the grid point type corresponding to the file. (If nothing specified, 'T' is assumed.). In any case, the resulting value are computed on the T-Point. For instance:

```
cdfrhoproj vozocrtx CONFIG-CASE_SIGMA0.nc CONFIG-CASE_gridU.nc U
will project the zonal velocity on the isopycnal defined in rho_lev at the T-Points.
cdfrhoproj -s0 27.8 vosaline CONFIG-CASE_SIGMA0.nc CONFIG-CASE_gridT.nc
will project the salinity on the 27.8 isopycnal.
```

Required mesh_mask files or other files: att.txt and rho.lev (if the option -s0 is not given).

Output: *nc_file.nc.interp*. The suffix **.interp** is appended to the input file to produce the output file. The variable name is the same, and an additional variable **vodepiso** is provided; it gives the depth of the isopycnal surfaces. In the output file, the dimension **deptht** is still used, but now represent the rho levels.

Remark: none

Associated script: not done already

cdfbn2:

Purpose: Compute the Brunt Vaissala frequency.

Usage: *cdfbn2 gridT*

Input: gridT is a file holding variables votemper and vosaline. For instance: cdfbn2 ORCA035-G32_y0008m01d10_gridT.nc
will compute N^2 for the given tag.

Required mesh_mask files or other files: mesh_zgr.nc, mesh_hgr.nc and att.txt

Output: *bn2.nc*. This file hold the variable vobn2.

Remark: This program uses the eosbn2 routine of the NEMO OPA9 code. It is based on an approximation formula given by Mc Dougall et al. ().

Associated script: *cdfbn2.ll*. This script can be used to compute N^2 for a run. It scan all the gridT file for a given year and CONFIG, and produce the corresponding N^2 file.

cdfets:

Purpose: Compute the Eddy Time Scale, and Rossby Radius.

Usage: *cdfets gridT*

Input: gridT is a file holding variables votemper and vosaline. For instance: cdfets ORCA035-G32_y0008m01d10_gridT.nc
will compute the eddy time scale and the 1st rossby radius for the given tag.

Required mesh_mask files or other files: mesh_zgr.nc, mesh_hgr.nc and att.txt

Output: *ets.nc*. This file hold two variables voets and sorosrad.

Remark: This routine is based on papers from Stammer et al (). See Julien.LeSommer@hmg.inpg.fr for more details.

Associated script: *cdfets.ll*.

cdfcurl:

Purpose: Compute the curl of a vector field.

Usage: *cdfcurl gridU gridV nameU nameV level*

If level is different from 0, the 2D curl at level 'level' will be computed. If level is ≤ 0 , then the full 3D curl is computed instead.

Input: gridU, gridV are the cdf files holding the U and V component of the vector, nameU and nameV the cdf name of the variables corresponding to these components. level in the model level where to compute the curl (if ≤ 0 the 3D curl is computed instead). For instance: *cdfcurl ORCA025-G32_y0008m01d10_gridU.nc ORCA025-G32_y0008m01d10_gridV.nc vozocrtx vomecrty 6* will compute the curl (relative vorticity) at level 6 for the flow field at the given tag. *cdfcurl ORCA025-G32_y0008m01d10_gridU.nc ORCA025-G32_y0008m01d10_gridV.nc sozo-taux sometauy 1* will compute the wind-stress curl from the file.

Required mesh_mask files or other files: att.txt, mesh_hgr.nc

Output: *curl.nc*. The variable name is socurl(2D) or vocurl (3D case).

Associated script: none

cdfpv:

Purpose: Compute the full potential vorticity (computed at W-point of the C-grid) taking T S U V as input.

Usage: *cdfpv gridT gridU gridV*

Input: gridT, gridU, gridV are the cdf files holding the temperature and salinity (gridT) and the U and V components of the velocity. Variables names are assumed to be as in OPA9, respectively *votemper*, *vosaline*, *vozocrtx*, *vomecrtx*.

For instance: *cdfpv ORCA025-G70_y2000m01d05_gridT.nc ORCA025-G70_y2000m01d05_gridU.nc ORCA025-G70_y2000m01d05_gridV.nc* will compute the potential vorticity.

Method: The potential vorticity is evaluated using the following formula :

$$PV = \frac{1}{\rho_0} (f + \zeta) \frac{\partial \sigma_0}{\partial z}$$

where ρ_0 takes the constant value of 1020 kg/m^3 , f is the coriolis parameter ($2\Omega \sin(\phi)$), ζ is the relative vorticity (or the curl) of the flow (as computed by *cdfcurl*). This program assumes that the model grid is a C-grid. PV is computed at W points, which is the natural point to compute $\frac{\partial \sigma_0}{\partial z}$. ζ is computed at f-points, u-v level. Therefore, in order to estimate ζ at W points, a 4-point horizontal average is required (to get ζ at T-points) followed by a 2-point vertical average to get it at W-points.

Required mesh_mask files or other files: *att.txt*, *mesh_hgr.nc*, *mesh_zgr*

Output: *pv.nc*. The variable name is *vopv*. Units are $s^{-1}m^{-1}x10^{11}$

Associated script: none

cdfw:

Purpose: Compute the vertical velocity field from the continuity equation and the horizontal flow field.

This is for partials steps.

Usage: *cdfw gridU gridV [nameU nameV]*

Input: gridU, gridV are the cdf files holding the U and V component of the vector, nameU and nameV the cdf name of the variables corresponding to these components. If they are not given, we assume that these names are respectively vozocrtx, and vomecrtz. For instance:
cdfw ORCA025-G32_y0008m01d10_gridU.nc ORCA025-G32_y0008m01d10_gridV.nc
will compute the wn (vertical velocity) for the flow field at the given tag.
cdfw vitU.nc vitV.nc uzonal vmeridional
will compute wn from vitU.nc and vitV.nc files.

Required mesh_mask files or other files: att.txt, mesh_hgr.nc, mesh_zgr.nc

Output: *w.nc*. The variable name is vovecrtz.

Associated script: none

Remark: Comparison of wn computed by the model and wn computed by this program shows small differences ($O(10^{-10})$) due to truncature on the flow field.

cdfmxl:

Purpose: Compute the mixed layer depth based on 3 different criteria: density criterion with $\rho_{crit} = 0.01 \text{ kg.m}^{-3}$, density criterion with $\rho_{crit} = 0.03 \text{ kg.m}^{-3}$ and temperature criteria with $|T_{crit}| = 0.2^\circ\text{C}$.

Usage: *cdfmxl gridT*

Input: The only file on input is the gridT type file where the program will look for temperature as *votemper*, and salinity as *vosaline*. For instance: *cdfmxl ORCA025-G42-y0008m01d10_gridT.nc* will compute the mixed layer depth based on the 3 criteria.

Required mesh_mask files or other files: *att.txt*, *mesh_hgr.nc*, *mesh_zgr.nc*

Output: *mxl.nc*. The variable names are *somxl010*, *somxl030* and *somxlt02*.

Associated script: *cdfmxl.ll* is a script which computes the mixed layer for each gridT file in a directory.

Remark: There is a recent climatology of the *somxlt02* variable [de Boyer Montegut *et al.*(2004)], which is built from observations of temperature profiles.

cdficediags:

Purpose: Compute ice volume, area and extend (defined as the area where the ice concentration > 0.15) for both hemisphere.

Usage: *cdficediags icemodfile*

Input: ncdf file for icemod output

Required mesh_mask files or other files: mesh_hgr.nc, mask.nc

Output: The output is done on the standard output. (see below).

Associated script: cdficediags.ll is used for monitoring the ice in ORCA025-G50

Remark: The ice volume is $\sum thickness * area * fraction$

Example of output :

Northern Hemisphere	
NVolume (10 ⁹ m3)	35268.1868656639999
NArea (10 ⁹ m2)	13113.4348328960004
NExtend (10 ⁹ m2)	13062.9962301439991
Southern Hemisphere	
SVolume (10 ⁹ m3)	4879.33240934399964
SArea (10 ⁹ m2)	3477.76693043200021
SExtend (10 ⁹ m2)	3394.26987212799986

cdfcensus:

Purpose: Compute the water mass census for a given TS file, with eventual limitation to a specified area.

Usage: *cdfcensus gridTfile nlog [-zoom imin imax jmin jmax] [-klm kmin kmax]*

The program computes the water mass census as the T,S binned volume for the whole area or a restricted area specified by the line options. The output is given as an array (S,T) where the value of the array is the volume in the corresponding T,S bin. Additionally, a second information is given as $\sigma_0(S,T)$ computed from the EOS. (Plotting purposes).

nlog is an integer number ≥ 0 which is used to distort the output: in fact some water masses are extremely dominant in the ocean, with volumes many order of magnitudes above other interesting waters. In order to rescale the output, we apply the following lines of code, as soon as nlog > 0 :

```
! use a distortion function ( n x log ) to reduce extrema in the output file.
DO ji=1,ns
  DO jj=1,nt
    dump(ji,jj)=rcensus(ji,jj)
    DO ilog=1,nlog
      dump(ji,jj)=ALOG10(1+dump(ji,jj))
    END DO
  END DO
END DO
```

Input: ncdf file for gridT

Required mesh_mask files or other files: mesh_hgr.nc, mesh_zgr.nc

Output: Output on bingfile called census.bing. The unit of the output is somewhat tricky, depending on the number of log rescaling that where used. Soon, a netcdf file output will be available.

Associated script: none

4 Extracting and information tools

cdfprofile:

Purpose: Extract a vertical profile for a given variable in a given file at a given I J

Usage: *cdfprofile I J file_name var_name*

Input: I J : i, j position where to look at the profile

file_name : name of the file

var_name : name of the variable

For instance: *cdfprofile 32 45 ORCA035-G32_y0008m01d10_gridT.nc votemper*

Required mesh_mask files or other files: none

Output: output is done on standard output

Associated script: none

cdfwhereij:

Purpose: Give the longitude and latitude of the (i,j) points from a coordinate file.

Usage: *cdfwhereij imin imax jmin jmax coordinate_file point_type*

Input: imin, imax, jmin, jmax : zoom in i,j coordinates

coordinate_file : either a coordinate or a mesh_hgr file

point_type : either T, U, V or F in upper or lower case

For instance: *cdfwhereij 32 45 123 432 coordinate_orca025.nc f*

will give the zoom position in longitude latitude coordinates, for the f points

Required mesh_mask files or other files: either coordinates or mesh_hgr files.

Output: output is done on standard output

Associated script: none

cdffindij:

Purpose: Give the i,j corresponding to longitude and latitude given as arguments.

Usage: *cdffindij xmin xmax ymin ymax [coordinate_file] [point_type]*

Input: xmin, xmax, ymin, ymax : zoom in geographical coordinates

coordinate_file : either a coordinate or a mesh_hgr file. If not given, assumes *coordinates.nc*

point_type : either T, U, V or F in upper or lower case. If not given assumes *F*

For instance: *cdffindij -30 0 -20 40 coordinate_orca025.nc f*

will give the zoom position imin imax jmin jmax for the given configuration, for nearest f-points.

cdffindij -180 0 -20 25 coordinates_ORCA_R025_lombok+ombai.nc F
gives:

```
rdis = 0.1316580027
rdis = 0.1287123561
430 1149 417 602
-179.88 -0.12 -19.96 25.03
```

rdis is a raw estimation (in Deg.) of the distance between the given position, and the real position. In some cases, the search algorithm fails (zoom across boundaries or in very distorted regions), and an error message is displayed.

Required mesh_mask files or other files: either coordinates or mesh_hgr files.

Output: output is done on standard output

Remark: The arguments must be in the proper order. In particular, if the user needs to specify the point_type it is necessary to also specify the name of the coordinate file.

Associated script: none

cdfmaxmoc:

Purpose: Give the max and min intensity of the MOC, previously computed with cdfmoc.

Usage: *cdfmaxmoc moc_file basin latmin latmax depmin depmax*

Input: moc_file is the netcdf file computed with *cdfmoc* or *cdfmoc - full*

basin is an indicator for the required basin. Can either atl, inp, ind, pac or glo.

latmin, latmax, depmin, depmax is the window (lat,dep) where the extrema are searched.

For instance: cdfmaxmoc ORCA025-G42_y0010_MOC.nc atl -30 70 200 5000

will indicates the max/min of the overturning and their respective location, for year 10 of the run ORCA025-G42, for Atlantic basin, limited to 30S-70N, between 200 m and 5000 m depth.

Required mesh_mask files or other files: none

Output: output is done on standard output

Remark: depmin and depmax are given as positive.

Associated script: none

cdfzoom:

Purpose: Shows an ASCII representation of a 2D (x-y, x-z or y-z) slab of any variable from an output file

Usage: *cdfzoom -f ncfile -zoom imin imax jmin jmax [-lev kmin kmax] [-fact scale_factor] [-var cdfvarname]*

Input: ncfile is the name of the file to look at.

imin,imax,jmin,jmax are the limits for the horizontal zoom.

kmin, kmax are the limits for the vertical zoom. If kmin = kmax, then the x-y slab is shown at level kmin. If kmax > kmin, then either a x-z or y-z slab will be shown, but in this case, either imin=imax, or jmin=jmax, otherwise the program will stop.

scale_factor is an optional dividing scale factor used to adjust the output values to the fortran format (f12.4)

cdfvarname is the name of the variable you want to look at. If not given, or wrong name, the program will propose the list of available variables.

For instance: *cdfzoom -f ORCA05-G50_y1949m01d30_gridT.nc -lev 1 43 -zoom 470 482 175 175 -var votemper*

will show the vertical slab (x-z) of the temperature field, at J=175, for I between 470 and 482, K from 1 to 43

Required mesh_mask files or other files: none

Output: output is done on standard output

Associated script: none

cdfmax:

Purpose: Display min/max of a variable in a file, and their respective location. A sub area can be specified either horizontally or vertically.

Usage: *cdfmax -f ncfile [-var cdfvarname] [-zoom imin imax jmin jmax] [-lev kmin kmax] [-fact scale_factor]*

Input: ncfile is the name of the file to look at.

cdfvarname is the name of the variable you want to look at. If not given, or wrong name, the program will propose the list of available variables.

imin,imax,jmin,jmax are the limits for the horizontal zoom.

kmin, kmax are the limits for the vertical zoom.

A vertical slab can thus be specified, playing around with the limits.

scale_factor is an optional multiplying scale factor usefull for units change.

For instance:

```
cdfmax -f ORCA05-G60_y1968m12d26_gridT.nc -zoom 500 500 300 500 -var votemper
votemper with multiplying factor of 1.000000000
i-slab MAX:  i    long  j    lat  k    dep    MaxValue    MIN:  i    long  j    lat  k    dep    MinValue
500          500 -37.75 300  24.24  8    63.88  0.24313E+02    500 -88.71 499  82.95  1    3.05 -0.17027E+01
```

will show the min/max temperature over the vertical slab (y-z) at I=500, for J between 300 and 500.

```
cdfmax -f ORCA05-G60_y1968m12d26_gridT.nc -fact 100 -var sossheig
sossheig with multiplying factor of 100.0000000
level  dep MAX:  i    long  j    lat    MaxValue MIN:  i    long  j    lat    MinValue
1      3.05      56 100.25 277 13.38  0.15471E+03    566 -4.75 92 -61.73 -0.13513E+03
```

will show the min/max SSH in cm.

Required mesh_mask files or other files: none

Output: output is done on standard output

Associated script: none

APPENDICES

A att.txt file

The att.txt file is used for a smooth management of the variables attributes of the netcdf files, particularly for the new variables, created by the cdftools programs. The first line give the name of the attribute, and the second line gives the type (T for text, F for float). Then, starting at line 4, for each variable name given in column 1, the attributes are documented. The actual att.txt file is as follows. Note that the very last line of the file is the key word 'EOF'.

varname	units	missing_value	valid_min	valid_max	long_name	short_name	online_operation	axis
T	T	F	F	F	T	T	T	T
votemper	C	0.	-2.	45.	Potential_Temperature	votemper	N/A	TZYX
vosaline	PSU	0.	0.001	40.	Salinity	vosaline	N/A	TZYX
vosigma0	kg/m3	0.	0.001	40.	Potential_density:sigma-0	vosigma0	N/A	TZYX
sobotsig0	kg/m3	0.	0.001	40.	Bottom_Potential_density	sobotsig0	N/A	TYX
vozocrtx	m/s	0.	-10.	10.	Zonal_Velocity	vozocrtx	N/A	TZYX
vomecrtx	m/s	0.	-10.	10.	Meridional_Velocity	vomecrtx	N/A	TZYX
vovecrtz	m/s	0.	-1.	1.	Vertical_Velocity	vovecrtz	N/A	TZYX
votkeavt	m2/s	0.	0.	100.	Vertical_Eddy_Diffusivity	votkeavt	N/A	TZYX
sozotaux	Pa	0.	-10.	10.	Zonal_Wind_Stress	sozotaux	N/A	TYX
sometauy	Pa	0.	-10.	10.	Meridional_Wind_Stress	sometauy	N/A	TYX
sosshheig	m	0.	-10.	10.	Sea_Surface_height	sosshheig	N/A	TYX
sohefldo	W/m2	0.	-1000.	1000.	Net_Downward_Heat_Flux	sohefldo	N/A	TYX
soshfldo	W/m2	0.	-1000.	1000.	Short_Wave_Radiation	soshfldo	N/A	TYX
solwfldo	W/m2	0.	-1000.	1000.	Long_Wave_Radiation	solwfldo	N/A	TYX
solhflup	W/m2	0.	-1000.	1000.	Latent_Heat_Flux	solhflup	N/A	TYX
sosbhflup	W/m2	0.	-1000.	1000.	Sensible_Heat_Flux	sosbhflup	N/A	TYX
sowafllup	kg/m2/s	0.	-1000.	1000.	Net_Upward_Water_Flux	sowafllup	N/A	TYX
sowafldp	kg/m2/s	0.	-1000.	1000.	Surface_Water_Flux:Damping	sowafldp	N/A	TYX
sosshheig_sqd	m2	0.	0.	100.	MS_Sea_Surface_height	sosshheig_sqd	N/A	TYX
sosshheig_rms	m	0.	0.	10.	RMS_Sea_Surface_height	sosshheig_rms	N/A	TYX
vozocrtx_sqd	m2/s2	0.	0.	100.	MS_Zonal_Velocity	vozocrtx_sqd	N/A	TZYX
vomecrtx_sqd	m2/s2	0.	0.	100.	MS_Meridional_Velocity	vomecrtx_sqd	N/A	TZYX
vovecrtz_sqd	m2/s2	0.	0.	100.	MS_Vertical_Velocity	vovecrtz_sqd	N/A	TZYX
vovecrtz_rms	m/s	0.	0.	1.	RMS_Vertical_Velocity	vovecrtz_rms	N/A	TZYX
voeke	m2/s2	0.	0.	10000.	Eddy_Kinetic_Energy	voeke	N/A	TZYX
vozout	mxdegC/s	0.	-100.	100.	Zonal_UT	UT	N/A	TZYX
vozous	mxPSU/s	0.	-100.	100.	Zonal_US	US	N/A	TZYX
vomevt	mxdegC/s	0.	-100.	100.	Meridional_VT	VT	N/A	TZYX
vomevs	mxPSU/s	0.	-100.	100.	Meridional_VS	VS	N/A	TZYX
sozout	W	0.	-100.	100.	Zonal_heat_transport	sozout	N/A	TYX
sozous	kg/sec	0.	-100.	100.	Zonal_salt_transport	sozous	N/A	TYX
somevt	W	0.	-100.	100.	Meridional_heat_transport	somevt	N/A	TYX
somevs	kg/sec	0.	-100.	100.	Meridional_salt_transport	somevs	N/A	TYX
isnowthi	m	0.	0.	100.	Snow_thickness	isnowthi	N/A	TYX
iicethic	m	0.	0.	100.	Ice_thickness	iicethic	N/A	TYX
iiceprod	m/kt	0.	0.	100.	Ice_produced	iiceprod	N/A	TYX
ileadfra	%	0.	0.	100.	Ice_concentration	ileadfra	N/A	TYX
iicetemp	C	0.	-30.	2.	Ice_Temperature	iicetemp	N/A	TYX
ioceflxb	W/m2	0.	-800.	800.	Ocean_ice_flux	ioceflxb	N/A	TYX
iicevelu	m/s	0.	-10.	10.	Ice_Velocity_u	iicevelu	N/A	TYX
iicevelv	m/s	0.	-10.	10.	Ice_Velocity_v	iicevelv	N/A	TYX
isstempe	C	0.	-3.	50.	Sea_Surface_Temperature	isstempe	N/A	TYX
issssalin	PSU	0.	0.01	40.	Sea_Surface_Salinity	issssalin	N/A	TYX
iocetflx	W/m2	0.	-800.	800.	Total_flux_at_ocean_surface	iocetflx	N/A	TYX
iocesflx	W/m2	0.	-800.	800.	Solar_flux_at_ocean_surface	iocesflx	N/A	TYX
iocwnsfl	W/m2	0.	-50.	50.	Non-solar_flux_at_ocean_surface	iocwnsfl	N/A	TYX
iocesafll	kg/m2/kt	0.	-50.	50.	Salt_flux_at_ocean_surface	iocesafll	N/A	TYX
iocestru	Pa	0.	-50.	50.	Wind_stress_u	iocestru	N/A	TYX
iocestrv	Pa	0.	-50.	50.	Wind_stress_v	iocestrv	N/A	TYX
iicesflx	W/m2	0.	-50.	50.	Solar_flux_at_ice/ocean_surface	iicesflx	N/A	TYX
iicenflx	W/m2	0.	-50.	50.	Non_Solar_flux_at_ice/ocean_surface	iicenflx	N/A	TYX
isnowpre	kg/day	0.	-50.	50.	Snow_precipitation	isnowpre	N/A	TYX
voets	day	-1000.	0.	50000.	Eddy_Time_Scale	voets	N/A	TZYX
vobn2	s-1	-1000.	0.	50000.	Brunt_Vaissala_Frequency	vobn2	N/A	TZYX

sorosrad	m	-1000.	0.	50000.	Rossby_Radius	sorosrad	N/A	TYX
socurl	s-1	0.	-1000.	1000.	Relative_Vorticity	socurl	N/A	TYX
sobarstf	m3/s	0.	-300.e6	300.e6	Barotropic_Stream_Function	sobarstf	N/A	TYX
sobarstfu	m3/s	0.	-300.e6	300.e6	Barotropic_Stream_Function_U	sobarstfu	N/A	TYX
sobarstfv	m3/s	0.	-300.e6	300.e6	Barotropic_Stream_Function_V	sobarstfv	N/A	TYX
somxl010	m	0.	0.	7000.	Mixed_Layer_Depth_on_0.01_rho_crit	somxl010	N/A	TYX
somxl030	m	0.	0.	7000.	Mixed_Layer_Depth_on_0.03_rho_crit	somxl030	N/A	TYX
somxlt02	m	0.	0.	7000.	Mixed_Layer_Depth_on_-0.2_temp_crit	somxlt02	N/A	TYX
iowaflup	kg/m2/s	0.	-1000.	1000.	ice_ocean_water_flux	iowaflup	N/A	TYX
sowaflcd	kg/m2/s	0.	-1000.	1000.	Concentration_dilution_water_flux	sowaflcd	N/A	TYX
zomsfglo	Sverdrup	99999	-1000.	1000.	Meridional_Overt.Cell_Global	zomsfglo	N/A	TZY
zomsfatl	Sverdrup	99999	-1000.	1000.	Meridional_Overt.Cell_Atlantic	zomsfatl	N/A	TZY
zomsfinp	Sverdrup	99999	-1000.	1000.	Meridional_Overt.Cell_IndoPacif	zomsfinp	N/A	TZY
zomsfind	Sverdrup	99999	-1000.	1000.	Meridional_Overt.Cell_Indian	zomsfind	N/A	TZY
zomsfpac	Sverdrup	99999	-1000.	1000.	Meridional_Overt.Cell_pacif	zomsfpac	N/A	TZY
EOF								

B cdfio module

FUNCTION closeout(kout)

Arguments: INTEGER, INTENT(in) :: kout. Netcdf ID of the file to be closed.

Purpose: Close an open netcdf file, specified by its ID

Example: istatus = closeout(ncout)

FUNCTION copyatt(cdvar, kidvar, kcin, kcout)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdvar !: Name of the variable
INTEGER,INTENT(in) :: kidvar !: var id of variable cdvar
INTEGER,INTENT(in) :: kcin !: ncid of the file where to read the attributes
INTEGER,INTENT(in) :: kcout !: ncid of the output file. INTEGER :: copyout !: function
return value: return an error status.

Purpose: Copy all the attributes for one variable, taking the example from another file, specified by its ncid. Return the status of the function. If $\neq 0$, indicates an error.

Example:

```
istatus = NF90\_DEF\_VAR(icout,'nav\_lon',NF90\_FLOAT,nvdim(1:2),id\_lon)
istatus = copyatt('nav\_lon',id\_lon,ncid,icout)
```

Remark: This function is used internally to cdfio, in the function create.

FUNCTION create(cdf, cdfref ,kx,ky,kz, cdep)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdf !: name of file to create
CHARACTER(LEN=*), INTENT(in) :: cdfref !: name of file used as reference for attributes
INTEGER,INTENT(in) :: kx, ky, kz !: value of the dimensions x, y and z (depth)
CHARACTER(LEN=*), OPTIONAL, INTENT(in) :: cdep !: name of depth variable if differs from cdf
INTEGER :: create !: function return value : the ncid of created variable.

Purpose: Create a netcdf file (IOIPSL type) and copy attributes for nav_lon, nav_lat, depth and time_counter from the reference file given in argument. It is supposed that the reference file is also IOIPSL compliant. For historical reason, there many different names for the depth dimension and variable. If we want to create the new data set with a depth name that differs from the reference file, the cdep optional argument can be used. The return value of the fuction is the ncid of the file just created.

Example:

```
! create output fileset
cfileout='cdfmoy.nc'
cfileout2='cdfmoy2.nc'
! create output file taking the sizes in cfile

ncout =create(cfileout, cfile,npiglo,npjglo,npk)
ncout2=create(cfileout2,cfile,npiglo,npjglo,npk)
```

or

```
! create output fileset
cfileout='w.nc'
! create output file taking the sizes in cfile

ncout =create(cfileout, cdf,cfile,npiglo,npjglo,npk,'depthw')
```

FUNCTION createvar (kout,cdvar,kvar,kpk, kidvo)

Arguments:

INTEGER, INTENT(in) :: kout !: ncid of the output file
CHARACTER(LEN=80),DIMENSION(kvar),INTENT(in) :: cdvar !: names of the variables to create
INTEGER,DIMENSION(kvar), INTENT(in) :: kpk !: number of vertical dim per variable
INTEGER, INTENT(in) :: kvar !: number of variables
INTEGER,DIMENSION(kvar), INTENT(out) :: kidvo !: varid of the created variables
INTEGER :: createvar !: function return value: error status.

Purpose: Creates the kvar variables defined by the cdvar and kpk arrays. Save the varid's in kidvo. This routine uses the att.txt file to look for the attributes of the created variables.

Example:

```
ncout =create(cfileout, cfile,npiglo,npjglo,npk)
ncout2=create(cfileout2,cfile,npiglo,npjglo,npk)

ierr= createvar(ncout , cvarname,  nvars, ipk, id_varout )
ierr= createvar(ncout2, cvarname2, nvars, ipk, id_varout2)
```

FUNCTION getatt(cdatt,cdfile,knvars,kid_var)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdatt, & ! attribute (real) name to look for
& cdfile ! file to look at
INTEGER, INTENT(in) :: knvars ! number of variables in the file
INTEGER, DIMENSION(*), INTENT(in):: kid_var ! varid of the variables
REAL(KIND=4), DIMENSION(knvars) :: getatt ! function return values: real array with
attributes.

Purpose: Return a REAL array with the values of the attribute cdatt for all the variables in
cdfile.

Example:

```
! get missing_value attribute
spval(:) = 0.
spval = getatt( 'missing_value', cfile, nvars, id_var)
```

FUNCTION getdim (cfile,cdim_name,cdtrue,kstatus)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cfile , & ! File name to look at
& cdim_name ! dimension name to look at
CHARACTER(LEN=80),OPTIONAL, INTENT(out) :: cdtrue ! full name of the read dimension
INTEGER, OPTIONAL, INTENT(out) :: kstatus ! status of the nf inquire
INTEGER :: getdim ! the value for dim cdim_name, in file cfile

Purpose: Return the INTEGER value of the dimension identified with cdim_name in cfile

Example:

```
npiglo= getdim (cfile,'x')
npjglo= getdim (cfile,'y')
npk    = getdim (cfile,'depth',kstatus=istatus)
....
idum=getdim(cdfilref,'depth',cldep)  ! return in cldep the name of the dim
                                   ! whose 'depth' is used as proxy
```

FUNCTION getvdim (cdfile,cdvar)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile ! File name to look at
CHARACTER(LEN=*), INTENT(inout) :: cdvar ! variable name to look at.
INTEGER :: getvdim ! number of dim for cdvar

Purpose: Return the number of dimension for variable cdvar in cdfile.

If *cdvar* is not found in *cdfile*, then a list a available variables is displayed and the user is asked to choose the required one. In this case, *cdvar* is updated to the choosen variable name, and is made available to the calling program.

This function is intended to be used with prognostic variables of the model, which are defined in the file either as [TZXY] (3D variable) or as [TXY] (2D variable). The time dimension is not considered. Erroneous results are produced if the variables is [ZXY] or [XY].

Example:

```
...  
cvar='variablex'  
nvdim = getvdim(cfilev,cvar)  
IF (nvdim == 2 ) nvpk = 1    ! 2D variable ==> 1 level  
IF (nvdim == 3 ) nvpk = npk ! 3D variable ==> npk levels  
PRINT *, TRIM(cvar), ' has ', nvdim, ' dimensions  
...
```

FUNCTION getipk (cdfile,knvars)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile ! File to look at
INTEGER, INTENT(in) :: knvars ! Number of variables in cdfile
INTEGER, DIMENSION(knvars) :: getipk ! array (variables) of levels

Purpose: Return the number of levels for all the variables in cdfile. Return 0 if the variable in a vector.

Example:

```
! ipk gives the number of level or 0 if not a T[Z]YX variable  
ipk(:)      = getipk (cfile,nvars)
```

FUNCTION getnvar (cdfile)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile ! file to look at
INTEGER :: getnvar ! return the number of variables

Purpose: Return the number of variables in cdfile

Example:

```
nvars = getnvar(cdfile)
PRINT *, ' nvars =', nvars
```


FUNCTION getvar (cdfile,cdvar,klev,kpi,kpj,kimin,kjmin)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kpi,kpj ! horizontal size of the 2D variable
INTEGER, OPTIONAL, INTENT(in) :: klev ! Optional variable. If missing 1 is assumed
INTEGER, OPTIONAL, INTENT(in) :: kimin,kjmin ! Optional : set initial point to get
REAL(KIND=4), DIMENSION(kpi,kpj) :: getvar ! 2D REAL 4 holding variable field at klev

Purpose: Return the 2D REAL variable cdvar, from cdfile at level klev.

kpi,kpj are the horizontal size of the 2D variable

Example:

```
v2d(:,:)= getvar(cfile, cvarname(jvar), jk ,npiglo, npjglo )
```

FUNCTION getvarxz (cdfile,cdvar,kj,kpi,kpz,kimin,kkmin)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kpi,kpz ! size of the 2D variable
INTEGER, INTENT(in) :: kj ! Optional variable. If missing 1 is assumed
INTEGER, OPTIONAL, INTENT(in) :: kimin,kkmin ! Optional set initial point to get
REAL(KIND=4), DIMENSION(kpi,kpz) :: getvarxz ! 2D REAL 4 holding variable x-z slab
at kj

Purpose: Return the 2D REAL variable x-z slab cvar, from cdfile at j=kj
kpi,kpz are the size of the 2D variable

Example:

```
v2d(:,:)= getvarxz(cfile, cvarname(jvar), jj ,npiglo,npk, imin, kmin )
```

FUNCTION getvaryz (cdfile,cdvar,ki,kpj,kpz,kjmin,kkmin)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kpj,kpz ! size of the 2D variable
INTEGER, INTENT(in) :: ki ! Optional variable. If missing 1 is assumed
INTEGER, OPTIONAL, INTENT(in) :: kjmin,kkmin ! Optional set initial point to get
REAL(KIND=4), DIMENSION(kpj,kpz) :: getvaryz ! 2D REAL 4 holding variable x-z slab
at kj

Purpose: Return the 2D REAL variable y-z slab cvar, from cdfile at i=ki
kpj,kpz are the size of the 2D variable

Example:

```
v2d(:,:)= getvaryz(cfile, cvarname(jvar), ji ,npjglo,npk,jmin,kmin )
```

FUNCTION getvar1d (cdfile,cdvar,kk,kstatus)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kk ! size of 1D vector to be returned
INTEGER, OPTIONAL, INTENT(out) :: kstatus ! return status concerning the variable existence
REAL(KIND=4), DIMENSION(kk) :: getvar1d ! real returned vector

Purpose: Return 1D variable cdvar from cdfile, of size kk

Example:

```
tim=getvar1d(cfile,'time_counter',1)
....
z1d=getvar1d(cdfile,'deptht',kpk,idept)
IF ( idept /= NF90_NOERR ) THEN
  z1d=getvar1d(cdfile,'depthu',kpk,idepu)
  IF ( idepu /= NF90_NOERR ) THEN
    z1d=getvar1d(cdfile,'depthv',kpk,idepv)
    IF ( idepv /= NF90_NOERR ) THEN
      z1d=getvar1d(cdfile,'depthw',kpk,idepw)
      IF ( idepw /= NF90_NOERR ) THEN
        PRINT *, ' No depth variable found in ', TRIM(cdfile)
        STOP
      ENDIF
    ENDIF
  ENDIF
ENDIF
ENDIF
```

This last example shows how to use the optional argument kstatus in order to figure out which is the real name of the depth variable.

FUNCTION getvare3 (cdfile,cdvar,kk)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kk ! size of 1D vector to be returned
REAL(KIND=4), DIMENSION(kk) :: getvare3 ! return e3 variable form the coordinate file

Purpose: Special routine for e3, which in fact is a 1D variable but defined as e3 (1,1,npk,1) in coordinates.nc (!!)

Example:

```
gdepw(:) = getvare3(coordzgr, 'gdepw',npk)
e3t(:)    = getvare3(coordzgr, 'e3t', npk )
```

FUNCTION getvarname (cdfile, knvars)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile ! name of file to work with
INTEGER, INTENT(in) :: knvars ! Number of variables in cdfile
CHARACTER(LEN=80), DIMENSION(knvars) :: getvarname ! return an array with the
names of the variables

Purpose: Return a character array with the knvars variable name corresponding to cdfile

Example:

```
cvarname(:)=getvarname(cfile,nvars)
```

FUNCTION putatt (cdvar,kout,kid)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdvar ! name of variable
INTEGER, INTENT(in) :: kout ! ncid of the output file
INTEGER, INTENT(in) :: kid ! variable id
INTEGER :: putatt ! return variable : error code.

Purpose: Scan file att.txt for finding the line corresponding to cdvar, then read the attributes for this variables, whose id is kid and write them in file id kout.

Example:

```
! add attributes  
istatus = putatt(cdvar(jv), kout,kidvo(jv))
```

Remark: This is almost an internal routine called by createvar.

FUNCTION putheadervar(kout, cdfile, kpi,kpj,kpk,pnavlon, pnavlat,pdep)

Arguments:

INTEGER, INTENT(in) :: kout ! ncid of the outputfile (already open)
CHARACTER(LEN=*), INTENT(in) :: cdfile ! file from where the headers will be copied
INTEGER, INTENT(in) :: kpi,kpj,kpk ! dimension of nav_lon,nav_lat (kpi,kpj), and depht(kpk)
REAL(KIND=4), OPTIONAL, DIMENSION(kpi,kpj) :: pnavlon, pnavlat ! to get rid of nav_lon , nav_lat of cdfile
REAL(KIND=4), OPTIONAL,DIMENSION(kpk), INTENT(in) :: pdep ! dep array if not on cdfile
INTEGER :: putheadervar ! return status

Purpose: Copy header variables from cdfile to the already open ncfile (ncid=kout)

If the 2 first optional arguments are given, they are taken for nav_lon and nav_lat, instead of those read in file cdfile. This is usefull for f-points results whne no basic "gridF" files exist. If the third optional argument is given, it is taken as the depht(:) array in place of the the depth read in cdfile. If all 3 optional arguments are used, cdfile will not be used and a dummy argument can be passed to the function instead.

Example:

```
ierr= putheadervar(ncout , cfile, npiglo, npjglo, npk)
ierr= putheadervar(ncout2, cfile, npiglo, npjglo, npk)
```

or

```
ierr= putheadervar(ncout , cfile, npiglo, npjglo, npk, glamf, gphif )
```

or

```
ierr= putheadervar(ncout , 'dummy', npiglo, npjglo, npk, glamt, gphit, gdepw )
```


FUNCTION putvar(kout, kid,ptab, klev, kpi, kpj)

Arguments:

INTEGER, INTENT(in) :: kout , & ! ncid of output file
& kid ! varid of output variable
REAL(KIND=4), DIMENSION(kpi,kpj),INTENT(in) :: ptab ! 2D array to write in file
INTEGER, INTENT(in) :: klev ! level at which ptab will be written
INTEGER, INTENT(in) :: kpi,kpj ! dimension of ptab
INTEGER :: putvar ! return status

Purpose: copy a 2D level of ptab in already open file kout, using variable kid

Example:

```
ierr = putvar(ncout, id_varout(jvar) ,rmean, jk, npiglo, npjglo)
```

FUNCTION putvar1d(kout,ptab,kk,cdtype)

Arguments:

INTEGER, INTENT(in) :: kout ! ncid of output file
REAL(KIND=4), DIMENSION(kk),INTENT(in) :: ptab ! 1D array to write in file
INTEGER, INTENT(in) :: kk ! number of elements in ptab
CHARACTER(LEN=1), INTENT(in) :: cdtype ! either T or D (for time or depth)
INTEGER :: putvar1d ! return status

Purpose: Copy 1D variable (size kk) hold in ptab, with id kid, into file id kout

Example:

```
                ierr=putvar1d(ncout,timean,1,'T')  
                ierr=putvar1d(ncout2,timean,1,'T')  
...  
                istatus = putvar1d(kout,depw(:),kpk,'D')
```

SUBROUTINE ERR_HDL(kstatus)

Arguments:

INTEGER, INTENT(in) :: kstatus

Purpose: Error handler for NetCDF routine. Stop if kstatus indicates error conditions. Else indicate the error message.

Example:

```
CALL ERR_HDL(istatus)
```

C eos module

FUNCTION sigma0 (ptem, psal, kpi,kpj)

Arguments:

REAL(KIND=4), DIMENSION(kpi,kpj), INTENT(in) :: ptem, psal ! Temperature and Salinity arrays
INTEGER,INTENT(in) :: kpi,kpj !: dimension of 2D arrays
REAL(KIND=8), DIMENSION(kpi,kpj) :: sigma0 ! Potential density

Purpose: Compute the potential volumic mass (Kg/m3) from potential temperature and salinity fields

Example:

FUNCTION sigmai(ptem, psal, pref, kpi,kpj)

Arguments:

REAL(KIND=4), DIMENSION(kpi,kpj), INTENT(in) :: ptem, psal ! Temperature and Salinity arrays
REAL(KIND=4), INTENT(in) :: pref !: reference pressure (dbar)
INTEGER,INTENT(in) :: kpi,kpj !: dimension of 2D arrays
REAL(KIND=8), DIMENSION(kpi,kpj) :: sigmai ! Potential density a level pref

Purpose: Compute the potential volumic mass (Kg/m3) from potential temperature and salinity fields at reference level specified by *pref*.

Example:

FUNCTION eosbn2 (ptem, psal, pdep,pe3w, kpi,kpj,kup,kdown)

Arguments:

REAL(KIND=4), DIMENSION(kpi,kpj,2), INTENT(in) :: ptem, psal ! temperature and salinity arrays
! (2 levels, only)
REAL(KIND=4) :: pdep ! depthw (W points)
REAL(KIND=4), DIMENSION(kpi,kpj), INTENT(in) :: pe3w ! vertical scale factor at W points
INTEGER, INTENT(in) :: kpi,kpj ! horizontal size of the grid
INTEGER, INTENT(in) :: kup,kdown ! index of upper and lower layer
! for the actual level
REAL(KIND=4), DIMENSION(kpi,kpj) :: eosbn2 ! result interpolated at T levels

Purpose: Compute the local Brunt-Vaisala frequency

Example:

```
DO jk = npk-1, 2, -1
  PRINT *, 'level ', jk
  zmask(:, :) = 1.
  ztemp(:, :, iup) = getvar(cfilet, 'votemper', jk-1, npiglo, npjglo)
  WHERE(ztemp(:, :, idown) == 0) zmask = 0
  zsal(:, :, iup) = getvar(cfilet, 'vosaline', jk-1, npiglo, npjglo)

  gdepw(:, :) = getvar(coordzgr, 'gdepw', jk, 1, 1)
  e3w(:, :) = getvar(coordzgr, 'e3w_ps', jk, 1, 1)

  zwk(:, :, iup) = eosbn2 ( ztemp, zsal, gdepw(1,1), e3w, npiglo, npjglo, &
    iup, idown) * zmask(:, :)
  ! now put zn2 at T level (k)
  WHERE ( zwk(:, :, idown) == 0 )
    zn2(:, :) = zwk(:, :, iup)
  ELSEWHERE
    zn2(:, :) = 0.5 * ( zwk(:, :, iup) + zwk(:, :, idown) ) * zmask(:, :)
  END WHERE

  ierr = putvar(ncout, id_varout(1), zn2, jk, npiglo, npjglo)
  itmp = idown ; idown = iup ; iup = itmp

END DO ! loop to next level
```

Contents

1	Statistics	2
	cdfmoy	2
	cdfmoy_sp	3
	cdfmean	4
	cdfheatc	5
	cdfmxlheatc	6
	cdfzonalmean	7
	cdfzonalsum	8
	cdfzonalout	9
	cdfvT	10
	cdfeke	11
	cdfrmsssh	12
	cdfstdevw	13
2	Transports	14
	cdfmhst	14
	cdfmhst-full	15
	cdfhflx	16
	cdfvhst	17
	cdfvhst-full	18
	cdfpsi	19
	cdfpsi-full	20
	cdfmoc	21
	cdfmocsig	22
	cdfmoc-full	23
	cdfmocatl	24
	cdftransportiz	25
	cdftransportiz-full	26
	cdmmasstrp	27
	cdfsigtrp	28
3	Derived quantities	29
	cdfsig0	29
	cdfbottomsig0	30
	cdfbottom	31
	cdfrhoproj	32
	cdfbn2	33
	cdfets	34
	cdfeurl	35
	cdfpv	36
	cdfw	37
	cdfmxl	38
	cdficediags	39
	cdfcensus	40

4	Extracting and information tools	41
	cdfprofile	41
	cdfwhereij	41
	cdffindij	42
	cdfmaxmoc	43
	cdfzoom	44
	cdfmax	45
	APPENDICES	46
A	att.txt file	46
B	cdfio module	48
	closeout	48
	copyatt	49
	create	50
	createvar	51
	getatt	52
	getdim	53
	getvdim	54
	getipk	55
	getnvar	56
	getvar	57
	getvarxz	58
	getvaryz	59
	getvar1d	60
	getvare3	61
	getvarname	62
	putatt	63
	putheadervar	64
	putvar	65
	putvar1d	66
	ERR_HDL	67
C	eos module	68
	sigma0	68
	sigmai	68
	eosbn2	69