

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

## Part I : User Guide

J.M. Molines \*

Last update: Rev: 14 Date: 2007-04-17 18:30:39 +0200 (Tue, 17 Apr 2007)

## 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, cdf variable names etc... 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. Developpers may read part II of this manual which is the programmer guide, where functions and subroutines included in these modules are described.

## What is changed since the initial version?

Release 2.1 for CDFTOOLS differs from 2.0, mainly by the elimination of the file `att.txt` which was used to describe variables attributes and that had become difficult to maintain (too big). Now the attributes are defined in a structure in each program. For the end user it is transparent. For the developer, this induces some changes in the `getvarxxx` series from `cdfio`. Also an effort have been made in this version to adapt the code for input files with multiple time-frame. This transition is realized when the necessity show up. If you wish to have this possibility for some tools, please ask a CDFTOOLS developer.

## Recently added tools

**cdfbathy** : A tool to modify and tune nemo bathymetry.

**cdfstd** : Compute the standard deviation of variables from a series of files given as input.

---

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

**cdfmoy\_annual** : Compute an annual mean from monthly means, applying weights to take into account that monthly mean were not computed with the same amount of data

**cdfclip** : Save functionality than nekss but does not change the order of the variables.

**cdfconvert** : Convert a set of dimg CLIPPER output file to DRAKKAR like netcdf file.

**cdfplxconv** : Convert a set of fluxes dimgfile (Clipper like) to a set of CDF files (Drakkar like )

**cdf16bit** : Convert a standard 32bits model output file into a 16 bit file, using scale\_factor and add\_offset

**cdfvita** : Compute surface ocean velocities components and module on the A-Grid (at T-points) from a opa C-grid input.

**cdfmeanvar** : Compute spatial 3D mean as well as corresponding variance.

**cdfstddevts** : Compute RMS for Temperature and Salinity. (goes together with cdfmoy\_sal2.temp2).

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

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

**cdfmxlsaltc** : Compute the Salt Content of the mixed layer (kg/m2)

**cdfsigtrp** : Compute density class transport across a section.

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

**cdfmocsig** : Compute the MOC as a function of potential density ( $\sigma_1$ )

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

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

# 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:** none.

**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 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:** none

**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*.

### cdfmoy\_sal2\_temp2:

**Purpose:** As cdfmoy above: Compute the mean fields for the file list given in argument but additional quadratic means for temperature and salinity are kept in the output file, if they appear in the input file.

**Usage:** *cdfmoy\_sal2\_temp2 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:** none.

**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, votemper, vosaline, vozocrtx, vomecrtx, vovecrtz are saved.

**Remark:** Assumes that land value have are set to 0. If you have used this tools, it is likely that you want to compute temperature and salinity variability. See **cdfstdts** for this purpose.

**Associated script:** You can adapt *cdfmoy.ll* and other for this case. Its almost the same. Only the content of the *cdfmoy2.nc* will differ.

### cdfmoy\_annual:

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

**Usage:** *cdfmoy\_annual 12 monthly mean files (DRAKKAR like)*

**Input:** A list of homogeneous 12 monthly means.

**Required mesh\_mask files or other files:** none.

**Output:** 1 files are produced : *cdfmoy\_annual.nc*

**Remark:** Assumes that land value are set to 0. This program is usefull for calculating the annual mean from the monthly means calculated in DRAKKAR project. In this case, the weight applied to the months are 6 5 7 6 6 6 6 6 7. Anne-Marie claims that this may change the computation of water mass balances.

**Associated script:** none

### **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).

### **cdfmean-full:**

**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. This is the -full version (full steps) of cdfmean.

**Usage:** *cdfmean-full 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-full ORCA025-G42.y0010-ANNUAL.gridT.nc votemper T* will compute the mean temperature over the whole domain.

*cdfmean-full 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-full 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).

### **cdfmeanvar:**

**Purpose:** This program is very similar to the previous in the list: It computes the mean value of a field, and its spatial variance, weighted by the local metric. If the variable is a 3D variable, the mean value and variance are given for each level, then the global mean/variance are printed. The mean/variance values can be computed on a limited domain, specified on the command line.

**Usage:** `cdfmeanvar 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: `cdfmeanvar ORCA025-G42-y0010-ANNUAL_gridT.nc votemper T` will compute the mean temperature over the whole domain.

`cdfmeanvar 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...).

`cdfmeanvar 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).

### **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-ful:**

**Purpose:** Compute the heat content for the ocean in the mixed layer read in the gridT file, FULL STEP case.

**Usage:** *cdfmxlheatc-full gridTfile*

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

**Required mesh\_mask files or other files:** 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

### **cdfmxlsaltc:**

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

**Usage:** *cdfmxlsaltc gridTfile*

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

For instance: `cdfmxlsaltc ORCA025-G42_y0010_m03d15.gridT.nc` will compute the salt content in the mixed layer for this file.

**Required mesh\_mask files or other files:** mesh\_zgr.nc, mask.nc must be in the current directory.

**Output:** Results are written in the netcdf file mxlsaltc.nc, in the variable *somxlsaltc*. Units are kg/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:** *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*.

**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:** *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*.

**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:** none

**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-transports (UT and US) are computed at the C-grid U-point, and that the V-transports (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:** none

**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 cdfmoy. 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:** none

**Output:** *rmsssh.nc*. This file hold the variable *sssheig\_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:** none

**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.

**cdfstdevts:**

**Purpose:** Compute the standard deviation for temperature and salinity

**Usage:** *cdfstdevts gridX gridX2*

**Input:** gridX and gridX2 hold respectively the mean and quadratic mean for T-points. These files are produced by *cdfmoy\_sal2\_temp2*. 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: *cdfstdevts ORCA035-G32\_y0008-0010\_gridT.nc ORCA035-G32\_y0008-0010\_gridT2.nc* will compute the standard deviation of T and D for the period y0008-0010.

**Required mesh\_mask files or other files:** none

**Output:** *rmsts.nc*. This file hold the variables *votemper\_rms* and *vosaline\_rms*

**Remark:** This quantity is not a very standard one. For this reason, we keep the standard *cdfmoy*, without saving the second order momentum for T and S and create a special tool *cdfmoy\_sal2\_temp2* for this purpose. Remember that both T2 and S2 are 3D fields...

**Associated script:** none.

**cdfstd:**

**Purpose:** Compute the standard deviation for all the physical variables of the serie of files given as input

**Usage:** *cdfstd list of files*

**Input:** The input files are model output files, all holding the same variables. Each file can have more than 1 time frame in it.

**Required mesh\_mask files or other files:** none

**Output:** *cdfstd.nc*. This file hold the standard deviation of the variables,whose name will be *varname\_std*

**Associated script:** none.

**Contributor:** Frederic Castruccio (Meom)

## 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 useful 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 usefull 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

**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

**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*

**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 (jpiglo, jpjglo) 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*

**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 (jpiglo, jpjglo) 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*.

### **cdfpsi-open:**

**Purpose:** Compute the Barotropic Stream Function from an open domain output.

**Usage:** *cdfpsi-open Ufile Vfile [-mask] [-moy]*

**Input:** Ufile and Vfile are the files holding vozocrtx and vomecrtx.

If -mask option is used, resulting sobarstf field is masked, else it is not masked.

If -moy option is used, the resulting field is the mean value between Ucomputation and Vcomputation.

**Required mesh\_mask files or other files:** mesh\_hgr.nc, mesh\_zgr.nc, mask.nc

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

**Remark:** It is very likely, that this program should be edited by the end user to fit its own configuration. In particular, in the standard version, the upper left corner is arbitrarily set to 0. When editing, take care of the sign for integration! When applied to an extruded domain from a larger model, this program will not give exactly the same BSF, because of a different starting reference. A modified version suitable for SALOMON025 configuration is available in cdfpsi-open\_AM.f90.

### **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* (if *new\_maskglo.nc* does'nt 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  $\sigma_1$  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`

**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:** *cdfmocsig.ll*.

### cdfmocsig-full:

**Purpose:** Compute the Meridional Overturning Circulation (partial step case) in function of  $\sigma_1$  from the meridional velocity, a TS file and a basin mask file. This is the full step version.

**Usage:** *cdfmocsig-full Vfile TSfile*

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

For instance: `cdfmocsig-full ORCA025-G32_y0010_ANNUAL_gridV.nc ORCA025-G32_y0010_ANNUAL_gri`  
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`

**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 —)

**!!!CAUTION !!!:** THIS PROGRAM IS NOT FINISHED YET. DONT USE WITH FULL\_STEPS!

**Associated script:** none

### **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`

**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`

**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\_trp.dat and the standard output (for interactivity).

**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\_trp.dat and the standard output for interactivity.

**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*.

#### **cdftransportiz\_noheat:**

**Purpose:** same as cdftransportiz but only for mass transport. Usefull when VT files are not available.

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

**Output:** section\_trp.dat and the standard output for interactivity.

### **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:** gridU 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\_trp.dat and standard output. .dat file is matlab ready.

**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. This tool is about the same than cdftransportiz\_noheat, and should be used in place of. The only difference is in that in this program there is no dummy heat/salt transport output.

### **cdfmasstrp-full:**

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

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

**Input:** gridU and gridV files are simultaneous velocity component files. For more details, read the documentation for cdfmasstrp.

**Required mesh\_mask files or other files:** mesh\_hgr.nc, mesh\_zgr.nc

**Output:** section\_trp.dat and standard output. .dat file is matlab ready.

## 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  $\sigma_0$ ,

*nbins* is the number of bins (density classes) to explore between  $\sigma_{min}$  and  $\sigma_{max}$ .

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, $\sigma_0$ ,U (2) (x/y,  $\sigma$ ) 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:** This program outputs its results on 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.

## cdfsigtrp-full:

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

**Usage:** *cdfsigtrp-full 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  $\sigma_0$ ,

*nbins* is the number of bins (density classes) to explore between  $\sigma_{min}$  and  $\sigma_{max}$ .

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-full 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 bimg files are produced per section. (1) (x/y,depth) for T,S, $\sigma_0$ ,U (2) (x/y,  $\sigma$ ) 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:** This program outputs its results on an ASCII file called **trpsig.txt** which is multicolumns file with the first column giving the density, and other colums 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
.../
```

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.

### cdftemptrp-full:

**Purpose:** Compute the transport between isotherms.

**Usage:** *cdftemptrp-full gridTfile gridUfile gridVfile temp\_max temp\_min nbins [options]*

**Input:** The syntax is almost the same than for cdfsigrp. Working with temperatures instead of density.

**Required mesh\_mask files or other files:** mesh\_hgr.nc, mesh\_zgr.nc must be in the current directory.

**Output:** Ascii file trptemp.txt.

**Remark:** Contribution of Fred Castruccio.

### cdftempvol-full:

**Purpose:** Compute water volume in a given domain between isotherms. FULL STEPS version

**Usage:** *cdftempvol-full gridTfile imin, imax, jmin, jmax temp\_max temp\_min nbins [options]*

**Input:** The syntax is almost the same than for cdfsigrp. Working with temperatures instead of density.

**Required mesh\_mask files or other files:** mesh\_hgr.nc, mesh\_zgr.nc must be in the current directory.

**Output:** Ascii file voltemp.txt.

**Remark:** Contribution of Fred Castruccio.

### 3 Derived quantities

#### cdfsig0:

**Purpose:** Compute the potential density  $\sigma_0$ .

**Usage:** *cdfsig0 gridT*

**Input:** gridT is a file holding variables votemper and vosaline. For instance: cdfsig0 ORCA035-G32\_y0008m01d10\_gridT.nc will compute  $\sigma_0$  for the given tag.

**Required mesh\_mask files or other files:** none

**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  $\sigma_0$  computed from mean temperature and salinity fields is not the same than the mean  $\sigma_0$  computed from instantaneous temperature and salinity fields.

**Associated script:** *cdfsigma0.ll*. This script can be used to compute  $\sigma_0$  for a run. It scan all the gridT file for a given year and CONFIG, and produce the corresponding sigma0 file.

#### cdfsigi:

**Purpose:** Compute the potential density  $\sigma_i$ , refered to a particular depth.

**Usage:** *cdfsigi gridT Reference\_depth*

**Input:** gridT is a file holding variables votemper and vosaline.  
Reference\_depth is the reference depth in meters. For instance:  
cdfsigi ORCA035-G32\_y0008m01d10\_gridT.nc 2000  
will compute  $\sigma_i$  for the given tag, refered to 2000 m, *ie*,  $\sigma_2$ .

**Required mesh\_mask files or other files:** none

**Output:** *sigi.nc*. This file hold the variable vosigmai, reference depth is documented on the 'long name' attriute.

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

### **cdfbottomsig0:**

**Purpose:** Compute the bottom potential density  $\sigma_{0bot}$  (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  $\sigma_{0bot}$  for the given tag.

**Required mesh\_mask files or other files:** none

**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  $\sigma_0$  computed from mean temperature and salinity fields is not the same than the mean  $\sigma_0$  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  $\sigma_{0bot}$  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:** 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  $\sigma_0$  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  $\sigma_0$  for the iso surface is given on the command line, then only this surface is calculated. In other cases, the values for  $\sigma_0$  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  $\sigma_0$  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  $\sigma_0$  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:** 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

**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 1<sup>st</sup> rossby radius for the given tag.

**Required mesh\_mask files or other files:** mesh\_zgr.nc, mesh\_hgr.nc

**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  $\leq 0$ , 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  $\leq 0$  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:** 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  $\rho_0$  takes the constant value of  $1020 \text{ kg/m}^3$ ,  $f$  is the coriolis parameter ( $2\Omega \sin(\phi)$ ),  $\zeta$  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  $\frac{\partial \sigma_0}{\partial z}$ .  $\zeta$  is computed at f-points, u-v level. Therefore, in order to estimate  $\zeta$  at W points, a 4-point horizontal average is required (to get  $\zeta$  at T-points) followed by a 2-point vertical average to get it at W-points.

**Required mesh\_mask files or other files:** *mesh\_hgr.nc*, *mesh\_zgr*

**Output:** *pv.nc*. The variable name is *vopv*. Units are  $s^{-1}m^{-1} \times 10^{11}$

**Associated script:** none

### cdfpvor:

**Purpose:** Compute the different component of the Ertel potential velocity.

**Usage:** *cdfpvor 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: *cdfpvor ORCA025-G70\_y2000m01d05\_gridT.nc ORCA025-G70\_y2000m01d05\_gridU.nc ORCA025-G70\_y2000m01d05\_gridV.nc* will compute the potential vorticity.

**Method:** The total potential vorticity is evaluated as the sum of the relative vorticity and the stretching.

$$PV = (f + \zeta_t) \frac{\partial \sigma_0}{\partial z}$$

$$f = 2\Omega \sin(\phi_t \pi / 180.)$$

$$\zeta_f = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}$$

and  $\zeta_t$  is the mean value of the 4 corners of the grid cell.

$\frac{\partial \sigma_0}{\partial z}$  is deduced from the Brunt-Vaissala frequency, using the code equation of state (McDougall, 1987).

**Required mesh\_mask files or other files:** *mesh\_hgr.nc*, *mesh\_zgr*

**Output:** *pvor.nc*. The variables names are *vorelvor*, *vostrvor* and *vototvor*. Units are  $kg \cdot m^{-4} \cdot s^{-1} \times 10^7$ .

**Associated script:** none

**Remark:** This tools is provided by Anne-Marie Treguier. It computes almost the same thing than *cdfpv*, but using different algorithm.

**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 vomecrty. 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:** 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  $|T_{crit}| = 0.2^\circ\text{C}$ . PARTIAL STEP version.

**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:** *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.

### **cdfmxl-full:**

**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  $|T_{crit}| = 0.2^\circ\text{C}$ . FULL STEP version

**Usage:** *cdfmxl-full 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-G03-y0008m01d10-gridT.nc* will compute the mixed layer depth based on the 3 criteria.

**Required mesh\_mask files or other files:** *mesh\_hgr.nc*, *mesh\_zgr.nc*

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

**Associated script:** *cdfmxl-full.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 <sup>9</sup> m3)	35268.1868656639999
NArea (10 <sup>9</sup> m2)	13113.4348328960004
NExtend (10 <sup>9</sup> m2)	13062.9962301439991

#### Southern Hemisphere

SVolume (10 <sup>9</sup> m3)	4879.33240934399964
SArea (10 <sup>9</sup> m2)	3477.76693043200021
SExtend (10 <sup>9</sup> 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] [-bing]*

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,  $\sigma_0(S,T)$ ,  $\sigma_1(S,T)$ ,  $\sigma_4(S,T)$ , are given computed from the EOS. (Plotting purposes).

nlog is an integer number  $\geq 0$  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
```

If option -bing specified, a bimg file is output instead of a netcdf file.

**Input:** ncdf file for gridT

**Required mesh\_mask files or other files:** mesh\_hgr.nc, mesh\_zgr.nc

**Output:** Output is done on census.nc (variables volcensus,sigma0,sigma2,sigma4). The unit of volcensus is somewhat tricky, depending on the number of log rescaling that where used. A bimg output file is also available if option -bing given.

**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

### **cdfprobe:**

**Purpose:** Display a 2 columns output time(d) value

**Usage:** *cdfprobe cdf\_file I J cdfvar [level]*

**Input:** cdf\_file = name of the file

I J : position where to look at cdfvar : name of the cdf variable

level : (optional) : Level where to look at example: cdfprobe u10.nc 300 350 u10

example: cdfprobe alltag\_gridT.nc 240 234 votemper 30

**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

### **cdfclip:**

**Purpose:** Extract a subzone of a netcdf file.

**Usage:** *cdfclip -f file -zoom imin imax jmin jmax*

**Input:** file is the netcdf file to clip

imin imax jmin jmax are the limit in (I,J) space to clip.

For instance: `cdfclip -f coordinates.nc -zoom 100 300 250 400`

**Required mesh\_mask files or other files:** none

**Output:** output is done on `cdfclip.nc` with same variables name as original.

**Remark:** Still to be done : limit to specified variables, clip on the vertical.

**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

## cdfbathy:

**Purpose:** Modify a bathymetric file for NEMO, in many different ways.

**Usage:** `cdfbathy -f bathy_file -zoom imin imax jmin jmax -fillzone -fullstep depmin -replace 'file'`  
`-dumpzone 'file' -a -o`  
-file (or -f ) : name of bathy file  
-zoom (or -z ) : sub area of the bathy file to work with (imin imax jmin jmax)  
-fillzone (or -fz ) : sub area will be filled with 0 up to the first coast line  
-raz\_zone (or -raz ) : sub area will be filled with 0 up  
-fullstep (or -fs ) : sub area will be reshaped as full-step, below depmin  
requires the presence of the file `zgr.bat.txt` (from `ocean.output`, eg )  
-dumpzone (or -d ) : sub area will be output to an ascii file, which can be used by -replace  
after manual editing  
-nicedumpzone (or -nd ) : sub area will be output to an ascii file (nice output)  
-replace (or -r ) : sub area defined by the file will replace the original bathy  
-append (or -a ) : fortran log file (`log.f90`) will be append with actual modif  
Standard behaviour is to overwrite/create log file  
-overwrite (or -o ) : input bathy file will be used as output.  
Standard behaviour is to use a work copy of the original file  
(indexed from 01 to 99 if necessary )

**Input:** This program allows a short syntax for option, and a longer, more mnemonic.

Basically, this program works on a copy of the input file, indexed from 01 to 99 if necessary. You can work directly on the input file with the option -overwrite ( or -o), but take care that original data will be modified.

The second important point is that all actions specified through the options apply to the sub-area indicated by the zoom option, given in the (I,J) space.

Specific actions are then controled by the options :

**information** : just create an ascii file with a formatted copy of the sub-area

-dumpzone (-d) 'file.txt'

-nicedumpzone (-nd) 'file.txt', same as -d but with a different format.

**modification** : For these kind of action, a log file (`log.f90`) is maintained. It records the different modification in a fortran 90 file, that can be used afterward to replace the original bathy. Changes can be append to the log file if -append (-a) option is given. The possible modifications are :

-fillzone (-fz) : Fill a subarea between the edge of the domain and the 1st coast line point. Usefull to fill the Pacific, for instance, in a NATL4 configuration, extracted from ORCA025.

-raz\_zone (-raz) : the sub-area is set to 0.

-replace (-r) 'file.txt' : replace the 'patch' of bathymetry given by file.txt in the input file. The file.txt is typically created from a previous call with -dumpzone option, and manually edited to fix some details of the topography.

-fullstep (-fs) : the bathy defined in the sub-area will be z-step like bathymetry. This option requires, an ASCII file describing the vertical discretization of the model. This file ( `zgr.bat.txt` ) is made from a copy of `ocean.output`

**Required mesh\_mask files or other files:** `zgr.bat.txt` for -fs option. ( Should be replace by `mesh_zgr.nc`)

**Output:** Output file is 'input\_file'.nn where nn is a 2 digit integer incremented as necessary not to overwrite existing file.

**Associated script:** none

**Remark:** This program was written for helping the user to tune a bathymetric file. Its dump/replace capability as well as the log file maintenance is very much appreciated. It can be improved or extended with new options, if necessary. However, it is not intended to replace the OPABAT package, which well performs initial interpolation and filtering.

**Example:** 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] [-xy]*

**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.

If -xy option is used, cdfmax is forced to work on an horizontal slab, whatever the limits are.

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

**Remark:** In the CDFTOOLS directory, the attentive reader will find a *cdfmax-test* version of this program. It is a beta version where the input file may have more than one time step in it. It requires modification in cdfio, and it is not working in the standard distribution. Wait for next one !

### **cdfmax\_sp:**

**Purpose:** Display min/max of a variable in a file, and their respective location. A sub area can be specified either horizontally or vertically. This is very similar to cdfmax, except that it takes into account the 'missing\_value' attribute for variables.

**Usage:** *cdfmax\_sp -f ncfile [-var cdfvarname] [-zoom imin imax jmin jmax] [-lev kmin kmax] [-fact scale\_factor] [-xy]*

**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.

If -xy is specified, the program is forced to work on an horizontal slab, even if 1 direction is degenerated.

**Required mesh\_mask files or other files:** none

**Output:** output is done on standard output

**Associated script:** none

**Remark:** In the CDFTOOLS directory, the attentive reader will find a *cdfmax-test* version of this program. It is a beta version where the input file may have more than one time step in it. It requires modification in cdfio, and it is not working in the standard distribution. Wait for next one !

## **cdf16bit:**

**Purpose:** Convert a 32 bit (real\*4, or float) model output file into a 16 bit (integer\*2 or short) outputfile. The program scans the variables of the file given as input, and if the variables name's is within a pre-defined list (see below), then a scale\_factor and an add\_offset values are determined to re-map the float onto a short. Used scale\_factor and add\_offset are written to the file as an attribute of the given variable. These attributes are now recognized by many Netcdf tools and if they exist in the file they are used as soon as the file is read. Default values are respectively 1 and 0.

Additional capability is provided for variables with a great range of values, for which the scaling results in a big loss in precision: The log10 of the field is taken before the scaling. This works only for positive values. The save\_log10 attribute is associated to this capability. It can take the value 0 (default, no log10 taken) or 1 ( log10 transform before scaling).

**Usage:** *cdf16bit 32bit\_ncfile [-check] [-verbose]*

**Input:** 32bit\_ncfile is the data file.

-check : This option enables a checking of the scaling : a warning is emitted if the scaling results in an overflow for short variable. The min and max values of the corresponding field is indicated, and a suggestion is made for changing scale\_factor and add\_offset.

-verbose : this implicitly activates -check. It gives the same kind of information but in case of 3D variables, details are given for each level.

For instance: *cdf16bit ORCA025-G70\_y2004m02\_gridT.nc -check*

will produce a cdf16bit file with the same variable name as in the input file.

**Required mesh\_mask files or other files:** none

**Output:** output is done on cdf16bit.nc file.

**Comments:** A DRAKKAR rule is to rename this file as the original 32 bits file, but with nc16 extension. The -check option is encouraged at least when initiating the conversion. If the scale\_factor and add\_offset are not adequate, the only way is to get the code and change the values, which are hard coded (in fact, what is hard coded are the min and max value for a given variable; scale\_factor and add\_offset are deduced from them).

**Details:** The standard variables of model output are automatically recognized. Variables not in the list are kept in float. At present, only the vertical diffusivity (votkeavt) is saved using log10.

votemper	! Potential temperature (Deg C)
vosaline	! Salinity (PSU)
sossheig	! Sea Surface Height (m)
somxl010	! Mixed layer depth (m)
sohefldo	! Total Heat flux Down (W/m2)
soshfldo	! Solar Heat flux Down (W/m2)
sowaflup	! Evaporation - Precipitation Up ( kg/m2/s)
sowaflup	! SSS damping term Up (kg/m2/s )
iowaflup	! ???
sowaflcd	! Concentration Dilution water flux (kg/m2/s)
solhflup	! Latent Heat Flux Up (W/m2)
solwfldo	! Long Wave radiation Heat flux Down (W/m2)
sosbhflup	! Sensible Heat Flux Up (W/m2)
vozocrtx	! Zonal Velocity U (m/s)
sozotaux	! Zonal Wind Stress (N/m2)
vomecrtx	! Meridional Velocity V (m/s)

```

sometauy      ! Meridional Wind Stress (N/m2)
vovecrtz      ! Vertical Velocity W (m/s)
votkeavt      ! Vertical mixing coef log(avt) log(m2/s)    : USE SAVE_LOG10
isnowthi      ! Snow Thickness (m)
iicethic      ! Ice Thickness (m)
iiceprod      ! Ice Production (m/kt) (step ice)
ileadfra      ! Ice Lead Fraction (%) (In fact, ice concentration)
iicetemp      ! Ice Temperature (Deg C )
ioceflxb      ! Ocean Ice flux (W/m2)
iicevelu      ! Zonal Ice Velocity (m/s) (at U point)
iicevelv      ! Meridional Ice Velocity (m/s) (at V point)
isstempe      ! Sea Surface Temperature (Deg C)
isssalin      ! Sea Surface Salinity (PSU)
iocetflx      ! Total Flux at Ocean Surface (W/m2)
iocesflx      ! Solar Flux at Ocean Surface (W/m2)
iocwnsfl      ! Non Solar Flux at Ocean surface (W/m2)
iocesafl      ! Salt Flux at Ocean Surface (kg/m2/kt)
iocestru      ! Zonal Ice Ocean Stress (N/m2)
iocestrv      ! Meridional Ice Ocean Stress (N/m2)
iicesflx      ! Solar FLux at ice/ocean Surface (W/m2)
iicenflx      ! Non Solar FLux at ice/ocean Surface (W/m2)
isnowpre      ! Snow Precipitation (kg/day)

```

A more flexible way to operate will be imagined soon !

Let float being the physical value and short the stored value, then the following formula applies

$$float = short \times scale\_factor + add\_offset$$

**Associated script:** cdf16bit.ll

### **cdfconvert:**

**Purpose:** Convert a set of dimgfile (Clipper like) to a set of CDF files (Drakkar like )

**Usage:** *cdfconvert 'Clipper tag ' 'CLIPPER confcase'*

**Input:** Clipper tag is the tag of the dimg clipper file.

CLIPPER confcase is for instance ATL6-V6

**Required mesh\_mask files or other files:** mesh\_hgr.nc and mesh\_zgr.nc (for Clipper grid, produced by coordinates2nc)

**Output:** CONFCASE\_TAG\_grid[TUV].nc, CONFCASE\_TAG\_PSI.nc with the standard name for variables.

**Associated script:** convclipper2nc.ksh

**Remark:** As you can see in the associated script, for a given tag, the full bunch of clipper files must be located in the current directory. (*ie* \_U\_, \_V\_, \_T\_, \_S\_, \_2D\_ and eventually \_SSH\_, \_UU\_ and \_VV\_ )

**cdfflxconv:**

**Purpose:** Convert a set of fluxes dimgfile (Clipper like) to a set of CDF files (Drakkar like )

**Usage:** *cdfflxconv YEAR CONFIG*

**Input:** YEAR is the year on 4 digits, CONFIG, is the CLIPPER config name (*eg* ATL3

**Required mesh\_mask files or other files:** clipper coordinates.diams file.

**Output:** This program creates 6 netcdf file with the standard NEMO name for forced simulation:

```
ECMWF_emp_1d_${year}.${config}.nc  
ECMWF_qnet_1d_${year}.${config}.nc  
ECMWF_qsr_1d_${year}.${config}.nc  
ECMWF_sst_1d_${year}.${config}.nc  
ECMWF_taux_1d_${year}.${config}.nc  
ECMWF_tauy_1d_${year}.${config}.nc
```

**Associated script:** cdfflxconv.ll

**Remark:**



### **cdfsstconv:**

**Purpose:** Convert a set of SST dimgfile (Clipper like) to a set of CDF files (Drakkar like )

**Usage:** *cdfsstconv YEAR CONFIG*

**Input:** YEAR is the year on 4 digits, CONFIG, is the CLIPPER config name (*eg* ATL3

**Required mesh\_mask files or other files:** clipper coordinates.diams file.

**Output:** This program creates 6 netcdf file with the standard NEMO name for forced simulation:

ECMWF\_sst\_1d\_\${year}.\${config}.nc

**Associated script:** cdfsstconv.ll

**Remark:** This is in fact a subset of cdfflxconv, limited to SST by ugly goto: :(

### **cdfstrconv:**

**Purpose:** Convert a set of STRESS dimgfile (Clipper like) to a set of CDF files (Drakkar like )

**Usage:** *cdfstrconv YEAR CONFIG*

**Input:** YEAR is the year on 4 digits, CONFIG, is the CLIPPER config name (*eg* ATL3

**Required mesh\_mask files or other files:** clipper coordinates.diams file.

**Output:** This program creates 6 netcdf file with the standard NEMO name for forced simulation:

ECMWF\_taux\_1d\_\${year}.\${config}.nc  
ECMWF\_tauy\_1d\_\${year}.\${config}.nc

**Associated script:** cdfstrconv.ll

**Remark:** This is in fact a subset of cdfflxconv, limited to STRESS by ugly goto: :(

**cdfmsk:**

**Purpose:** Compute the number of sea grid points from a mask file given on input

**Usage:** *cdfmsk maskfile*

**Input:** nfile is the name of the mask file to look at.

**Required mesh\_mask files or other files:** none, except the mask given as input.

**Output:** output is done on standard output.

**Associated script:** none

**Remark:** The interest of this program is limited; it provides a very stable info for a given config  
...

**cdfmsksal:**

**Purpose:** Compute a bimg (chart compliant) mask file for the surface of the model, from the salinity field.

**Usage:** *cdfmsksal gridT\_file*

**Input:** any file holding a vosaline field

**Required mesh\_mask files or other files:** none.

**Output:** Output is done on a bimg file, called tmask.bimg

**Associated script:** none

**Remark:** The output is specifically dedicated to the chart plotting program. It is usefull for masking (*e.g.* forcing files) on the fly. Purely netcdf/Ferret acros may skip this one !

**cdfvita:**

**Purpose:** Compute surface velocity components on the A-grid (T-point), from the C-grid opa output. It also computes the module of the velocity on the A-grid.

**Usage:** *cdfvita gridU gridV*

**Input:** gridU, gridV: files holding the variables vozocrtx and vomecrty on the C-grid

**Required mesh\_mask files or other files:** none.

**Output:** Output file is vita.nc, with the variables sovitua, sovitva , sovitmod

**Associated script:** none

**Remark:** This program is practically ready to treat the full 3D case, if necessary

### **cdfimprovechk:**

**Purpose:** Given a file with gridded 'observed' or 'sea-truth' value (*e.g* Levitus files), and a reference model output for the same quantity as the observation, this program tests a 3<sup>rd</sup> file (model output), giving an estimate of the improvement of the test, with respect to the reference.

**Usage:** *cdfimprovechk cdfvariable obs.nc ref.nc test.nc*

**Input:** *cdfvariable* = variable name used for checking. Must be in all 3 files.

*obs.nc* : observation file ('sea truth')

*ref.nc* : reference file ('model base run')

*test.nc* : test file (' model sensitivity experiment')

**Required mesh\_mask files or other files:** none.

**Output:** *chk.nc*, same variable (but not same sense!)

Better than a long speech, the improvement estimates is given by:

$$chk = (reference - test)/(reference - observation) \cdot mask$$

Where this value is  $< 1$  and  $> 0$ , the test is better than the reference (with respect to the working variable). Where it is greater than 1 it indicates a degradation of the test. Where the value is  $< 0$ , it denotes an over-shoot of the correction. Where the value is  $< -1$ , the overshoot gives a worse solution than the reference.

**Associated script:** none

**Example:** In particular, it was written for the tuning of TKE parameter and impact on the summer mixed layer depth. The observations files comes from the de Boyer Montaigne climatology; the reference case was with a standard TKE scheme. Different test cases where with the new improved tke scheme. The results are interesting but not necessarily easy to understand or interpret.

**cdfcsp:**

**Purpose:** Change missing value to 0.

**Usage:** *cdfcsp 'list of files'*

**Input:** input file are typically files from OPA8.2, with a missing value of 1.e20

**Required mesh\_mask files or other files:** none.

**Output:** CAUTION: The input file is rewritten with the correction on the missing value.

# Contents

<b>1</b>	<b>Statistics</b>	<b>3</b>
	cdfmoy . . . . .	3
	cdfmoy_sp . . . . .	3
	cdfmoy_sal2_temp2 . . . . .	4
	cdfmoy_annual . . . . .	4
	cdfmean . . . . .	5
	cdfmean-full . . . . .	6
	cdfmeanvar . . . . .	7
	cdfheatc . . . . .	8
	cdfheatc . . . . .	8
	cdfmxlheatc-ful . . . . .	10
	cdfmxlsaltc . . . . .	10
	cdfzonalmean . . . . .	11
	cdfzonalsum . . . . .	12
	cdfzonalout . . . . .	13
	cdfvT . . . . .	14
	cdfeke . . . . .	15
	cdfrmsssh . . . . .	16
	cdfstdevw . . . . .	17
	cdfstdevts . . . . .	18
	cdfstd . . . . .	19
<b>2</b>	<b>Transports</b>	<b>20</b>
	cdfmhst . . . . .	20
	cdfmhst-full . . . . .	21
	cdfhflx . . . . .	22
	cdfvhst . . . . .	23
	cdfvhst-full . . . . .	24
	cdfpsi . . . . .	25
	cdfpsi-full . . . . .	26
	cdfpsi-open . . . . .	27
	cdfmoc . . . . .	28
	cdfmocsig . . . . .	29
	cdfmocsig-full . . . . .	30
	cdfmoc-full . . . . .	31
	cdfmocatl . . . . .	32
	cdftransportiz . . . . .	33
	cdftransportiz-full . . . . .	33
	cdftransportiz_noheat . . . . .	34
	cdmmasstrp . . . . .	35
	cdmmasstrp-full . . . . .	35
	cdfsigtrp . . . . .	36
	cdfsigtrp-full . . . . .	37
	cdftemptrp-full . . . . .	38
	cdftempvol-full . . . . .	38

<b>3</b>	<b>Derived quantities</b>	<b>39</b>
	cdfsig0 . . . . .	39
	cdfsigi . . . . .	39
	cdfbottomsig0 . . . . .	40
	cdfbottom . . . . .	41
	cdfrhoproj . . . . .	42
	cdfbn2 . . . . .	43
	cdfets . . . . .	44
	cdfcurl . . . . .	45
	cdfpv . . . . .	46
	cdfpvor . . . . .	47
	cdfw . . . . .	48
	cdfmxl . . . . .	49
	cdfmxl-full . . . . .	49
	cdficediags . . . . .	50
	cdfcensus . . . . .	51
<b>4</b>	<b>Extracting and information tools</b>	<b>52</b>
	cdfprofile . . . . .	52
	cdprobe . . . . .	53
	cdwhereij . . . . .	53
	cdffindij . . . . .	54
	cdflip . . . . .	55
	cdmaxmoc . . . . .	56
	cdzoom . . . . .	57
	cdfbathy . . . . .	58
	cdmax . . . . .	60
	cdmax_sp . . . . .	61
	cdf16bit . . . . .	62
	cdconvert . . . . .	63
	cdfflxconv . . . . .	64
	cdsstconv . . . . .	65
	cdfstreconv . . . . .	65
	cdfmsk . . . . .	66
	cdfmsksal . . . . .	67
	cdfvita . . . . .	68
	cdfimprovechk . . . . .	69
	cdfcsp . . . . .	70



## Index

cdf16bit, 62  
cdfbathy, 58  
cdfbn2, 43  
cdfbottom, 41  
cdfbottomsig0, 40  
cdfcensus, 51  
cdfclip, 55  
cdfconvert, 63  
cdfcsp, 70  
cdfcurl, 45  
cdfeke, 15  
cdfets, 44  
cdffindij, 54  
cdfflxconv, 64  
cdfheatc, 8  
cdfhflx, 22  
cdficediags, 50  
cdfimprovechk, 69  
cdfmasstrp, 35  
cdfmasstrp-full, 35  
cdfmax, 60  
cdfmax\_sp, 61  
cdfmaxmoc, 56  
cdfmean, 5  
cdfmean-full, 6  
cdfmeanvar, 7  
cdfmhst, 20  
cdfmhst-full, 21  
cdfmoc, 28  
cdfmoc-full, 31  
cdfmocatl, 32  
cdfmocsig, 29  
cdfmocsig-full, 30  
cdfmoy, 3  
cdfmoy\_annual, 4  
cdfmoy\_sal2\_temp2, 4  
cdfmoy\_sp, 3  
cdfmsk, 66  
cdfmsksal, 67  
cdfmxl, 49  
cdfmxl-full, 49  
cdfmxlheate-ful, 10  
cdfmxlsaltc, 10  
cdfprobe, 53  
cdfprofile, 52  
cdfpsi, 25  
cdfpsi-full, 26  
cdfpsi-open, 27  
cdfpv, 46  
cdfpvor, 47  
cdfrhoproj, 42  
cdfrmssh, 16  
cdfsig0, 39  
cdfsigi, 39  
cdfsigtrp, 36  
cdfsigtrp-full, 37  
cdfsstconv, 65  
cdfstd, 19  
cdfstdevts, 4, 18  
cdfstdevw, 17  
cdfstrconv, 65  
cdftemptrp-full, 38  
cdftempvol-full, 38  
cdftransportiz, 33  
cdftransportiz-full, 33  
cdftransportiz\_noheat, 34  
cdfvhst, 23  
cdfvhst-full, 24  
cdfvita, 68  
cdfvT, 14  
cdfw, 48  
cdfwhereij, 53  
cdfzonalmean, 11  
cdfzonalout, 13  
cdfzonalsum, 12  
cdfzoom, 57