

# CDO User's Guide

---

Climate Data Operators  
Version 1.7.0  
October 2015

Uwe Schulzweida – *MPI for Meteorology*

---

# Contents

<b>1. Introduction</b>	<b>6</b>
1.1. Building from sources	6
1.1.1. Compilation	7
1.1.2. Installation	7
1.2. Usage	7
1.2.1. Options	8
1.2.2. Environment variables	9
1.2.3. Operators	9
1.2.4. Operator chaining	9
1.2.5. Parallelized operators	10
1.2.6. Operator parameter	10
1.3. Horizontal grids	10
1.3.1. Grid area weights	10
1.3.2. Grid description	11
1.4. Z-axis description	13
1.5. Time axis	14
1.5.1. Absolute time	14
1.5.2. Relative time	15
1.5.3. Conversion of the time	15
1.6. Parameter table	15
1.7. Missing values	15
1.7.1. Mean and average	16
1.8. Percentile	16
1.8.1. Percentile over timesteps	17
<b>2. Reference manual</b>	<b>18</b>
2.1. Information	19
2.1.1. INFO - Information and simple statistics	20
2.1.2. SINFO - Short information	21
2.1.3. DIFF - Compare two datasets field by field	22
2.1.4. NINFO - Print the number of parameters, levels or times	23
2.1.5. SHOWINFO - Show variables, levels or times	24
2.1.6. FILEDES - Dataset description	25
2.2. File operations	26
2.2.1. COPY - Copy datasets	27
2.2.2. REPLACE - Replace variables	28
2.2.3. DUPLICATE - Duplicates a dataset	28
2.2.4. MERGEGRID - Merge grid	28
2.2.5. MERGE - Merge datasets	29
2.2.6. SPLIT - Split a dataset	30
2.2.7. SPLITTIME - Split timesteps of a dataset	32
2.2.8. SPLITSEL - Split selected timesteps	33
2.2.9. DISTGRID - Distribute horizontal grid	34
2.2.10. COLLGRID - Collect horizontal grid	35
2.3. Selection	36
2.3.1. SELECT - Select fields	37
2.3.2. SELVAR - Select fields	38
2.3.3. SELTIME - Select timesteps	40
2.3.4. SELBOX - Select a box of a field	42

2.4.	Conditional selection	43
2.4.1.	COND - Conditional select one field	44
2.4.2.	COND2 - Conditional select two fields	44
2.4.3.	CONDC - Conditional select a constant	45
2.5.	Comparison	46
2.5.1.	COMP - Comparison of two fields	47
2.5.2.	COMPC - Comparison of a field with a constant	48
2.6.	Modification	49
2.6.1.	SETPARTAB - Set parameter table	51
2.6.2.	SET - Set field info	53
2.6.3.	SETTIME - Set time	54
2.6.4.	CHANGE - Change field header	56
2.6.5.	SETGRID - Set grid information	57
2.6.6.	SETZAXIS - Set z-axis information	58
2.6.7.	SETGATT - Set global attribute	59
2.6.8.	INVERT - Invert latitudes	60
2.6.9.	INVERTLEV - Invert levels	60
2.6.10.	MASKREGION - Mask regions	61
2.6.11.	MASKBOX - Mask a box	62
2.6.12.	SETBOX - Set a box to constant	63
2.6.13.	ENLARGE - Enlarge fields	64
2.6.14.	SETMISS - Set missing value	65
2.7.	Arithmetic	67
2.7.1.	EXPR - Evaluate expressions	69
2.7.2.	MATH - Mathematical functions	71
2.7.3.	ARITHC - Arithmetic with a constant	72
2.7.4.	ARITH - Arithmetic on two datasets	73
2.7.5.	MONARITH - Monthly arithmetic	74
2.7.6.	YHOURARITH - Multi-year hourly arithmetic	75
2.7.7.	YDAYARITH - Multi-year daily arithmetic	76
2.7.8.	YMONARITH - Multi-year monthly arithmetic	77
2.7.9.	YSEASARITH - Multi-year seasonal arithmetic	78
2.7.10.	ARITHDAYS - Arithmetic with days	78
2.8.	Statistical values	79
2.8.1.	CONSECSTAT - Consecutive timestep periods	85
2.8.2.	ENSSTAT - Statistical values over an ensemble	86
2.8.3.	ENSSTAT2 - Statistical values over an ensemble	88
2.8.4.	ENSVAL - Ensemble validation tools	89
2.8.5.	FLDSTAT - Statistical values over a field	91
2.8.6.	ZONSTAT - Zonal statistical values	93
2.8.7.	MERSTAT - Meridional statistical values	94
2.8.8.	GRIDBOXSTAT - Statistical values over grid boxes	95
2.8.9.	VERTSTAT - Vertical statistical values	96
2.8.10.	TIMSELSTAT - Time range statistical values	97
2.8.11.	TIMSELPCTL - Time range percentile values	98
2.8.12.	RUNSTAT - Running statistical values	99
2.8.13.	RUNPCTL - Running percentile values	100
2.8.14.	TIMSTAT - Statistical values over all timesteps	101
2.8.15.	TIMPCTL - Percentile values over all timesteps	102
2.8.16.	HOURLSTAT - Hourly statistical values	103
2.8.17.	HOURLPCTL - Hourly percentile values	104
2.8.18.	DAYSTAT - Daily statistical values	105
2.8.19.	DAYPCTL - Daily percentile values	106
2.8.20.	MONSTAT - Monthly statistical values	107
2.8.21.	MONPCTL - Monthly percentile values	108
2.8.22.	YEARMONSTAT - Yearly mean from monthly data	109
2.8.23.	YEARSTAT - Yearly statistical values	110
2.8.24.	YEARPCTL - Yearly percentile values	111

2.8.25.	SEASSTAT - Seasonal statistical values	112
2.8.26.	SEASPCTL - Seasonal percentile values	113
2.8.27.	YHOURLSTAT - Multi-year hourly statistical values	114
2.8.28.	YDAYSTAT - Multi-year daily statistical values	116
2.8.29.	YDAYPCTL - Multi-year daily percentile values	118
2.8.30.	YMONSTAT - Multi-year monthly statistical values	119
2.8.31.	YMONPCTL - Multi-year monthly percentile values	121
2.8.32.	YSEASSTAT - Multi-year seasonal statistical values	122
2.8.33.	YSEASPCTL - Multi-year seasonal percentile values	123
2.8.34.	YDRUNSTAT - Multi-year daily running statistical values	124
2.8.35.	YDRUNPCTL - Multi-year daily running percentile values	126
2.9.	Correlation and co.	127
2.9.1.	FLDCOR - Correlation in grid space	128
2.9.2.	TIMCOR - Correlation over time	128
2.9.3.	FLDCOVAR - Covariance in grid space	129
2.9.4.	TIMCOVAR - Covariance over time	129
2.10.	Regression	130
2.10.1.	REGRES - Regression	131
2.10.2.	DETREND - Detrend time series	131
2.10.3.	TREND - Trend of time series	132
2.10.4.	SUBTREND - Subtract a trend	132
2.11.	EOFs	133
2.11.1.	EOFS - Empirical Orthogonal Functions	134
2.11.2.	EOFCOEFF - Principal coefficients of EOFs	136
2.12.	Interpolation	137
2.12.1.	REMAPBIL - Bilinear interpolation	138
2.12.2.	REMAPBIC - Bicubic interpolation	139
2.12.3.	REMAPNN - Nearest neighbor remapping	140
2.12.4.	REMAPDIS - Distance-weighted average remapping	141
2.12.5.	REMAPYCON - First order conservative remapping	142
2.12.6.	REMAPCON - First order conservative remapping	143
2.12.7.	REMAPCON2 - Second order conservative remapping	144
2.12.8.	REMAPLAF - Largest area fraction remapping	145
2.12.9.	REMAP - Grid remapping	146
2.12.10.	REMAPETA - Remap vertical hybrid level	147
2.12.11.	VERTINTML - Vertical interpolation	149
2.12.12.	VERTINTAP - Vertical interpolation	149
2.12.13.	INTLEVEL - Linear level interpolation	150
2.12.14.	INTLEVEL3D - Linear level interpolation from/to 3d vertical coordinates	151
2.12.15.	INTTIME - Time interpolation	152
2.12.16.	INTYEAR - Year interpolation	153
2.13.	Transformation	154
2.13.1.	SPECTRAL - Spectral transformation	155
2.13.2.	WIND - Wind transformation	156
2.14.	Import/Export	157
2.14.1.	IMPORTBINARY - Import binary data sets	158
2.14.2.	IMPORTCMSAF - Import CM-SAF HDF5 files	159
2.14.3.	IMPORTAMSR - Import AMSR binary files	160
2.14.4.	INPUT - Formatted input	161
2.14.5.	OUTPUT - Formatted output	162
2.14.6.	OUTPUTTAB - Table output	163
2.15.	Miscellaneous	164
2.15.1.	GRADSDES - GrADS data descriptor file	165
2.15.2.	AFTERBURNER - ECHAM standard post processor	166
2.15.3.	FILTER - Time series filtering	168
2.15.4.	GRIDCELL - Grid cell quantities	169
2.15.5.	SMOOTH9 - 9 point smoothing	170
2.15.6.	REPLACEVALUES - Replace variable values	170

2.15.7. TIMSORT - Timsort . . . . .	171
2.15.8. VARGEN - Generate a field . . . . .	171
2.15.9. ROTUVB - Rotation . . . . .	173
2.15.10.MASTRFU - Mass stream function . . . . .	173
2.15.11.DERIVEPAR - Sea level pressure . . . . .	174
2.15.12.ADISIT - Potential temperature to in-situ temperature and vice versa . . . . .	175
2.15.13.RHOPOT - Calculates potential density . . . . .	175
2.15.14.HISTOGRAM - Histogram . . . . .	176
2.15.15.SETHALO - Set the left and right bounds of a field . . . . .	176
2.15.16.WCT - Windchill temperature . . . . .	177
2.15.17.FDNS - Frost days where no snow index per time period . . . . .	177
2.15.18.STRWIN - Strong wind days index per time period . . . . .	177
2.15.19.STRBRE - Strong breeze days index per time period . . . . .	178
2.15.20.STRGAL - Strong gale days index per time period . . . . .	178
2.15.21.HURR - Hurricane days index per time period . . . . .	178
<b>A. Environment Variables</b>	<b>180</b>
<b>B. Parallelized operators</b>	<b>181</b>
<b>C. Standard name table</b>	<b>182</b>
<b>D. Grid description examples</b>	<b>183</b>
D.1. Example of a curvilinear grid description . . . . .	183
D.2. Example description for an unstructured grid . . . . .	184
<b>Operator index</b>	<b>185</b>

# 1. Introduction

The Climate Data Operators (**CDO**) software is a collection of many operators for standard processing of climate and forecast model data. The operators include simple statistical and arithmetic functions, data selection and subsampling tools, and spatial interpolation. **CDO** was developed to have the same set of processing functions for GRIB [GRIB] and netCDF [netCDF] datasets in one package.

The Climate Data Interface [CDI] is used for the fast and file format independent access to GRIB and netCDF datasets. The local MPI-MET data formats SERVICE, EXTRA and IEG are also supported.

There are some limitations for GRIB and netCDF datasets. A GRIB dataset has to be consistent, similar to netCDF. That means all time steps need to have the same variables, and within a time step each variable may occur only once. NetCDF datasets are only supported for the classic data model and arrays up to 4 dimensions. These dimensions should only be used by the horizontal and vertical grid and the time. The netCDF attributes should follow the GDT, COARDS or CF Conventions.

The user interface and some operators are similar to the PINGO [PINGO] package.

The main **CDO** features are:

- More than 700 operators available
- Modular design and easily extendable with new operators
- Very simple UNIX command line interface
- A dataset can be processed by several operators, without storing the interim results in files
- Most operators handle datasets with missing values
- Fast processing of large datasets
- Support of many different grid types
- Tested on many UNIX/Linux systems, Cygwin, and MacOS-X

## 1.1. Building from sources

This section describes how to build **CDO** from the sources on a UNIX system. **CDO** uses the GNU configure and build system for compilation. The only requirement is a working ANSI C99 compiler.

First go to the [download](https://code.zmaw.de/projects/cdo) page (<https://code.zmaw.de/projects/cdo>) to get the latest distribution, if you do not have it yet.

To take full advantage of **CDO** features the following additional libraries should be installed:

- Unidata [netCDF](http://www.unidata.ucar.edu/packages/netcdf) library (<http://www.unidata.ucar.edu/packages/netcdf>) version 3 or higher. This is needed to process netCDF [netCDF] files with **CDO**.
- The ECMWF [GRIB\\_API](http://www.ecmwf.int/products/data/software/grib_api.html) ([http://www.ecmwf.int/products/data/software/grib\\_api.html](http://www.ecmwf.int/products/data/software/grib_api.html)) version 1.12 or higher. This library is needed to process GRIB2 files with **CDO**.
- HDF5 [szip](http://www.hdfgroup.org/doc_resource/SZIP) library ([http://www.hdfgroup.org/doc\\_resource/SZIP](http://www.hdfgroup.org/doc_resource/SZIP)) version 2.1 or higher. This is needed to process szip compressed GRIB [GRIB] files with **CDO**.
- [HDF5](http://www.hdfgroup.org/HDF5) library (<http://www.hdfgroup.org/HDF5>) version 1.6 or higher. This is needed to import CM-SAF [CM-SAF] HDF5 files with the **CDO** operator `import_cmsaf`.

- **PROJ.4** library (<http://trac.osgeo.org/proj>) version 4.6 or higher.  
This is needed to convert Sinusoidal and Lambert Azimuthal Equal Area coordinates to geographic coordinates, for e.g. remapping.

**CDO** is a multi-threaded application. Therefor all the above libraries should be compiled thread safe. Using non-threadsafe libraries could cause unexpected errors!

### 1.1.1. Compilation

Compilation is done by performing the following steps:

1. Unpack the archive, if you haven't done that yet:

```
gunzip cdo-$VERSION.tar.gz    # uncompress the archive
tar xf cdo-$VERSION.tar       # unpack it
cd cdo-$VERSION
```

2. Run the configure script:

```
./configure
```

- Optionally with netCDF [[netCDF](#)] support:

```
./configure --with-netcdf=<netCDF root directory>
```

- and with GRIB\_API:

```
./configure --with-grib_api=<GRIB_API root directory>
```

For an overview of other configuration options use

```
./configure --help
```

3. Compile the program by running make:

```
make
```

The program should compile without problems and the binary (`cdo`) should be available in the `src` directory of the distribution.

### 1.1.2. Installation

After the compilation of the source code do a `make install`, possibly as root if the destination permissions require that.

```
make install
```

The binary is installed into the directory `<prefix>/bin`. `<prefix>` defaults to `/usr/local` but can be changed with the `-prefix` option of the configure script.

Alternatively, you can also copy the binary from the `src` directory manually to some `bin` directory in your search path.

## 1.2. Usage

This section describes how to use **CDO**. The syntax is:

```
cdo [ Options ] Operator1 [ -Operator2 [ -OperatorN ] ]
```

### 1.2.1. Options

All options have to be placed before the first operator. The following options are available for all operators:

- a Generate an absolute time axis.  
 -b <nbits> Set the number of bits for the output precision. The valid precisions depend on the file format:

<format>	<nbits>
grb, grb2	P1 - P24
nc, nc2, nc4, nc4c	I8/I16/I32/F32/F64
grb2, srv, ext, ieg	F32/F64

For srv, ext and ieg format the letter L or B can be added to set the byteorder to Little or Big endian.

- f <format> Set the output file format. The valid file formats are:

File format	<format>
GRIB version 1	grb
GRIB version 2	grb2
netCDF	nc
netCDF version 2 (64-bit)	nc2
netCDF-4 (HDF5)	nc4
netCDF-4 classic	nc4c
SERVICE	srv
EXTRA	ext
IEG	ieg

GRIB2 is only available if **CDO** was compiled with GRIB\_API support and all netCDF file types are only available if **CDO** was compiled with netCDF support!

- g <grid> Define the default grid description by name or from file (see chapter 1.3 on page 11). Available grid names are: r<NX>x<NY>, lon=<LON>/lat=<LAT>, n<N>, gme<NI>  
 -h, --help Help information for the operators.  
 --no\_history Do not append to netCDF *history* global attribute.  
 --netcdf\_hdr\_pad, --hdr\_pad, --header\_pad <nbr> Pad netCDF output header with *nbr* bytes.  
 -k <chunktype> NetCDF4 chunk type: auto, grid or lines.  
 -L Lock I/O (sequential access).  
 -M Switch to indicate that the I/O streams have missing values.  
 -m <missval> Set the default missing value (default: -9e+33).  
 -O Overwrite existing output file, if checked.  
 Existing output file is checked only for: ens<STAT>, merge, mergetime  
 -P <nthreads> Set number of OpenMP threads (Only available if OpenMP support was compiled in).  
 --percentile <method> Percentile method: nrank nist numpy numpy\_lower numpy\_higher numpy\_nearest  
 -Q Alphanumeric sorting of netCDF parameter names.  
 --reduce\_dim Reduce netCDF dimensions (module: TIMSTAT, FLDSTAT).  
 -R, --regular Convert GRIB1 data from reduced to regular grid (only with cgribex lib).  
 -r Generate a relative time axis.  
 -S Create an extra output stream for the module TIMSTAT. This stream contains the number of non missing values for each output period.  
 -s, --silent Silent mode.  
 -t <partab> Set the GRIB1 default parameter table name or file (see chapter 1.6 on page 15). Predefined tables are: echam4 echam5 echam6 mpiom1 ecmwf remo  
 -V, --version Print the version number.  
 -v, --verbose Print extra details for some operators.  
 -W Print extra warning messages.  
 -z szip SZIP compression of GRIB1 records.  
 jpeg JPEG compression of GRIB2 records.



`zip[_1-9]` Deflate compression of netCDF4 variables.

### 1.2.2. Environment variables

There are some environment variables which influence the behavior of **CDO**. An incomplete list can be found in [Appendix A](#).

Here is an example to set the environment variable `CDO_RESET_HISTORY` for different shells:

```
Bourne shell (sh):  CDO_RESET_HISTORY=1 ; export CDO_RESET_HISTORY
Korn shell (ksh):   export CDO_RESET_HISTORY=1
C shell (csh):      setenv CDO_RESET_HISTORY 1
```

### 1.2.3. Operators

There are more than 700 operators available. A detailed description of all operators can be found in the [Reference Manual](#) section.

### 1.2.4. Operator chaining

All operators with a fixed number of input streams and one output stream can pipe the result directly to an other operator. The operator must begin with "-", in order to combine it with others. This can improve the performance by:

- reducing unnecessary disk I/O
- parallel processing

Use

```
cdo sub -dayavg ifile2 -timavg ifile1 ofile
```

instead of

```
cdo timavg ifile1 tmp1
cdo dayavg ifile2 tmp2
cdo sub tmp2 tmp1 ofile
rm tmp1 tmp2
```

All operators with an arbitrary number of input streams (*ifiles*) can't be combined with other operators if these operators are used with more than one input stream. Here is an incomplete list of these operators: [copy](#), [cat](#), [merge](#), [mergetime](#), [select](#), [ens<STAT>](#)

Use single quotes if the input stream names are generated with wildcards. In this case CDO will do the pattern matching and the output can be combined with other operators. Here is an example for this feature:

```
cdo timavg -select,name=temperature 'ifile?' ofile
```

The CDO internal wildcard expansion is using the *glob()* function. Therefore internal wildcard expansion is not available on operating systems without the *glob()* function!

**Note:** Operator chaining is implemented over POSIX Threads (pthreads). Therefore this **CDO** feature is not available on operating systems without POSIX Threads support!

### 1.2.5. Parallelized operators

Some of the **CDO** operators are shared memory parallelized with OpenMP. An OpenMP-enabled C compiler is needed to use this feature. Users may request a specific number of OpenMP threads `nthreads` with the `'-P'` switch.

Here is an example to distribute the bilinear interpolation on 8 OpenMP threads:

```
cdo -P 8 remapbil,targetgrid ifile ofile
```

Many **CDO** operators are I/O-bound. This means most of the time is spend in reading and writing the data. Only compute intensive **CDO** operators are parallelized. An incomplete list of OpenMP parallelized operators can be found in [Appendix B](#).

### 1.2.6. Operator parameter

Some operators need one or more parameter. A list of parameter is indicated by the seperator `','`.

- **STRING**

Unquoted characters without blanks and tabs. The following command select variables with the name pressure and tsurf:

```
cdo selvar,pressure,tsurf ifile ofile
```

- **FLOAT**

Floating point number in any representation. The following command sets the range between 0 and 273.15 of all fields to missing value:

```
cdo setrtomiss,0,273.15 ifile ofile
```

- **INTEGER**

A range of integer parameter can be specified by *first/last/inc*. To select the days 5, 6, 7, 8 and 9 use:

```
cdo selday,5/9 ifile ofile
```

The result is the same as:

```
cdo selday,5,6,7,8,9 ifile ofile
```

## 1.3. Horizontal grids

Physical quantities of climate models are typically stored on a horizontal grid. The maximum number of supported grid cells is 2147483647 (`INT_MAX`). This corresponds to a global regular lon/lat grid with 65455x32727 grid cells and a global resolution of 0.0055 degree.

### 1.3.1. Grid area weights

One single point of a horizontal grid represents the mean of a grid cell. These grid cells are typically of different sizes, because the grid points are of varying distance.

Area weights are individual weights for each grid cell. They are needed to compute the area weighted mean or variance of a set of grid cells (e.g. [fldmean](#) - the mean value of all grid cells). In **CDO** the area weights are derived from the grid cell area. If the cell area is not available then it will be computed from the geographical coordinates via spherical triangles. This is only possible if the geographical coordinates of the grid cell corners are available or derivable. Otherwise **CDO** gives a warning message and uses constant area weights for all grid cells.

The cell area is read automatically from a netCDF input file if a variable has the corresponding `"cell_measures"` attribute, e.g.:

```
var: cell_measures = "area: cell_area" ;
```

If the computed cell area is not desired then the **CDO** operator [setgridarea](#) can be used to set or overwrite the grid cell area.

### 1.3.2. Grid description

In the following situations it is necessary to give a description of a horizontal grid:

- Changing the grid description (operator: [setgrid](#))
- Horizontal interpolation (operator: [remapXXX](#) and [genXXX](#))
- Generating of variables (operator: [const](#), [random](#))

As now described, there are several possibilities to define a horizontal grid.

#### 1.3.2.1. Predefined grids

Predefined grids are available for global regular, gaussian or icosahedral-hexagonal GME grids.

**Global regular grid:** `global_<DXY>`

`global_<DXY>` defines a global regular lon/lat grid. The grid increment `<DXY>` can be selected at will. The longitudes start at  $\langle DXY \rangle / 2 - 180^\circ$  and the latitudes start at  $\langle DXY \rangle / 2 - 90^\circ$ .

**Global regular grid:** `r<NX>x<NY>`

`r<NX>x<NY>` defines a global regular lon/lat grid. The number of the longitudes `<NX>` and the latitudes `<NY>` can be selected at will. The longitudes start at  $0^\circ$  with an increment of  $(360 / \langle NX \rangle)^\circ$ . The latitudes go from south to north with an increment of  $(180 / \langle NY \rangle)^\circ$ .

**One grid point:** `lon=<LON>/lat=<LAT>`

`lon=<LON>/lat=<LAT>` defines a lon/lat grid with only one grid point.

**Global Gaussian grid:** `n<N>`

`n<N>` defines a global Gaussian grid. `N` specifies the number of latitudes lines between the Pole and the Equator. The longitudes start at  $0^\circ$  with an increment of  $(360 / nlon)^\circ$ . The gaussian latitudes go from north to south.

**Global icosahedral-hexagonal GME grid:** `gme<NI>`

`gme<NI>` defines a global icosahedral-hexagonal GME grid. `NI` specifies the number of intervals on a main triangle side.

#### 1.3.2.2. Grids from data files

You can use the grid description from an other datafile. The format of the datafile and the grid of the data field must be supported by **CDO**. Use the operator '[sinfo](#)' to get short informations about your variables and the grids. If there are more then one grid in the datafile the grid description of the first variable will be used.

### 1.3.2.3. SCRIP grids

SCRIP (Spherical Coordinate Remapping and Interpolation Package) uses a common grid description for curvilinear and unstructured grids. For more information about the convention see [SCRIP]. This grid description is stored in netCDF. Therefor it is only available if **CDO** was compiled with netCDF support!

SCRIP grid description example of a curvilinear MPIOM [MPIOM] GROB3 grid (only the netCDF header):

```
netcdf grob3s {
  dimensions:
    grid_size = 12120 ;
    grid_xsize = 120 ;
    grid_ysize = 101 ;
    grid_corners = 4 ;
    grid_rank = 2 ;
  variables:
    int grid_dims(grid_rank) ;
    float grid_center_lat(grid_ysize, grid_xsize) ;
      grid_center_lat:units = "degrees" ;
      grid_center_lat:bounds = "grid_corner_lat" ;
    float grid_center_lon(grid_ysize, grid_xsize) ;
      grid_center_lon:units = "degrees" ;
      grid_center_lon:bounds = "grid_corner_lon" ;
    int grid_imask(grid_ysize, grid_xsize) ;
      grid_imask:units = "unitless" ;
      grid_imask:coordinates = "grid_center_lon grid_center_lat" ;
    float grid_corner_lat(grid_ysize, grid_xsize, grid_corners) ;
      grid_corner_lat:units = "degrees" ;
    float grid_corner_lon(grid_ysize, grid_xsize, grid_corners) ;
      grid_corner_lon:units = "degrees" ;

  // global attributes:
    :title = "grob3s" ;
}
```

### 1.3.2.4. CDO grids

All supported grids can also be described with the **CDO** grid description. The following keywords can be used to describe a grid:

Keyword	Datatype	Description
<b>gridtype</b>	STRING	Type of the grid (gaussian, lonlat, curvilinear, unstructured).
<b>gridsize</b>	INTEGER	Size of the grid.
<b>xsize</b>	INTEGER	Size in x direction (number of longitudes).
<b>ysize</b>	INTEGER	Size in y direction (number of latitudes).
<b>xvals</b>	FLOAT ARRAY	X values of the grid cell center.
<b>yvals</b>	FLOAT ARRAY	Y values of the grid cell center.
<b>xnpole</b>	FLOAT	X value of the north pole (rotated grid).
<b>ynpole</b>	FLOAT	Y value of the north pole (rotated grid).
<b>angle</b>	FLOAT	Angle of the rotated north pole (default: 0).
<b>nvertex</b>	INTEGER	Number of the vertices for all grid cells.
<b>xbounds</b>	FLOAT ARRAY	X bounds of each gridbox.
<b>ybounds</b>	FLOAT ARRAY	Y bounds of each gridbox.
<b>xfirst, xinc</b>	FLOAT, FLOAT	Macros to define xvals with a constant increment, xfirst is the x value of the first grid cell center.
<b>yfirst, yinc</b>	FLOAT, FLOAT	Macros to define yvals with a constant increment, yfirst is the y value of the first grid cell center.

Which keywords are necessary depends on the gridtype. The following table gives an overview of the default values or the size with respect to the different grid types.

gridtype	lonlat	gaussian	curvilinear	unstructured
gridsize	xsize*ysize	xsize*ysize	xsize*ysize	<b>ncell</b>
xsize	<b>nlon</b>	<b>nlon</b>	<b>nlon</b>	gridsize
ysize	<b>nlat</b>	<b>nlat</b>	<b>nlat</b>	gridsize
xvals	xsize	xsize	gridsize	gridsize
yvals	ysize	ysize	gridsize	gridsize
xnpole	0			
ynpole	90			
angle	0			
nvertex	2	2	4	<b>nv</b>
xbounds	2*xsize	2*xsize	4*gridsize	nv*gridsize
ybounds	2*ysize	2*ysize	4*gridsize	nv*gridsize

The keywords `nvertex`, `xbounds` and `ybounds` are optional if area weights are not needed. The grid cell corners `xbounds` and `ybounds` have to rotate counterclockwise.

**CDO** grid description example of a T21 gaussian grid:

```

gridtype = gaussian
xsize    = 64
ysize    = 32
xfirst   = 0
xinc     = 5.625
yvals    = 85.76  80.27  74.75  69.21  63.68  58.14  52.61  47.07
           41.53  36.00  30.46  24.92  19.38  13.84  8.31  2.77
           -2.77 -8.31 -13.84 -19.38 -24.92 -30.46 -36.00 -41.53
           -47.07 -52.61 -58.14 -63.68 -69.21 -74.75 -80.27 -85.76

```

**CDO** grid description example of a global regular grid with 60x30 points:

```

gridtype = lonlat
xsize    = 60
ysize    = 30
xfirst   = -177
xinc     = 6
yfirst   = -87
yinc     = 6

```

For a lon/lat grid with a rotated pole, the north pole must be defined. As far as you define the keywords `xnpole`/`ynpole` all coordinate values are for the rotated system.

**CDO** grid description example of a regional rotated lon/lat grid:

```

gridtype = lonlat
xsize    = 81
ysize    = 91
xfirst   = -19.5
xinc     = 0.5
yfirst   = -25.0
yinc     = 0.5
xnpole   = -170
ynpole   = 32.5

```

Example **CDO** descriptions of a curvilinear and an unstructured grid can be found in [Appendix C](#).

## 1.4. Z-axis description

Sometimes it is necessary to change the description of a z-axis. This can be done with the operator `setzaxis`. This operator needs an ASCII formatted file with the description of the z-axis. The following keywords can be used to describe a z-axis:

Keyword	Datatype	Description
<b>zaxistype</b>	STRING	type of the z-axis
<b>size</b>	INTEGER	number of levels
<b>levels</b>	FLOAT ARRAY	values of the levels
<b>lbounds</b>	FLOAT ARRAY	lower level bounds
<b>ubounds</b>	FLOAT ARRAY	upper level bounds
<b>vctsize</b>	INTEGER	number of vertical coordinate parameters
<b>vct</b>	FLOAT ARRAY	vertical coordinate table

The keywords **lbounds** and **ubounds** are optional. **vctsize** and **vct** are only necessary to define hybrid model levels.

Available z-axis types:

Z-axis type	Description	Units
<b>surface</b>	Surface	
<b>pressure</b>	Pressure level	pascal
<b>hybrid</b>	Hybrid model level	
<b>height</b>	Height above ground	meter
<b>depth_below_sea</b>	Depth below sea level	meter
<b>depth_below_land</b>	Depth below land surface	centimeter
<b>isentropic</b>	Isentropic (theta) level	kelvin

Z-axis description example for pressure levels 100, 200, 500, 850 and 1000 hPa:

```
zaxistype = pressure
size      = 5
levels    = 10000 20000 50000 85000 100000
```

Z-axis description example for ECHAM5 L19 hybrid model levels:

```
zaxistype = hybrid
size      = 19
levels    = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
vctsize   = 40
vct       = 0 2000 4000 6046.10938 8267.92578 10609.5117 12851.1016 14698.5
            15861.125 16116.2383 15356.9258 13621.4609 11101.5625 8127.14453
            5125.14062 2549.96875 783.195068 0 0 0
            0 0 0 0.000338993268 0.00335718691 0.0130700432 0.0340771675
            0.0706498027 0.12591666 0.201195419 0.295519829 0.405408859
            0.524931908 0.646107674 0.759697914 0.856437683 0.928747177
            0.972985268 0.992281914 1
```

Note that the vctsize is twice the number of levels plus two and the vertical coordinate table must be specified for the level interfaces.

## 1.5. Time axis

A time axis describes the time for every timestep. Two time axis types are available: absolute time and relative time axis. **CDO** tries to maintain the actual type of the time axis for all operators.

### 1.5.1. Absolute time

An absolute time axis has the current time to each time step. It can be used without knowledge of the calendar. This is preferably used by climate models. In netCDF files the absolute time axis is represented by the unit of the time: "day as %Y%m%d.%f".

### 1.5.2. Relative time

A relative time is the time relative to a fixed reference time. The current time results from the reference time and the elapsed interval. The result depends on the calendar used. **CDO** supports the standard Gregorian, proleptic Gregorian, 360 days, 365 days and 366 days calendars. The relative time axis is preferably used by numerical weather prediction models. In netCDF files the relative time axis is represented by the unit of the time: "*time-units since reference-time*", e.g. "days since 1989-6-15 12:00".

### 1.5.3. Conversion of the time

Some programs which work with netCDF data can only process relative time axes. Therefore it may be necessary to convert from an absolute into a relative time axis. This conversion can be done for each operator with the **CDO** option '-r'. To convert a relative into an absolute time axis use the **CDO** option '-a'.

## 1.6. Parameter table

A parameter table is an ASCII formatted file to convert code numbers to variable names. Each variable has one line with its code number, name and a description with optional units in a blank separated list. It can only be used for GRIB, SERVICE, EXTRA and IEG formatted files. The **CDO** option '-t <partab>' sets the default parameter table for all input files. Use the operator 'setpartab' to set the parameter table for a specific file.

Example of a **CDO** parameter table:

134	aps	surface pressure [Pa]
141	sn	snow depth [m]
147	ahfl	latent heat flux [W/m**2]
172	slm	land sea mask
175	albedo	surface albedo
211	siced	ice depth [m]

## 1.7. Missing values

Most operators can handle missing values. The default missing value for GRIB, SERVICE, EXTRA and IEG files is  $-9.e^{33}$ . The **CDO** option '-m <missval>' overwrites the default missing value. In netCDF files the variable attribute '\_FillValue' is used as a missing value. The operator 'setmissval' can be used to set a new missing value.

The **CDO** use of the missing value is shown in the following tables, where one table is printed for each operation. The operations are applied to arbitrary numbers  $a$ ,  $b$ , the special case 0, and the missing value *miss*. For example the table named "addition" shows that the sum of an arbitrary number  $a$  and the missing value is the missing value, and the table named "multiplication" shows that 0 multiplied by missing value results in 0.

<b>addition</b>	b		miss
a	$a + b$		<i>miss</i>
miss	<i>miss</i>		<i>miss</i>
<b>subtraction</b>	b		miss
a	$a - b$		<i>miss</i>
miss	<i>miss</i>		<i>miss</i>
<b>multiplication</b>	b	0	miss
a	$a * b$	0	<i>miss</i>
0	0	0	0
miss	<i>miss</i>	0	<i>miss</i>
<b>division</b>	b	0	miss
a	$a/b$	<i>miss</i>	<i>miss</i>
0	0	<i>miss</i>	<i>miss</i>
miss	<i>miss</i>	<i>miss</i>	<i>miss</i>
<b>maximum</b>	b		miss
a	$\max(a, b)$		a
miss	b		<i>miss</i>
<b>minimum</b>	b		miss
a	$\min(a, b)$		a
miss	b		<i>miss</i>
<b>sum</b>	b		miss
a	$a + b$		a
miss	b		<i>miss</i>

The handling of missing values by the operations "minimum" and "maximum" may be surprising, but the definition given here is more consistent with that expected in practice. Mathematical functions (e.g. *log*, *sqrt*, etc.) return the missing value if an argument is the missing value or an argument is out of range.

All statistical functions ignore missing values, treating them as not belonging to the sample, with the side-effect of a reduced sample size.

### 1.7.1. Mean and average

An artificial distinction is made between the notions mean and average. The mean is regarded as a statistical function, whereas the average is found simply by adding the sample members and dividing the result by the sample size. For example, the mean of 1, 2, *miss* and 3 is  $(1 + 2 + 3)/3 = 2$ , whereas the average is  $(1 + 2 + \text{miss} + 3)/4 = \text{miss}/4 = \text{miss}$ . If there are no missing values in the sample, the average and mean are identical.

## 1.8. Percentile

There is no standard definition of percentile. All definitions yield to similar results when the number of values is very large. The following percentile methods are available in **CDO**:



Percentile method	Description
nrank	Nearest Rank method, the default method used in <b>CDO</b>
nist	The primary method recommended by NIST
numpy	numpy.percentile with the option interpolation set to 'linear'
numpy_lower	numpy.percentile with the option interpolation set to 'lower'
numpy_higher	numpy.percentile with the option interpolation set to 'higher'
numpy_nearest	numpy.percentile with the option interpolation set to 'nearest'

The percentile method can be selected with the **CDO** option `--percentile`. The Nearest Rank method is the default percentile method in **CDO**.

The different percentile methods can lead to different results, especially for small number of data values. Consider the ordered list {15, 20, 35, 40, 50, 55}, which contains six data values. Here is the result for the 30th, 40th, 50th, 75th and 100th percentiles of this list using the different percentile methods:

Percentile P	nrank	nist	numpy	numpy lower	numpy higher	numpy nearest
30th	20	21.5	27.5	20	35	35
40th	35	32	35	35	35	35
50th	35	37.5	37.5	35	40	40
75th	50	51.25	47.5	40	50	50
100th	55	55	55	55	55	55

### 1.8.1. Percentile over timesteps

The amount of data for time series can be very large. All data values need to be held in memory to calculate the percentile. The percentile over timesteps uses a histogram algorithm, to limit the amount of required memory. The default number of histogram bins is 101. That means the histogram algorithm is used, when the dataset has more than 101 time steps. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The histogram algorithm is implemented only for the Nearest Rank method.

## 2. Reference manual

This section gives a description of all operators. Related operators are grouped to modules. For easier description all single input files are named `ifile` or `ifile1`, `ifile2`, etc., and an arbitrary number of input files are named `ifiles`. All output files are named `ofile` or `ofile1`, `ofile2`, etc. Further the following notion is introduced:

- $i(t)$       Timestep  $t$  of `ifile`
- $i(t, x)$     Element number  $x$  of the field at timestep  $t$  of `ifile`
- $o(t)$       Timestep  $t$  of `ofile`
- $o(t, x)$     Element number  $x$  of the field at timestep  $t$  of `ofile`

## 2.1. Information

This section contains modules to print information about datasets. All operators print there results to standard output.

Here is a short overview of all operators in this section:

<b>info</b>	Dataset information listed by parameter identifier
<b>infn</b>	Dataset information listed by parameter name
<b>map</b>	Dataset information and simple map
<b>sinfo</b>	Short information listed by parameter identifier
<b>sinfn</b>	Short information listed by parameter name
<b>diff</b>	Compare two datasets listed by parameter id
<b>diffn</b>	Compare two datasets listed by parameter name
<b>npar</b>	Number of parameters
<b>nlevel</b>	Number of levels
<b>nyear</b>	Number of years
<b>nmon</b>	Number of months
<b>ndate</b>	Number of dates
<b>ntime</b>	Number of timesteps
<b>showformat</b>	Show file format
<b>showcode</b>	Show code numbers
<b>showname</b>	Show variable names
<b>showstdname</b>	Show standard names
<b>showlevel</b>	Show levels
<b>showltype</b>	Show GRIB level types
<b>showyear</b>	Show years
<b>showmon</b>	Show months
<b>showdate</b>	Show date information
<b>showtime</b>	Show time information
<b>showtimestamp</b>	Show timestamp
<b>pardes</b>	Parameter description
<b>griddes</b>	Grid description
<b>zaxisdes</b>	Z-axis description
<b>vct</b>	Vertical coordinate table

### 2.1.1. INFO - Information and simple statistics

#### Synopsis

```
<operator> ifiles
```

#### Description

This module writes information about the structure and contents of all input files to standard output. All input files need to have the same structure with the same variables on different timesteps. The information displayed depends on the chosen operator.

#### Operators

- info** Dataset information listed by parameter identifier  
Prints information and simple statistics for each field of all input datasets. For each field the operator prints one line with the following elements:
- Date and Time
  - Level, Gridsize and number of Missing values
  - Minimum, Mean and Maximum  
The mean value is computed without the use of area weights!
  - Parameter identifier
- infon** Dataset information listed by parameter name  
The same as operator [info](#) but using the name instead of the identifier to label the parameter.
- map** Dataset information and simple map  
Prints information, simple statistics and a map for each field of all input datasets. The map will be printed only for fields on a regular lon/lat grid.

#### Example

To print information and simple statistics for each field of a dataset use:

```
cdo infon ifile
```

This is an example result of a dataset with one 2D parameter over 12 timesteps:

-1 :	Date	Time	Level	Size	Miss	:	Minimum	Mean	Maximum	:	Name
1 :	1987-01-31	12:00:00	0	2048	1361	:	232.77	266.65	305.31	:	SST
2 :	1987-02-28	12:00:00	0	2048	1361	:	233.64	267.11	307.15	:	SST
3 :	1987-03-31	12:00:00	0	2048	1361	:	225.31	267.52	307.67	:	SST
4 :	1987-04-30	12:00:00	0	2048	1361	:	215.68	268.65	310.47	:	SST
5 :	1987-05-31	12:00:00	0	2048	1361	:	215.78	271.53	312.49	:	SST
6 :	1987-06-30	12:00:00	0	2048	1361	:	212.89	272.80	314.18	:	SST
7 :	1987-07-31	12:00:00	0	2048	1361	:	209.52	274.29	316.34	:	SST
8 :	1987-08-31	12:00:00	0	2048	1361	:	210.48	274.41	315.83	:	SST
9 :	1987-09-30	12:00:00	0	2048	1361	:	210.48	272.37	312.86	:	SST
10 :	1987-10-31	12:00:00	0	2048	1361	:	219.46	270.53	309.51	:	SST
11 :	1987-11-30	12:00:00	0	2048	1361	:	230.98	269.85	308.61	:	SST
12 :	1987-12-31	12:00:00	0	2048	1361	:	241.25	269.94	309.27	:	SST

## 2.1.2. SINFO - Short information

### Synopsis

```
<operator> ifiles
```

### Description

This module writes information about the structure of ifiles to standard output. ifiles is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. The information displayed depends on the chosen operator.

### Operators

- sinfo** Short information listed by parameter identifier  
Prints short information of a dataset. The information is divided into 4 sections. Section 1 prints one line per parameter with the following information:
- institute and source
  - timestep type
  - number of levels and z-axis number
  - horizontal grid size and number
  - data type
  - parameter identifier
- Section 2 and 3 gives a short overview of all grid and vertical coordinates. And the last section contains short information of the time coordinate.
- sinfon** Short information listed by parameter name  
The same as operator [sinfo](#) but using the name instead of the identifier to label the parameter.

### Example

To print short information of a dataset use:

```
cdo sinfon ifile
```

This is the result of an ECHAM5 dataset with 3 parameter over 12 timesteps:

```
-1 : Institut Source Ttype Levels Num Points Num Dtype : Name
 1 : MPIMET ECHAM5 constant 1 1 2048 1 F32 : GEOSP
 2 : MPIMET ECHAM5 instant 4 2 2048 1 F32 : T
 3 : MPIMET ECHAM5 instant 1 1 2048 1 F32 : TSURF
Grid coordinates :
 1 : gaussian : points=2048 (64x32) np=16
           longitude : 0 to 354.375 by 5.625 degrees_east circular
           latitude : 85.7606 to -85.7606 degrees_north
Vertical coordinates :
 1 : surface : levels=1
 2 : pressure : levels=4
           level : 92500 to 20000 Pa
Time coordinate : 12 steps
YYYY-MM-DD hh:mm:ss YYYY-MM-DD hh:mm:ss YYYY-MM-DD hh:mm:ss YYYY-MM-DD hh:mm:ss
1987-01-31 12:00:00 1987-02-28 12:00:00 1987-03-31 12:00:00 1987-04-30 12:00:00
1987-05-31 12:00:00 1987-06-30 12:00:00 1987-07-31 12:00:00 1987-08-31 12:00:00
1987-09-30 12:00:00 1987-10-31 12:00:00 1987-11-30 12:00:00 1987-12-31 12:00:00
```

### 2.1.3. DIFF - Compare two datasets field by field

#### Synopsis

```
<operator> ifile1 ifile2
```

#### Description

Compares the contents of two datasets field by field. The input datasets need to have the same structure and its fields need to have the same header information and dimensions.

#### Operators

**diff** Compare two datasets listed by parameter id  
Provides statistics on differences between two datasets. For each pair of fields the operator prints one line with the following information:

- Date and Time
- Level, Gridsize and number of Missing values
- Number of different values
- Occurrence of coefficient pairs with different signs (S)
- Occurrence of zero values (Z)
- Maxima of absolute difference of coefficient pairs
- Maxima of relative difference of non-zero coefficient pairs with equal signs
- Parameter identifier

$$Absdiff(t, x) = |i\_1(t, x) - i\_2(t, x)|$$

$$Reldiff(t, x) = \frac{|i\_1(t, x) - i\_2(t, x)|}{\max(|i\_1(t, x)|, |i\_2(t, x)|)}$$

**diffn** Compare two datasets listed by parameter name  
The same as operator [diff](#). Using the name instead of the identifier to label the parameter.

#### Example

To print the difference for each field of two datasets use:

```
cdo diffn ifile1 ifile2
```

This is an example result of two datasets with one 2D parameter over 12 timesteps:

	Date	Time	Level	Size	Miss	Diff	:	S	Z	Max_Absdiff	Max_Reldiff	:	Name
1	: 1987-01-31	12:00:00	0	2048	1361	273	:	F	F	0.00010681	4.1660e-07	:	SST
2	: 1987-02-28	12:00:00	0	2048	1361	309	:	F	F	6.1035e-05	2.3742e-07	:	SST
3	: 1987-03-31	12:00:00	0	2048	1361	292	:	F	F	7.6294e-05	3.3784e-07	:	SST
4	: 1987-04-30	12:00:00	0	2048	1361	183	:	F	F	7.6294e-05	3.5117e-07	:	SST
5	: 1987-05-31	12:00:00	0	2048	1361	207	:	F	F	0.00010681	4.0307e-07	:	SST
7	: 1987-07-31	12:00:00	0	2048	1361	317	:	F	F	9.1553e-05	3.5634e-07	:	SST
8	: 1987-08-31	12:00:00	0	2048	1361	219	:	F	F	7.6294e-05	2.8849e-07	:	SST
9	: 1987-09-30	12:00:00	0	2048	1361	188	:	F	F	7.6294e-05	3.6168e-07	:	SST
10	: 1987-10-31	12:00:00	0	2048	1361	297	:	F	F	9.1553e-05	3.5001e-07	:	SST
11	: 1987-11-30	12:00:00	0	2048	1361	234	:	F	F	6.1035e-05	2.3839e-07	:	SST
12	: 1987-12-31	12:00:00	0	2048	1361	267	:	F	F	9.3553e-05	3.7624e-07	:	SST
11 of 12 records differ													

## 2.1.4. NINFO - Print the number of parameters, levels or times

### Synopsis

```
<operator> ifile
```

### Description

This module prints the number of variables, levels or times of the input dataset.

### Operators

<b>npar</b>	Number of parameters Prints the number of parameters (variables).
<b>nlevel</b>	Number of levels Prints the number of levels for each variable.
<b>nyear</b>	Number of years Prints the number of different years.
<b>nmon</b>	Number of months Prints the number of different combinations of years and months.
<b>ndate</b>	Number of dates Prints the number of different dates.
<b>ntime</b>	Number of timesteps Prints the number of timesteps.

### Example

To print the number of parameters (variables) in a dataset use:

```
cdo npar ifile
```

To print the number of months in a dataset use:

```
cdo nmon ifile
```

## 2.1.5. SHOWINFO - Show variables, levels or times

### Synopsis

```
<operator> ifile
```

### Description

This module prints the format, variables, levels or times of the input dataset.

### Operators

<b>showformat</b>	Show file format Prints the file format of the input dataset.
<b>showcode</b>	Show code numbers Prints the code number of all variables.
<b>showname</b>	Show variable names Prints the name of all variables.
<b>showstdname</b>	Show standard names Prints the standard name of all variables.
<b>showlevel</b>	Show levels Prints all levels for each variable.
<b>showltype</b>	Show GRIB level types Prints the GRIB level type for all z-axes.
<b>showyear</b>	Show years Prints all years.
<b>showmon</b>	Show months Prints all months.
<b>showdate</b>	Show date information Prints date information of all timesteps (format YYYY-MM-DD).
<b>showtime</b>	Show time information Prints time information of all timesteps (format hh:mm:ss).
<b>showtimestamp</b>	Show timestamp Prints timestamp of all timesteps (format YYYY-MM-DDThh:mm:ss).

### Example

To print the code number of all variables in a dataset use:

```
cdo showcode ifile
```

This is an example result of a dataset with three variables:

```
129 130 139
```

To print all months in a dataset use:

```
cdo showmon ifile
```

This is an examples result of a dataset with an annual cycle:

```
1 2 3 4 5 6 7 8 9 10 11 12
```



## 2.1.6. FILEDES - Dataset description

### Synopsis

```
<operator> ifile
```

### Description

This module prints the description of the parameters, the grids, the z-axis or the vertical coordinate table.

### Operators

<b>pardes</b>	Parameter description Prints a table with a description of all variables. For each variable the operator prints one line listing the code, name, description and units.
<b>griddes</b>	Grid description Prints the description of all grids.
<b>zaxisdes</b>	Z-axis description Prints the description of all z-axes.
<b>vct</b>	Vertical coordinate table Prints the vertical coordinate table.

### Example

Assume all variables of the dataset are on a Gaussssian N16 grid. To print the grid description of this dataset use:

```
cdo griddes ifile
```

Result:

```
gridtype   : gaussian
gridsize   : 2048
xname      : lon
xlongname   : longitude
xunits     : degrees_east
yname      : lat
ylongname   : latitude
yunits     : degrees_north
xsize      : 64
ysize      : 32
xfirst     : 0
xinc       : 5.625
yvals      : 85.76058 80.26877 74.74454 69.21297 63.67863 58.1429 52.6065
              47.06964 41.53246 35.99507 30.4575 24.91992 19.38223 13.84448
              8.306702 2.768903 -2.768903 -8.306702 -13.84448 -19.38223
              -24.91992 -30.4575 -35.99507 -41.53246 -47.06964 -52.6065
              -58.1429 -63.67863 -69.21297 -74.74454 -80.26877 -85.76058
```

## 2.2. File operations

This section contains modules to perform operations on files.

Here is a short overview of all operators in this section:

<b>copy</b>	Copy datasets
<b>cat</b>	Concatenate datasets
<b>replace</b>	Replace variables
<b>duplicate</b>	Duplicates a dataset
<b>mergegrid</b>	Merge grid
<b>merge</b>	Merge datasets with different fields
<b>mergetime</b>	Merge datasets sorted by date and time
<b>splitcode</b>	Split code numbers
<b>splitparam</b>	Split parameter identifiers
<b>splitname</b>	Split variable names
<b>splitlevel</b>	Split levels
<b>splitgrid</b>	Split grids
<b>splitzaxis</b>	Split z-axes
<b>splittabnum</b>	Split parameter table numbers
<b>splithour</b>	Split hours
<b>splitday</b>	Split days
<b>splitseas</b>	Split seasons
<b>splityear</b>	Split years
<b>splityearmon</b>	Split in years and months
<b>splitmon</b>	Split months
<b>splitsel</b>	Split time selection
<b>distgrid</b>	Distribute horizontal grid
<b>collgrid</b>	Collect horizontal grid

### 2.2.1. COPY - Copy datasets

#### Synopsis

```
<operator> ifiles ofile
```

#### Description

This module contains operators to copy or concatenate datasets. `ifiles` is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps.

#### Operators

- |             |  |
|-------------|--|
| <b>copy</b> | Copy datasets<br>Copies all input datasets to <code>ofile</code> .   |
| <b>cat</b>  | Concatenate datasets<br>Concatenates all input datasets and appends the result to the end of <code>ofile</code> . If <code>ofile</code> does not exist it will be created. |

#### Example

To change the format of a dataset to netCDF use:

```
cdo -f nc copy ifile ofile.nc
```

Add the option `'-r'` to create a relative time axis, as is required for proper recognition by GrADS or Ferret:

```
cdo -r -f nc copy ifile ofile.nc
```

To concatenate 3 datasets with different timesteps of the same variables use:

```
cdo copy ifile1 ifile2 ifile3 ofile
```

If the output dataset already exists and you wish to extend it with more timesteps use:

```
cdo cat ifile1 ifile2 ifile3 ofile
```

### 2.2.2. REPLACE - Replace variables

#### Synopsis

```
replace ifile1 ifile2 ofile
```

#### Description

The replace operator replaces variables in ifile1 by variables from ifile2 and write the result to ofile. Both input datasets need to have the same number of timesteps.

#### Example

Assume the first input dataset ifile1 has three variables with the names geosp, t and tslm1 and the second input dataset ifile2 has only the variable tslm1. To replace the variable tslm1 in ifile1 by tslm1 from ifile2 use:

```
cdo replace ifile1 ifile2 ofile
```

### 2.2.3. DUPLICATE - Duplicates a dataset

#### Synopsis

```
duplicate[,ndup] ifile ofile
```

#### Description

This operator duplicates the contents of ifile and writes the result to ofile. The optional parameter sets the number of duplicates, the default is 1.

#### Parameter

*ndup*     INTEGER     Number of duplicates, default is 1.

### 2.2.4. MERGEGRID - Merge grid

#### Synopsis

```
mergegrid ifile1 ifile2 ofile
```

#### Description

Merges grid points of all variables from ifile2 to ifile1 and write the result to ofile. Only the non missing values of ifile2 will be used. The horizontal grid of ifile2 should be smaller or equal to the grid of ifile1 and the resolution must be the same. Only rectilinear grids are supported. Both input files need to have the same variables and the same number of timesteps.

## 2.2.5. MERGE - Merge datasets

### Synopsis

```
<operator> ifiles ofile
```

### Description

This module reads datasets from several input files, merges them and writes the resulting dataset to ofile.

### Operators

<b>merge</b>	<p>Merge datasets with different fields</p> <p>Merges time series of different fields from several input datasets. The number of fields per timestep written to ofile is the sum of the field numbers per timestep in all input datasets. The time series on all input datasets are required to have different fields and the same number of timesteps. The fields in each different input file either have to be different variables or different levels of the same variable. A mixture of different variables on different levels in different input files is not allowed.</p>
<b>mergetime</b>	<p>Merge datasets sorted by date and time</p> <p>Merges all timesteps of all input files sorted by date and time. All input files need to have the same structure with the same variables on different timesteps. After this operation every input timestep is in ofile and all timesteps are sorted by date and time.</p>

### Environment

SKIP_SAME_TIME	If set to 1, skips all consecutive timesteps with a double entry of the same timestamp.
----------------	---

### Example

Assume three datasets with the same number of timesteps and different variables in each dataset. To merge these datasets to a new dataset use:

```
cdo merge ifile1 ifile2 ifile3 ofile
```

Assume you split a 6 hourly dataset with [splithour](#). This produces four datasets, one for each hour. The following command merges them together:

```
cdo mergetime ifile1 ifile2 ifile3 ifile4 ofile
```

## 2.2.6. SPLIT - Split a dataset

### Synopsis

```
<operator>[,params] ifile obase
```

### Description

This module splits ifile into pieces. The output files will be named <obase><xxx><suffix> where suffix is the filename extension derived from the file format. xxx and the contents of the output files depends on the chosen operator. params is a comma separated list of processing parameters.

### Operators

<b>splitcode</b>	Split code numbers Splits a dataset into pieces, one for each different code number. xxx will have three digits with the code number.
<b>splitparam</b>	Split parameter identifiers Splits a dataset into pieces, one for each different parameter identifier. xxx will be a string with the parameter identifier.
<b>splitname</b>	Split variable names Splits a dataset into pieces, one for each variable name. xxx will be a string with the variable name.
<b>splitlevel</b>	Split levels Splits a dataset into pieces, one for each different level. xxx will have six digits with the level.
<b>splitgrid</b>	Split grids Splits a dataset into pieces, one for each different grid. xxx will have two digits with the grid number.
<b>splitzaxis</b>	Split z-axes Splits a dataset into pieces, one for each different z-axis. xxx will have two digits with the z-axis number.
<b>splittabnum</b>	Split parameter table numbers Splits a dataset into pieces, one for each GRIB1 parameter table number. xxx will have three digits with the GRIB1 parameter table number.

### Parameter

<i>swap</i>	STRING	Swap the position of obase and xxx in the output filename
<i>uuid=&lt;attname&gt;</i>	STRING	Add a UUID as global attribute <attname> to each output file

### Environment

CDO_FILE_SUFFIX	Set the default file suffix. This suffix will be added to the output file names instead of the filename extension derived from the file format. Set this variable to NULL to disable the adding of a file suffix.
-----------------	---

## Example

Assume an input GRIB1 dataset with three variables, e.g. code number 129, 130 and 139. To split this dataset into three pieces, one for each code number use:

```
cdo splitcode ifile code
```

Result of 'dir code\*':

```
code129.grb code130.grb code139.grb
```

## 2.2.7. SPLITTIME - Split timesteps of a dataset

### Synopsis

```
<operator> ifile obase
splitmon[,format] ifile obase
```

### Description

This module splits `ifile` into timesteps pieces. The output files will be named `<obase><xxx><suffix>` where `suffix` is the filename extension derived from the file format. `xxx` and the contents of the output files depends on the chosen operator.

### Operators

<b>splithour</b>	Split hours Splits a file into pieces, one for each different hour. <code>xxx</code> will have two digits with the hour.
<b>splitday</b>	Split days Splits a file into pieces, one for each different day. <code>xxx</code> will have two digits with the day.
<b>splitseas</b>	Split seasons Splits a file into pieces, one for each different season. <code>xxx</code> will have three characters with the season.
<b>splityear</b>	Split years Splits a file into pieces, one for each different year. <code>xxx</code> will have four digits with the year (YYYY).
<b>splityearmon</b>	Split in years and months Splits a file into pieces, one for each different year and month. <code>xxx</code> will have six digits with the year and month (YYYYMM).
<b>splitmon</b>	Split months Splits a file into pieces, one for each different month. <code>xxx</code> will have two digits with the month.

### Parameter

<i>format</i>	STRING	C-style format for <code>strftime()</code> (e.g. <code>%B</code> for the full month name)
---------------	--------	---

### Environment

<code>CDO_FILE_SUFFIX</code>	Set the default file suffix. This suffix will be added to the output file names instead of the filename extension derived from the file format. Set this variable to <code>NULL</code> to disable the adding of a file suffix.
------------------------------	--

### Example

Assume the input GRIB1 dataset has timesteps from January to December. To split each month with all variables into one separate file use:

```
cdo splitmon ifile mon
```

Result of `'dir mon*':`

mon01.grb	mon02.grb	mon03.grb	mon04.grb	mon05.grb	mon06.grb
mon07.grb	mon08.grb	mon09.grb	mon10.grb	mon11.grb	mon12.grb



## 2.2.8. SPLITSEL - Split selected timesteps

### Synopsis

```
splitsel,nsets[,noffset[,nskip]] ifile obase
```

### Description

This operator splits *ifile* into pieces, one for each adjacent sequence  $t_1, \dots, t_n$  of timesteps of the same selected time range. The output files will be named `<obase><nnnnnn><suffix>` where *nnnnnn* is the sequence number and *suffix* is the filename extension derived from the file format.

### Parameter

<i>nsets</i>	INTEGER	Number of input timesteps for each output file
<i>noffset</i>	INTEGER	Number of input timesteps skipped before the first timestep range (optional)
<i>nskip</i>	INTEGER	Number of input timesteps skipped between timestep ranges (optional)

### Environment

CDO_FILE_SUFFIX	Set the default file suffix. This suffix will be added to the output file names instead of the filename extension derived from the file format. Set this variable to NULL to disable the adding of a file suffix.
-----------------	---

## 2.2.9. DISTGRID - Distribute horizontal grid

### Synopsis

```
distgrid,nx[,ny] ifile obase
```

### Description

This operator distributes a dataset into smaller pieces. Each output file contains a different region of the horizontal source grid. A target grid region contains a structured longitude/latitude box of the source grid. Only rectilinear and curvilinear source grids are supported by this operator. The number of different regions can be specified with the parameter *nx* and *ny*. The output files will be named `<obase><xxx><suffix>` where suffix is the filename extension derived from the file format. xxx will have five digits with the number of the target region.

### Parameter

<i>nx</i>	INTEGER	Number of regions in x direction
<i>ny</i>	INTEGER	Number of regions in y direction [default: 1]

### Note

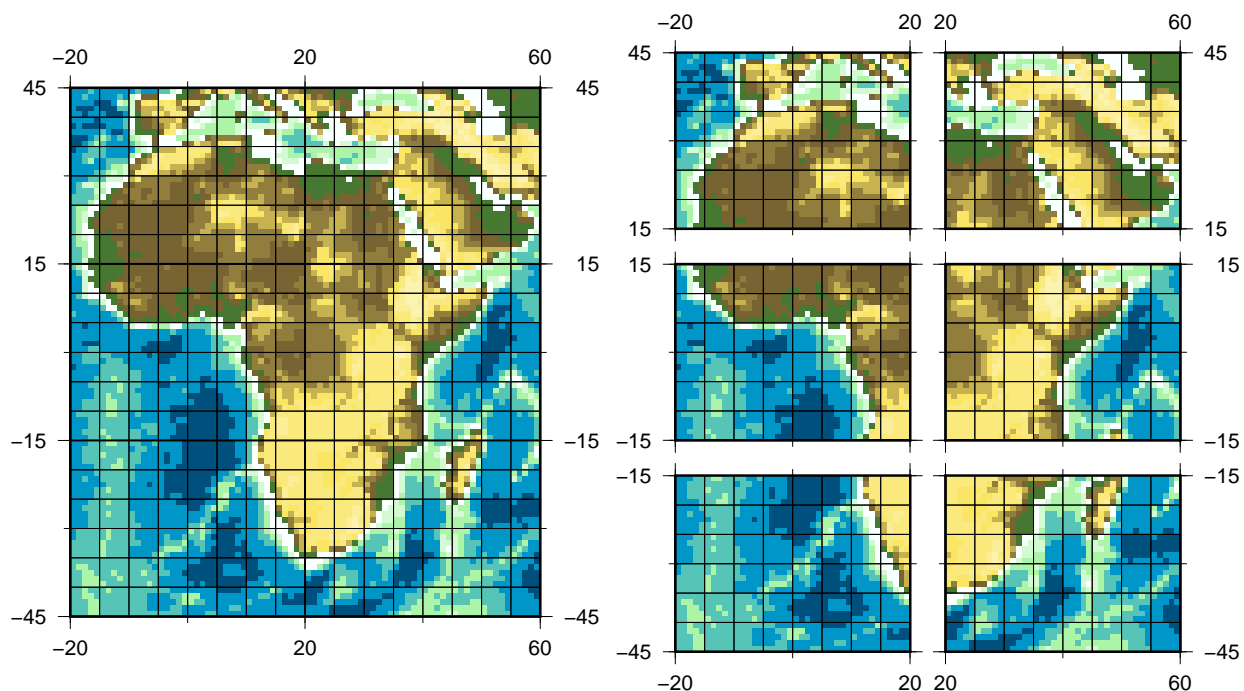
This operator needs to open all output files simultaneously. The maximum number of open files depends on the operating system!

### Example

Distribute a file into 6 smaller files, each output file receives one half of x and a third of y of the source grid:

```
cdo distgrid,2,3 ifile.nc obase
```

Below is a schematic illustration of this example:



On the left side is the data of the input file and on the right side is the data of the six output files.

## 2.2.10. COLLGRID - Collect horizontal grid

### Synopsis

```
collgrid[,nx[,names]] ifiles ofile
```

### Description

This operator collects the data of the input files to one output file. All input files need to have the same variables and the same number of timesteps on a different horizontal grid region. A source region must be a structured longitude/latitude grid box. The parameter *nx* needs to be specified only for curvilinear grids.

### Parameter

<i>nx</i>	INTEGER	Number of regions in x direction, only needed for curvilinear grids
<i>names</i>	STRING	Comma separated list of variable names [default: all variables]

### Note

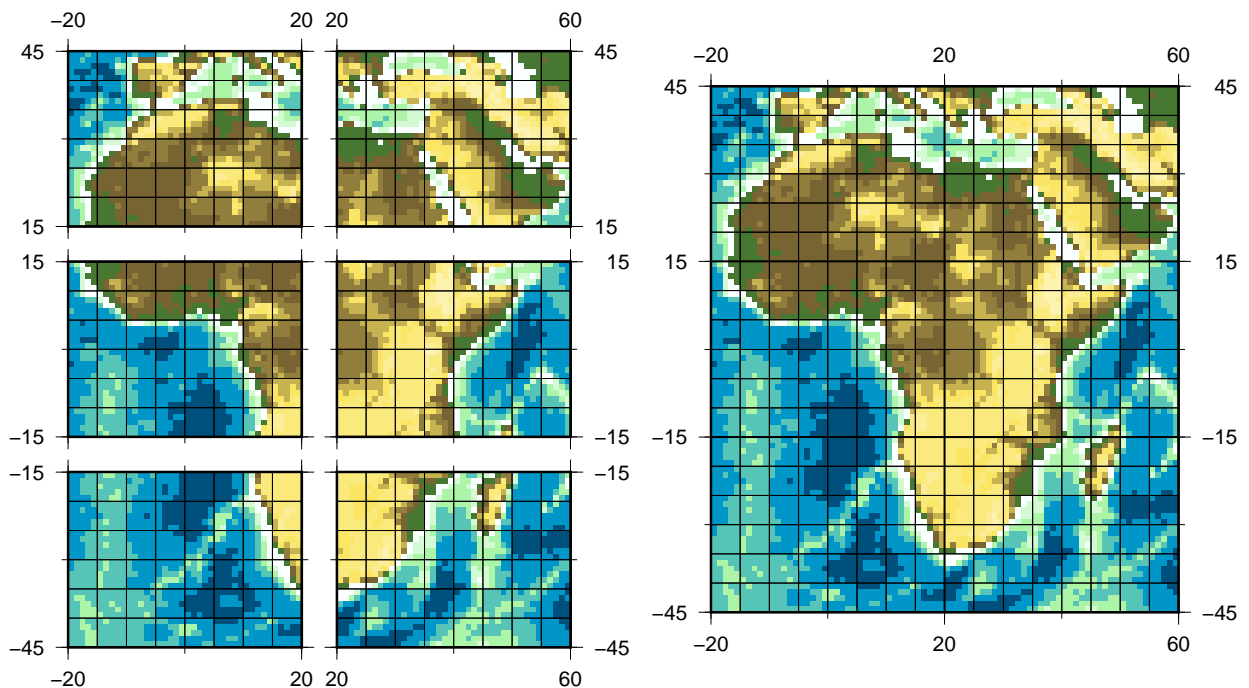
This operator needs to open all input files simultaneously. The maximum number of open files depends on the operating system!

### Example

Collect the horizontal grid of 6 input files. Each input file contains a lon/lat region of the target grid:

```
cdo collgrid ifile[1-6] ofile
```

Below is a schematic illustration of this example:



On the left side is the data of the six input files and on the right side is the collected data of the output file.

## 2.3. Selection

This section contains modules to select time steps, fields or a part of a field from a dataset.

Here is a short overview of all operators in this section:

<b>select</b>	Select fields
<b>delete</b>	Delete fields
<b>selparam</b>	Select parameters by identifier
<b>delparam</b>	Delete parameters by identifier
<b>selcode</b>	Select parameters by code number
<b>delcode</b>	Delete parameters by code number
<b>selname</b>	Select parameters by name
<b>delname</b>	Delete parameters by name
<b>selstdname</b>	Select parameters by standard name
<b>sellevel</b>	Select levels
<b>sellevidx</b>	Select levels by index
<b>selgrid</b>	Select grids
<b>selzaxis</b>	Select z-axes
<b>selzaxisname</b>	Select z-axes by name
<b>selltype</b>	Select GRIB level types
<b>seltabnum</b>	Select parameter table numbers
<b>sel timestep</b>	Select timesteps
<b>seltime</b>	Select times
<b>selhour</b>	Select hours
<b>selday</b>	Select days
<b>selmon</b>	Select months
<b>selyear</b>	Select years
<b>selseas</b>	Select seasons
<b>seldate</b>	Select dates
<b>selsmon</b>	Select single month
<b>sellonlatbox</b>	Select a longitude/latitude box
<b>selindexbox</b>	Select an index box

### 2.3.1. SELECT - Select fields

#### Synopsis

```
<operator>,params ifiles ofile
```

#### Description

This module selects some fields from *ifiles* and writes them to *ofile*. *ifiles* is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. The fields selected depends on the chosen parameters. Parameter is a comma separated list of key-value pairs. Wildcards can be used for string parameter.

#### Operators

<b>select</b>	Select fields Selects all fields with parameters in a user given list.
<b>delete</b>	Delete fields Deletes all fields with parameters in a user given list.

#### Parameter

<i>name</i>	STRING	Comma separated list of variable names.
<i>param</i>	STRING	Comma separated list of parameter identifiers.
<i>code</i>	INTEGER	Comma separated list of code numbers.
<i>ltype</i>	INTEGER	Comma separated list of GRIB level types.
<i>levidx</i>	INTEGER	Comma separated list of index of levels.
<i>level</i>	FLOAT	Comma separated list of vertical levels.
<i>date</i>	STRING	Comma separated list of dates (format YYYY-MM-DDThh:mm:ss).
<i>startdate</i>	STRING	Start date (format YYYY-MM-DDThh:mm:ss).
<i>enddate</i>	STRING	End date (format YYYY-MM-DDThh:mm:ss).
<i>minute</i>	INTEGER	Comma separated list of minutes.
<i>hour</i>	INTEGER	Comma separated list of hours.
<i>day</i>	INTEGER	Comma separated list of days.
<i>month</i>	INTEGER	Comma separated list of months.
<i>year</i>	INTEGER	Comma separated list of years.
<i>timestep</i>	INTEGER	Comma separated list of timesteps. Negative values selects timesteps from the end (netCDF only).
<i>timestep_of_year</i>	INTEGER	Comma separated list of timesteps of year.

#### Example

Assume you have 3 inputfiles. Each inputfile contains the same variables for a different time period. To select the variable T,U and V on the levels 200, 500 and 850 from all 3 input files, use:

```
cdo select,name=T,U,V,level=200,500,850 ifile1 ifile2 ifile3 ofile
```

## 2.3.2. SELVAR - Select fields

### Synopsis

```

<operator> ,params ifile ofile
selcode, codes ifile ofile
delcode, codes ifile ofile
selname, names ifile ofile
delname, names ifile ofile
selstdname, stdnames ifile ofile
sellevel, levels ifile ofile
sellevidx, levidx ifile ofile
selgrid, grids ifile ofile
selzaxis, zaxes ifile ofile
selzaxisname, zaxisnames ifile ofile
selltype, ltypes ifile ofile
seltabnum, tabnums ifile ofile

```

### Description

This module selects some fields from ifile and writes them to ofile. The fields selected depends on the chosen operator and the parameters.

### Operators

<b>selparam</b>	Select parameters by identifier Selects all fields with parameter identifiers in a user given list.
<b>delparam</b>	Delete parameters by identifier Deletes all fields with parameter identifiers in a user given list.
<b>selcode</b>	Select parameters by code number Selects all fields with code numbers in a user given list.
<b>delcode</b>	Delete parameters by code number Deletes all fields with code numbers in a user given list.
<b>selname</b>	Select parameters by name Selects all fields with parameter names in a user given list.
<b>delname</b>	Delete parameters by name Deletes all fields with parameter names in a user given list.
<b>selstdname</b>	Select parameters by standard name Selects all fields with standard names in a user given list.
<b>sellevel</b>	Select levels Selects all fields with levels in a user given list.
<b>sellevidx</b>	Select levels by index Selects all fields with index of levels in a user given list.
<b>selgrid</b>	Select grids Selects all fields with grids in a user given list.

<b>selzaxis</b>	Select z-axes Selects all fields with z-axes in a user given list.
<b>selzaxisname</b>	Select z-axes by name Selects all fields with z-axis names in a user given list.
<b>selltype</b>	Select GRIB level types Selects all fields with GRIB level type in a user given list.
<b>seltabnum</b>	Select parameter table numbers Selects all fields with parameter table numbers in a user given list.

## Parameter

<i>params</i>	INTEGER	Comma separated list of parameter identifiers
<i>codes</i>	INTEGER	Comma separated list of code numbers
<i>names</i>	STRING	Comma separated list of variable names
<i>stdnames</i>	STRING	Comma separated list of standard names
<i>levels</i>	FLOAT	Comma separated list of vertical levels
<i>levidx</i>	INTEGER	Comma separated list of index of levels
<i>ltypes</i>	INTEGER	Comma separated list of GRIB level types
<i>grids</i>	STRING	Comma separated list of grid names or numbers
<i>zaxes</i>	STRING	Comma separated list of z-axis types or numbers
<i>zaxisnames</i>	STRING	Comma separated list of z-axis names
<i>tabnums</i>	INTEGER	Comma separated list of parameter table numbers

## Example

Assume an input dataset has three variables with the code numbers 129, 130 and 139. To select the variables with the code number 129 and 139 use:

```
cdo selcode,129,139 ifile ofile
```

You can also select the code number 129 and 139 by deleting the code number 130 with:

```
cdo delcode,130 ifile ofile
```

### 2.3.3. SELTIME - Select timesteps

#### Synopsis

```

sel timestep,timesteps ifile ofile
sel time,times ifile ofile
sel hour,hours ifile ofile
sel day,days ifile ofile
sel mon,months ifile ofile
sel year,years ifile ofile
sel seas,seasons ifile ofile
sel date,date1[,date2] ifile ofile
sel mon,month[,nts1[,nts2]] ifile ofile

```

#### Description

This module selects user specified timesteps from ifile and writes them to ofile. The timesteps selected depends on the chosen operator and the parameters.

#### Operators

<b>sel timestep</b>	Select timesteps Selects all timesteps with a timestep in a user given list.
<b>sel time</b>	Select times Selects all timesteps with a time in a user given list.
<b>sel hour</b>	Select hours Selects all timesteps with a hour in a user given list.
<b>sel day</b>	Select days Selects all timesteps with a day in a user given list.
<b>sel mon</b>	Select months Selects all timesteps with a month in a user given list.
<b>sel year</b>	Select years Selects all timesteps with a year in a user given list.
<b>sel seas</b>	Select seasons Selects all timesteps with a month of a season in a user given list.
<b>sel date</b>	Select dates Selects all timesteps with a date in a user given range.
<b>sel mon</b>	Select single month Selects a month and optional an arbitrary number of timesteps before and after this month.

#### Parameter

<i>timesteps</i>	INTEGER	Comma separated list of timesteps. Negative values selects timesteps from the end (netCDF only).
<i>times</i>	STRING	Comma separated list of times (format hh:mm:ss).
<i>hours</i>	INTEGER	Comma separated list of hours.



---

<i>days</i>	INTEGER	Comma separated list of days.
<i>months</i>	INTEGER	Comma separated list of months.
<i>years</i>	INTEGER	Comma separated list of years.
<i>seasons</i>	STRING	Comma separated list of seasons (DJF, MAM, JJA, SON).
<i>date1</i>	STRING	Start date (format YYYY-MM-DDThh:mm:ss).
<i>date2</i>	STRING	End date (format YYYY-MM-DDThh:mm:ss) [default: <i>date1</i> ].
<i>nts1</i>	INTEGER	Number of timesteps before the selected month [default: 0].
<i>nts2</i>	INTEGER	Number of timesteps after the selected month [default: <i>nts1</i> ].

### 2.3.4. SELBOX - Select a box of a field

#### Synopsis

```
sellonlatbox,lon1,lon2,lat1,lat2 ifile ofile
```

```
selindexbox,idx1,idx2,idy1,idy2 ifile ofile
```

#### Description

Selects a box of the rectangularly understood field. All input fields need to have the same horizontal grid.

#### Operators

<b>sellonlatbox</b>	Select a longitude/latitude box Selects a regular longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box. Considered are only those grid cells with the grid center inside the lon/lat box. For rotated lon/lat grids the parameter needs to be rotated coordinates.
<b>selindexbox</b>	Select an index box Selects an index box. The user has to give the indexes of the edges of the box. The index of the left edge may be greater then that of the right edge.

#### Parameter

<i>lon1</i>	FLOAT	Western longitude
<i>lon2</i>	FLOAT	Eastern longitude
<i>lat1</i>	FLOAT	Southern or northern latitude
<i>lat2</i>	FLOAT	Northern or southern latitude
<i>idx1</i>	INTEGER	Index of first longitude
<i>idx2</i>	INTEGER	Index of last longitude
<i>idy1</i>	INTEGER	Index of first latitude
<i>idy2</i>	INTEGER	Index of last latitude

#### Example

To select the region with the longitudes from 120E to 90W and latitudes from 20N to 20S from all input fields use:

```
cdo sellonlatbox,120,-90,20,-20 ifile ofile
```

If the input dataset has fields on a Gaussian N16 grid, the same box can be selected with [selindexbox](#) by:

```
cdo selindexbox,23,48,13,20 ifile ofile
```

## 2.4. Conditional selection

This section contains modules to conditional select field elements. The fields in the first input file are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false".

Here is a short overview of all operators in this section:

<b>ifthen</b>	If then
<b>ifnotthen</b>	If not then
<b>ifthenelse</b>	If then else
<b>ifthenc</b>	If then constant
<b>ifnotthenc</b>	If not then constant

### 2.4.1. COND - Conditional select one field

#### Synopsis

```
<operator> ifile1 ifile2 ofile
```

#### Description

This module selects field elements from ifile2 with respect to ifile1 and writes them to ofile. The fields in ifile1 are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false". The number of fields in ifile1 has either to be the same as in ifile2 or the same as in one timestep of ifile2 or only one. The fields in ofile inherit the meta data from ifile2.

#### Operators

**ifthen**      If then

$$o(t, x) = \begin{cases} i_2(t, x) & \text{if } i_1([t, ]x) \neq 0 \quad \wedge \quad i_1([t, ]x) \neq \text{miss} \\ \text{miss} & \text{if } i_1([t, ]x) = 0 \quad \vee \quad i_1([t, ]x) = \text{miss} \end{cases}$$

**ifnotthen**    If not then

$$o(t, x) = \begin{cases} i_2(t, x) & \text{if } i_1([t, ]x) = 0 \quad \wedge \quad i_1([t, ]x) \neq \text{miss} \\ \text{miss} & \text{if } i_1([t, ]x) \neq 0 \quad \vee \quad i_1([t, ]x) = \text{miss} \end{cases}$$

#### Example

To select all field elements of ifile2 if the corresponding field element of ifile1 is greater than 0 use:

```
cdo ifthen ifile1 ifile2 ofile
```

### 2.4.2. COND2 - Conditional select two fields

#### Synopsis

```
ifthenelse ifile1 ifile2 ifile3 ofile
```

#### Description

This operator selects field elements from ifile2 or ifile3 with respect to ifile1 and writes them to ofile. The fields in ifile1 are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false". The number of fields in ifile1 has either to be the same as in ifile2 or the same as in one timestep of ifile2 or only one. ifile2 and ifile3 need to have the same number of fields. The fields in ofile inherit the meta data from ifile2.

$$o(t, x) = \begin{cases} i_2(t, x) & \text{if } i_1([t, ]x) \neq 0 \quad \wedge \quad i_1([t, ]x) \neq \text{miss} \\ i_3(t, x) & \text{if } i_1([t, ]x) = 0 \quad \wedge \quad i_1([t, ]x) \neq \text{miss} \\ \text{miss} & \text{if } i_1([t, ]x) = \text{miss} \end{cases}$$

#### Example

To select all field elements of ifile2 if the corresponding field element of ifile1 is greater than 0 and from ifile3 otherwise use:

```
cdo ifthenelse ifile1 ifile2 ifile3 ofile
```

### 2.4.3. CONDC - Conditional select a constant

#### Synopsis

```
<operator>,c ifile ofile
```

#### Description

This module creates fields with a constant value or missing value. The fields in `ifile` are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false".

#### Operators

<b>ifthenc</b>	If then constant
	$o(t, x) = \begin{cases} c & \text{if } i(t, x) \neq 0 \wedge i(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = 0 \vee i(t, x) = \text{miss} \end{cases}$
<b>ifnotthenc</b>	If not then constant
	$o(t, x) = \begin{cases} c & \text{if } i(t, x) = 0 \wedge i(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) \neq 0 \vee i(t, x) = \text{miss} \end{cases}$

#### Parameter

<code>c</code>	Float	Constant
----------------	-------	----------

#### Example

To create fields with the constant value 7 if the corresponding field element of `ifile` is greater than 0 use:

```
cdo ifthenc,7 ifile ofile
```

## 2.5. Comparison

This section contains modules to compare datasets. The resulting field is a mask containing 1 if the comparison is true and 0 if not.

Here is a short overview of all operators in this section:

<code>eq</code>	Equal
<code>ne</code>	Not equal
<code>le</code>	Less equal
<code>lt</code>	Less than
<code>ge</code>	Greater equal
<code>gt</code>	Greater than
<code>eqc</code>	Equal constant
<code>nec</code>	Not equal constant
<code>lec</code>	Less equal constant
<code>ltc</code>	Less than constant
<code>gec</code>	Greater equal constant
<code>gtc</code>	Greater than constant

## 2.5.1. COMP - Comparison of two fields

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module compares two datasets field by field. The resulting field is a mask containing 1 if the comparison is true and 0 if not. The number of fields in ifile1 should be the same as in ifile2. One of the input files can contain only one timestep or one field. The fields in ofile inherit the meta data from ifile1 or ifile2. The type of comparison depends on the chosen operator.

### Operators

<b>eq</b>	Equal	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) = i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \neq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$
<b>ne</b>	Not equal	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \neq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) = i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$
<b>le</b>	Less equal	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \leq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) > i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$
<b>lt</b>	Less than	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) < i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \geq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$
<b>ge</b>	Greater equal	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \geq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) < i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$
<b>gt</b>	Greater than	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) > i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \leq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$

### Example

To create a mask containing 1 if the elements of two fields are the same and 0 if the elements are different use:

```
cdo eq ifile1 ifile2 ofile
```

## 2.5.2. COMPC - Comparison of a field with a constant

### Synopsis

`<operator>,c ifile ofile`

### Description

This module compares all fields of a dataset with a constant. The resulting field is a mask containing 1 if the comparison is true and 0 if not. The type of comparison depends on the chosen operator.

### Operators

<b>eqc</b>	Equal constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) = c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) \neq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>nec</b>	Not equal constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \neq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) = c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>lec</b>	Less equal constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \leq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) > c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>ltc</b>	Less than constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) < c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) \geq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>gec</b>	Greater equal constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \geq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) < c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>gtc</b>	Greater than constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) > c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) \leq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	

### Parameter

`c`    `FLOAT`    Constant

### Example

To create a mask containing 1 if the field element is greater than 273.15 and 0 if not use:

```
cdo gtc,273.15 ifile ofile
```



## 2.6. Modification

This section contains modules to modify the metadata, fields or part of a field in a dataset.

Here is a short overview of all operators in this section:

<b>setpartabp</b>	Set parameter table
<b>setpartabn</b>	Set parameter table
<b>setpartab</b>	Set parameter table
<b>setcode</b>	Set code number
<b>setparam</b>	Set parameter identifier
<b>setname</b>	Set variable name
<b>setunit</b>	Set variable unit
<b>setlevel</b>	Set level
<b>setltype</b>	Set GRIB level type
<b>setdate</b>	Set date
<b>settime</b>	Set time of the day
<b>setday</b>	Set day
<b>setmon</b>	Set month
<b>setyear</b>	Set year
<b>settunits</b>	Set time units
<b>settaxis</b>	Set time axis
<b>setreftime</b>	Set reference time
<b>setcalendar</b>	Set calendar
<b>shifttime</b>	Shift timesteps
<b>chcode</b>	Change code number
<b>chparam</b>	Change parameter identifier
<b>chname</b>	Change variable name
<b>chunit</b>	Change variable unit
<b>chlevel</b>	Change level
<b>chlevelc</b>	Change level of one code
<b>chlevelv</b>	Change level of one variable
<b>setgrid</b>	Set grid
<b>setgridtype</b>	Set grid type
<b>setgridarea</b>	Set grid cell area
<b>setzaxis</b>	Set z-axis
<b>genlevelbounds</b>	Generate level bounds
<b>setgatt</b>	Set global attribute
<b>setgatts</b>	Set global attributes
<b>invertlat</b>	Invert latitudes
<b>invertlev</b>	Invert levels
<b>maskregion</b>	Mask regions
<b>masklonlatbox</b>	Mask a longitude/latitude box
<b>maskindexbox</b>	Mask an index box
<b>setclonlatbox</b>	Set a longitude/latitude box to constant
<b>setcindexbox</b>	Set an index box to constant

<b>enlarge</b>	Enlarge fields
<b>setmissval</b>	Set a new missing value
<b>setctomiss</b>	Set constant to missing value
<b>setmisstoc</b>	Set missing value to constant
<b>setrtomiss</b>	Set range to missing value
<b>setvrangle</b>	Set valid range
<b>setmisstonn</b>	Set missing value to nearest neighbor
<b>setmisstodis</b>	Set missing value to distance-weighted average

## 2.6.1. SETPARTAB - Set parameter table

### Synopsis

```
<operator>,table[,convert] ifile ofile
```

### Description

This module transforms data and metadata of ifile via a parameter table and writes the result to ofile. A parameter table is an ASCII formatted file with a set of parameter entries for each variable. Each new set have to start with "&parameter" and to end with "/".

The following parameter table entries are supported:

Entry	Type	Description
name	WORD	Name of the variable
out_name	WORD	New name of the variable
param	WORD	Parameter identifier (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]])
out_param	WORD	New parameter identifier
type	WORD	Data type (real or double)
standard_name	WORD	As defined in the CF standard name table
long_name	STRING	Describing the variable
units	STRING	Specifying the units for the variable
comment	STRING	Information concerning the variable
cell_methods	STRING	Information concerning calculation of means or climatologies
cell_measures	STRING	Indicates the names of the variables containing cell areas and volumes
missing_value	FLOAT	Specifying how missing data will be identified
valid_min	FLOAT	Minimum valid value
valid_max	FLOAT	Maximum valid value
ok_min_mean_abs	FLOAT	Minimum absolute mean
ok_max_mean_abs	FLOAT	Maximum absolute mean
factor	FLOAT	Scale factor
delete	INTEGER	Set to 1 to delete variable
convert	INTEGER	Set to 1 to convert the unit if necessary

The search key for the variable depends on the operator. Use [setpartabn](#) to search variables by the name. This is typically used for netCDF datasets. The operator [setpartabp](#) searches variables by the parameter ID.

### Operators

**setpartabp**     Set parameter table  
                   Search variables by the parameter identifier.

**setpartabn**     Set parameter table  
                   Search variables by name.

### Parameter

*table*            STRING        Parameter table file or name  
*convert*          STRING        Converts the units if necessary

## Example

Here is an example of a parameter table for one variable:

```
prompt> cat mypartab
&parameter
  name           = t
  out_name       = ta
  standard_name  = air_temperature
  convert        = 1
  units          = "K"
  missing_value  = 1e+20
  valid_min      = 157.1
  valid_max      = 336.3
/
```

To apply this parameter table to a dataset use:

```
cdo setpartabn,mypartab,convert ifile ofile
```

This command renames the variable **t** to **ta**. The standard name of this variable is set to **air\_temperature** and the unit is set to **[K]** (converts the unit if necessary). The missing value will be set to **1e+20**. In addition it will be checked whether the values of the variable are in the range of **157.1** to **336.3**.

## 2.6.2. SET - Set field info

### Synopsis

```

setpartab,table ifile ofile
setcode,code ifile ofile
setparam,param ifile ofile
setname,name ifile ofile
setunit,unit ifile ofile
setlevel,level ifile ofile
setltype,ltype ifile ofile

```

### Description

This module sets some field information. Depending on the chosen operator the parameter table, code number, parameter identifier, variable name or level is set.

### Operators

<b>setpartab</b>	Set parameter table Sets the parameter table for all variables.
<b>setcode</b>	Set code number Sets the code number for all variables to the same given value.
<b>setparam</b>	Set parameter identifier Sets the parameter identifier of the first variable.
<b>setname</b>	Set variable name Sets the name of the first variable.
<b>setunit</b>	Set variable unit Sets the unit of the first variable.
<b>setlevel</b>	Set level Sets the first level of all variables.
<b>setltype</b>	Set GRIB level type Sets the GRIB level type of all variables.

### Parameter

<i>table</i>	STRING	Parameter table file or name
<i>code</i>	INTEGER	Code number
<i>param</i>	STRING	Parameter identifier (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]])
<i>name</i>	STRING	Variable name
<i>level</i>	FLOAT	New level
<i>ltype</i>	INTEGER	GRIB level type

### 2.6.3. SETTIME - Set time

#### Synopsis

```

setdate,date ifile ofile
settime,time ifile ofile
setday,day ifile ofile
setmon,month ifile ofile
setyear,year ifile ofile
settunits,units ifile ofile
settaxis,date,time[inc] ifile ofile
setreftime,date,time[units] ifile ofile
setcalendar,calendar ifile ofile
shifttime,sval ifile ofile

```

#### Description

This module sets the time axis or part of the time axis. Which part of the time axis is overwritten depends on the chosen operator.

#### Operators

<b>setdate</b>	Set date Sets the date in every timestep to the same given value.
<b>settime</b>	Set time of the day Sets the time in every timestep to the same given value.
<b>setday</b>	Set day Sets the day in every timestep to the same given value.
<b>setmon</b>	Set month Sets the month in every timestep to the same given value.
<b>setyear</b>	Set year Sets the year in every timestep to the same given value.
<b>settunits</b>	Set time units Sets the base units of a relative time axis.
<b>settaxis</b>	Set time axis Sets the time axis.
<b>setreftime</b>	Set reference time Sets the reference time of a relative time axis.
<b>setcalendar</b>	Set calendar Sets the calendar of a relative time axis.
<b>shifttime</b>	Shift timesteps Shifts all timesteps by the parameter sval.

**Parameter**

<i>day</i>	INTEGER	Value of the new day
<i>month</i>	INTEGER	Value of the new month
<i>year</i>	INTEGER	Value of the new year
<i>units</i>	STRING	Base units of the time axis (seconds, minutes, hours, days, months, years)
<i>date</i>	STRING	Date (format: YYYY-MM-DD)
<i>time</i>	STRING	Time (format: hh:mm:ss)
<i>inc</i>	STRING	Optional increment (seconds, minutes, hours, days, months, years) [default: 0hour]
<i>calendar</i>	STRING	Calendar (standard, proleptic_gregorian, 360_day, 365_day, 366_day)
<i>sval</i>	STRING	Shift value (e.g. -3hour)

**Example**

To set the time axis to 1987-01-16 12:00:00 with an increment of one month for each timestep use:

```
cdo settaxis,1987-01-16,12:00:00,1mon ifile ofile
```

Result of 'cdo showdate ofile' for a dataset with 12 timesteps:

```
1987-01-16 1987-02-16 1987-03-16 1987-04-16 1987-05-16 1987-06-16 \
1987-07-16 1987-08-16 1987-09-16 1987-10-16 1987-11-16 1987-12-16
```

To shift this time axis by -15 days use:

```
cdo shifttime,-15days ifile ofile
```

Result of 'cdo showdate ofile':

```
1987-01-01 1987-02-01 1987-03-01 1987-04-01 1987-05-01 1987-06-01 \
1987-07-01 1987-08-01 1987-09-01 1987-10-01 1987-11-01 1987-12-01
```

## 2.6.4. CHANGE - Change field header

### Synopsis

```

chcode,oldcode,newcode[...] ifile ofile
chparam,oldparam,newparam,... ifile ofile
chname,oldname,newname,... ifile ofile
chunit,oldunit,newunit,... ifile ofile
chlevel,oldlev,newlev,... ifile ofile
chlevelc,code,oldlev,newlev ifile ofile
chlevelv,name,oldlev,newlev ifile ofile

```

### Description

This module reads fields from ifile, changes some header values and writes the results to ofile. The kind of changes depends on the chosen operator.

### Operators

<b>chcode</b>	Change code number Changes some user given code numbers to new user given values.
<b>chparam</b>	Change parameter identifier Changes some user given parameter identifiers to new user given values.
<b>chname</b>	Change variable name Changes some user given variable names to new user given names.
<b>chunit</b>	Change variable unit Changes some user given variable units to new user given units.
<b>chlevel</b>	Change level Changes some user given levels to new user given values.
<b>chlevelc</b>	Change level of one code Changes one level of a user given code number.
<b>chlevelv</b>	Change level of one variable Changes one level of a user given variable name.

### Parameter

<i>code</i>	INTEGER	Code number
<i>oldcode,newcode,...</i>	INTEGER	Pairs of old and new code numbers
<i>oldparam,newparam,...</i>	STRING	Pairs of old and new parameter identifiers
<i>name</i>	STRING	Variable name
<i>oldname,newname,...</i>	STRING	Pairs of old and new variable names
<i>oldlev</i>	FLOAT	Old level
<i>newlev</i>	FLOAT	New level
<i>oldlev,newlev,...</i>	FLOAT	Pairs of old and new levels

### Example

To change the code number 98 to 179 and 99 to 211 use:

```
cdo chcode,98,179,99,211 ifile ofile
```



## 2.6.5. SETGRID - Set grid information

### Synopsis

```
setgrid,grid ifile ofile
setgridtype,gridtype ifile ofile
setgridarea,gridarea ifile ofile
```

### Description

This module modifies the metadata of the horizontal grid. Depending on the chosen operator a new grid description is set, the coordinates are converted or the grid cell area is added.

### Operators

<b>setgrid</b>	Set grid Sets a new grid description. The input fields need to have the same grid size as the size of the target grid description.										
<b>setgridtype</b>	Set grid type Sets the grid type of all input fields. The following grid types are available: <table> <tr> <td>curvilinear</td><td>Converts a regular grid to a curvilinear grid</td></tr> <tr> <td>unstructured</td><td>Converts a regular or curvilinear grid to an unstructured grid</td></tr> <tr> <td>dereference</td><td>Dereference a reference to a grid</td></tr> <tr> <td>regular</td><td>Converts a reduced Gaussian grid to a regular Gaussian grid</td></tr> <tr> <td>lonlat</td><td>Converts a regular lonlat grid stored as a curvilinear grid back to a lonlat grid</td></tr> </table>	curvilinear	Converts a regular grid to a curvilinear grid	unstructured	Converts a regular or curvilinear grid to an unstructured grid	dereference	Dereference a reference to a grid	regular	Converts a reduced Gaussian grid to a regular Gaussian grid	lonlat	Converts a regular lonlat grid stored as a curvilinear grid back to a lonlat grid
curvilinear	Converts a regular grid to a curvilinear grid										
unstructured	Converts a regular or curvilinear grid to an unstructured grid										
dereference	Dereference a reference to a grid										
regular	Converts a reduced Gaussian grid to a regular Gaussian grid										
lonlat	Converts a regular lonlat grid stored as a curvilinear grid back to a lonlat grid										
<b>setgridarea</b>	Set grid cell area Sets the grid cell area. The parameter <i>gridarea</i> is the path to a data file, the first field is used as grid cell area. The input fields need to have the same grid size as the grid cell area. The grid cell area is used to compute the weights of each grid cell if needed by an operator, e.g. for <a href="#">fldmean</a> .										

### Parameter

<i>grid</i>	STRING	Grid description file or name
<i>gridtype</i>	STRING	Grid type (curvilinear, unstructured, regular, lonlat or dereference)
<i>gridarea</i>	STRING	Data file, the first field is used as grid cell area

### Example

Assuming a dataset has fields on a grid with 2048 elements without or with wrong grid description. To set the grid description of all input fields to a Gaussian N32 grid (8192 gridpoints) use:

```
cdo setgrid,n32 ifile ofile
```

## 2.6.6. SETZAXIS - Set z-axis information

### Synopsis

```
setzaxis,zaxis ifile ofile  
genlevelbounds[,zbot[,ztop]] ifile ofile
```

### Description

This module modifies the metadata of the vertical grid.

### Operators

<b>setzaxis</b>	Set z-axis This operator sets the z-axis description of all variables with the same number of level as the new z-axis.
<b>genlevelbounds</b>	Generate level bounds Generates the layer bounds of the z-axis.

### Parameter

<i>zaxis</i>	STRING	Z-axis description file or name of the target z-axis
<i>zbot</i>	FLOAT z-axis.	Specifying the bottom of the vertical column. Must have the same units as
<i>ztop</i>	FLOAT	Specifying the top of the vertical column. Must have the same units as z-axis.

## 2.6.7. SETGATT - Set global attribute

### Synopsis

```
setgatt,attname,attstring ifile ofile
```

```
setgatts,attfile ifile ofile
```

### Description

This module sets global text attributes of a dataset. Depending on the chosen operator the attributes are read from a file or can be specified by a parameter.

### Operators

<b>setgatt</b>	Set global attribute Sets one user defined global text attribute.
<b>setgatts</b>	Set global attributes Sets user defined global text attributes. The name and text of the global attributes are read from a file.

### Parameter

<i>attname,attstring</i>	STRING	Name and text of the global attribute (without spaces!)
<i>attfile</i>	STRING	File name which contains global text attributes

### Note

Besides netCDF none of the supported data formats supports global attributes.

### Example

To set the global text attribute "myatt" to "myattcontents" in a netCDF file use:

```
cdo setgatt,myatt,myattcontents ifile ofile
```

Result of 'ncdump -h ofile':

```
netcdf ofile {  
  dimensions: ...  
  
  variables: ...  
  
  // global attributes:  
      :myatt = "myattcontents" ;  
}
```

### 2.6.8. INVERT - Invert latitudes

#### Synopsis

```
invertlat ifile ofile
```

#### Description

This operator inverts the latitudes of all fields on a rectilinear grid.

#### Example

To invert the latitudes of a 2D field from N->S to S->N use:

```
cdo invertlat ifile ofile
```

### 2.6.9. INVERTLEV - Invert levels

#### Synopsis

```
invertlev ifile ofile
```

#### Description

This operator inverts the levels of all 3D variables.

## 2.6.10. MASKREGION - Mask regions

### Synopsis

```
maskregion,regions ifile ofile
```

### Description

Masks different regions of fields with a regular lon/lat grid. The elements inside a region are untouched, the elements outside are set to missing value. Considered are only those grid cells with the grid center inside the regions. All input fields must have the same horizontal grid. The user has to give ASCII formatted files with different regions. A region is defined by a polygon. Each line of a polygon description file contains the longitude and latitude of one point. Each polygon description file can contain one or more polygons separated by a line with the character &.

### Parameter

<i>regions</i>	STRING	Comma separated list of ASCII formatted files with different regions
----------------	--------	--

### Example

To mask the region with the longitudes from 120E to 90W and latitudes from 20N to 20S on all input fields use:

```
cdo maskregion,myregion ifile ofile
```

For this example the polygon description file myregion should contain the following four coordinates:

```
120 20
120 -20
270 -20
270 20
```

## 2.6.11. MASKBOX - Mask a box

### Synopsis

```
masklonlatbox,lon1,lon2,lat1,lat2 ifile ofile
```

```
maskindexbox,idx1,idx2,idy1,idy2 ifile ofile
```

### Description

Masked a box of the rectangularly understood field. The elements inside the box are untouched, the elements outside are set to missing value. All input fields need to have the same horizontal grid. Use [sellonlatbox](#) or [selindexbox](#) if only the data inside the box are needed.

### Operators

<b>masklonlatbox</b>	Mask a longitude/latitude box Masked a regular longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box. Considered are only those grid cells with the grid center inside the lon/lat box.
<b>maskindexbox</b>	Mask an index box Masked an index box. The user has to give the indexes of the edges of the box. The index of the left edge can be greater then the one of the right edge.

### Parameter

<i>lon1</i>	FLOAT	Western longitude
<i>lon2</i>	FLOAT	Eastern longitude
<i>lat1</i>	FLOAT	Southern or northern latitude
<i>lat2</i>	FLOAT	Northern or southern latitude
<i>idx1</i>	INTEGER	Index of first longitude
<i>idx2</i>	INTEGER	Index of last longitude
<i>idy1</i>	INTEGER	Index of first latitude
<i>idy2</i>	INTEGER	Index of last latitude

### Example

To mask the region with the longitudes from 120E to 90W and latitudes from 20N to 20S on all input fields use:

```
cdo masklonlatbox,120,-90,20,-20 ifile ofile
```

If the input dataset has fields on a Gaussian N16 grid, the same box can be masked with [maskindexbox](#) by:

```
cdo maskindexbox,23,48,13,20 ifile ofile
```

## 2.6.12. SETBOX - Set a box to constant

### Synopsis

```
setclonlatbox,c,lon1,lon2,lat1,lat2 ifile ofile
```

```
setcindexbox,c,idx1,idx2,idy1,idy2 ifile ofile
```

### Description

Sets a box of the rectangularly understood field to a constant value. The elements outside the box are untouched, the elements inside are set to the given constant. All input fields need to have the same horizontal grid.

### Operators

<b>setclonlatbox</b>	Set a longitude/latitude box to constant Sets the values of a longitude/latitude box to a constant value. The user has to give the longitudes and latitudes of the edges of the box.
<b>setcindexbox</b>	Set an index box to constant Sets the values of an index box to a constant value. The user has to give the indexes of the edges of the box. The index of the left edge can be greater than the one of the right edge.

### Parameter

<i>c</i>	FLOAT	Constant
<i>lon1</i>	FLOAT	Western longitude
<i>lon2</i>	FLOAT	Eastern longitude
<i>lat1</i>	FLOAT	Southern or northern latitude
<i>lat2</i>	FLOAT	Northern or southern latitude
<i>idx1</i>	INTEGER	Index of first longitude
<i>idx2</i>	INTEGER	Index of last longitude
<i>idy1</i>	INTEGER	Index of first latitude
<i>idy2</i>	INTEGER	Index of last latitude

### Example

To set all values in the region with the longitudes from 120E to 90W and latitudes from 20N to 20S to the constant value -1.23 use:

```
cdo setclonlatbox,-1.23,120,-90,20,-20 ifile ofile
```

If the input dataset has fields on a Gaussian N16 grid, the same box can be set with [setcindexbox](#) by:

```
cdo setcindexbox,-1.23,23,48,13,20 ifile ofile
```

## 2.6.13. ENLARGE - Enlarge fields

### Synopsis

```
enlarge,grid ifile ofile
```

### Description

Enlarge all fields of *ifile* to a user given grid. Normally only the last field element is used for the enlargement. If however the input and output grid are regular lon/lat grids, a zonal or meridional enlargement is possible. Zonal enlargement takes place, if the xsize of the input field is 1 and the ysize of both grids are the same. For meridional enlargement the ysize have to be 1 and the xsize of both grids should have the same size.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Example

Assumed you want to add two datasets. The first dataset is a field on a global grid (n field elements) and the second dataset is a global mean (1 field element). Before you can add these two datasets the second dataset have to be enlarged to the grid size of the first dataset:

```
cdo enlarge,ifile1 ifile2 tmpfile
cdo add ifile1 tmpfile ofile
```

Or shorter using operator piping:

```
cdo add ifile1 -enlarge,ifile1 ifile2 ofile
```



## 2.6.14. SETMISS - Set missing value

### Synopsis

```

setmissval,newmiss ifile ofile
setctomiss,c ifile ofile
setmisstoc,c ifile ofile
setrtomiss,rmin,rmax ifile ofile
setvrange,rmin,rmax ifile ofile
setmisstonn ifile ofile
setmisstodis[,neighbors] ifile ofile

```

### Description

This module sets part of a field to missing value or missing values to a constant value. Which part of the field is set depends on the chosen operator.

### Operators

<b>setmissval</b>	Set a new missing value $o(t, x) = \begin{cases} \text{newmiss} & \text{if } i(t, x) = \text{miss} \\ i(t, x) & \text{if } i(t, x) \neq \text{miss} \end{cases}$
<b>setctomiss</b>	Set constant to missing value $o(t, x) = \begin{cases} \text{miss} & \text{if } i(t, x) = c \\ i(t, x) & \text{if } i(t, x) \neq c \end{cases}$
<b>setmisstoc</b>	Set missing value to constant $o(t, x) = \begin{cases} c & \text{if } i(t, x) = \text{miss} \\ i(t, x) & \text{if } i(t, x) \neq \text{miss} \end{cases}$
<b>setrtomiss</b>	Set range to missing value $o(t, x) = \begin{cases} \text{miss} & \text{if } i(t, x) \geq rmin \wedge i(t, x) \leq rmax \\ i(t, x) & \text{if } i(t, x) < rmin \vee i(t, x) > rmax \end{cases}$
<b>setvrange</b>	Set valid range $o(t, x) = \begin{cases} \text{miss} & \text{if } i(t, x) < rmin \vee i(t, x) > rmax \\ i(t, x) & \text{if } i(t, x) \geq rmin \wedge i(t, x) \leq rmax \end{cases}$
<b>setmisstonn</b>	Set missing value to nearest neighbor Set all missing values to the nearest non missing value. $o(t, x) = \begin{cases} i(t, y) & \text{if } i(t, x) \geq \text{miss} \wedge i(t, y) \neq \text{miss} \\ i(t, x) & \text{if } i(t, x) \neq \text{miss} \end{cases}$
<b>setmisstodis</b>	Set missing value to distance-weighted average Set all missing values to the distance-weighted average of the nearest non missing values. The default number of nearest neighbors is 4.

### Parameter

<i>neighbors</i>	INTEGER	Number of nearest neighbors
<i>newmiss</i>	FLOAT	New missing value
<i>c</i>	FLOAT	Constant
<i>rmin</i>	FLOAT	Lower bound
<i>rmax</i>	FLOAT	Upper bound

## Example

Assume an input dataset has one field with temperatures in the range from 246 to 304 Kelvin. To set all values below 273.15 Kelvin to missing value use:

```
cdo setrtomiss,0,273.15 ifile ofile
```

Result of 'cdo info ifile':

-1 :	Date	Time	Code	Level	Size	Miss :	Minimum	Mean	Maximum
1 :	1987-12-31	12:00:00	139	0	2048	0 :	246.27	276.75	303.71

Result of 'cdo info ofile':

-1 :	Date	Time	Code	Level	Size	Miss :	Minimum	Mean	Maximum
1 :	1987-12-31	12:00:00	139	0	2048	871 :	273.16	287.08	303.71

## 2.7. Arithmetic

This section contains modules to arithmetically process datasets.

Here is a short overview of all operators in this section:

<b>expr</b>	Evaluate expressions
<b>exprf</b>	Evaluate expressions script
<b>aexpr</b>	Evaluate expressions and append results
<b>aexprf</b>	Evaluate expression script and append results
<b>abs</b>	Absolute value
<b>int</b>	Integer value
<b>nint</b>	Nearest integer value
<b>pow</b>	Power
<b>sqr</b>	Square
<b>sqrt</b>	Square root
<b>exp</b>	Exponential
<b>ln</b>	Natural logarithm
<b>log10</b>	Base 10 logarithm
<b>sin</b>	Sine
<b>cos</b>	Cosine
<b>tan</b>	Tangent
<b>asin</b>	Arc sine
<b>acos</b>	Arc cosine
<b>reci</b>	Reciprocal value
<b>addc</b>	Add a constant
<b>subc</b>	Subtract a constant
<b>mulc</b>	Multiply with a constant
<b>divc</b>	Divide by a constant
<b>add</b>	Add two fields
<b>sub</b>	Subtract two fields
<b>mul</b>	Multiply two fields
<b>div</b>	Divide two fields
<b>min</b>	Minimum of two fields
<b>max</b>	Maximum of two fields
<b>atan2</b>	Arc tangent of two fields
<b>monadd</b>	Add monthly time series
<b>monsub</b>	Subtract monthly time series
<b>monmul</b>	Multiply monthly time series
<b>monddiv</b>	Divide monthly time series
<b>yhouradd</b>	Add multi-year hourly time series
<b>yhoursub</b>	Subtract multi-year hourly time series
<b>yhourmul</b>	Multiply multi-year hourly time series
<b>yhourdiv</b>	Divide multi-year hourly time series
<b>ydayadd</b>	Add multi-year daily time series
<b>ydaysub</b>	Subtract multi-year daily time series
<b>ydaymul</b>	Multiply multi-year daily time series
<b>ydaydiv</b>	Divide multi-year daily time series

---

<b>ymonadd</b>	Add multi-year monthly time series
<b>ymonsub</b>	Subtract multi-year monthly time series
<b>ymonmul</b>	Multiply multi-year monthly time series
<b>ymonddiv</b>	Divide multi-year monthly time series
<b>yseasadd</b>	Add multi-year seasonal time series
<b>yseassub</b>	Subtract multi-year seasonal time series
<b>yseasmul</b>	Multiply multi-year seasonal time series
<b>yseasdiv</b>	Divide multi-year seasonal time series
<b>muldpm</b>	Multiply with days per month
<b>divdpm</b>	Divide by days per month
<b>muldpy</b>	Multiply with days per year
<b>divdpy</b>	Divide by days per year

## 2.7.1. EXPR - Evaluate expressions

### Synopsis

```

expr,instr ifile ofile
exprf,filename ifile ofile
aexpr,instr ifile ofile
aexprf,filename ifile ofile

```

### Description

This module arithmetically processes every timestep of the input dataset. Each individual assignment statement have to end with a semi-colon.

The following operators are supported:

Operator	Meaning	Example	Result
=	assignment	$x = y$	Assigns y to x
+	addition	$x + y$	Sum of x and y
-	subtraction	$x - y$	Difference of x and y
*	multiplication	$x * y$	Product of x and y
/	division	$x / y$	Quotient of x and y
^	exponentiation	$x ^ y$	Exponentiates x with y
==	equal to	$x == y$	1, if x equal to y; else 0
!=	not equal to	$x != y$	1, if x not equal to y; else 0
>	greater than	$x > y$	1, if x greater than y; else 0
<	less than	$x < y$	1, if x less than y; else 0
>=	greater equal	$x >= y$	1, if x greater equal y; else 0
<=	less equal	$x <= y$	1, if x less equal y; else 0
<=>	less equal greater	$x <=> y$	-1, if x less y; 1, if x greater y; else 0
&&	logical AND	$x \&\& y$	1, if x and y not equal 0; else 0
	logical OR	$x    y$	1, if x or y not equal 0; else 0
?:	ternary conditional	$x ? y : z$	y, if x not equal 0, else z

The following intrinsic functions are available:

```

abs(x)      Absolute value of x
floor(x)    Round to largest integral value not greater than x
ceil(x)     Round to smallest integral value not less than x
int(x)      Integer value of x
nint(x)     Nearest integer value of x
sqr(x)      Square of x
sqrt(x)     Square Root of x
exp(x)      Exponential of x
log(x)      Natural logarithm of x
log10(x)    Base 10 logarithm of x
sin(x)      Sine of x, where x is specified in radians
cos(x)      Cosine of x, where x is specified in radians

```

<code>tan(x)</code>	Tangent of x, where x is specified in radians
<code>asin(x)</code>	Arc-sine of x, where x is specified in radians
<code>acos(x)</code>	Arc-cosine of x, where x is specified in radians
<code>atan(x)</code>	Arc-tangent of x, where x is specified in radians

## Operators

<b>expr</b>	Evaluate expressions The processing instructions are read from the parameter.
<b>exprf</b>	Evaluate expressions script Contrary to <a href="#">expr</a> the processing instructions are read from a file.
<b>aexpr</b>	Evaluate expressions and append results Same as <a href="#">expr</a> , but keep input variables and append results
<b>aexprf</b>	Evaluate expression script and append results Same as <a href="#">exprf</a> , but keep input variables and append results

## Parameter

<i>instr</i>	STRING	Processing instructions (need to be 'quoted' in most cases)
<i>filename</i>	STRING	File with processing instructions

## Example

Assume an input dataset contains at least the variables 'aprl', 'aprc' and 'ts'. To create a new variable 'var1' with the sum of 'aprl' and 'aprc' and a variable 'var2' which convert the temperature 'ts' from Kelvin to Celsius use:

```
cdo expr,'var1=aprl+aprc;var2=ts-273.15;' ifile ofile
```

The same example, but the instructions are read from a file:

```
cdo exprf,myexpr ifile ofile
```

The file myexpr contains:

```
var1 = aprl + aprc;
var2 = ts - 273.15;
```

## 2.7.2. MATH - Mathematical functions

### Synopsis

`<operator> ifile ofile`

### Description

This module contains some standard mathematical functions. All trigonometric functions calculate with radians.

### Operators

<b>abs</b>	Absolute value $o(t, x) = \text{abs}(i(t, x))$
<b>int</b>	Integer value $o(t, x) = \text{int}(i(t, x))$
<b>nint</b>	Nearest integer value $o(t, x) = \text{nint}(i(t, x))$
<b>pow</b>	Power $o(t, x) = i(t, x)^y$
<b>sqr</b>	Square $o(t, x) = i(t, x)^2$
<b>sqrt</b>	Square root $o(t, x) = \sqrt{i(t, x)}$
<b>exp</b>	Exponential $o(t, x) = e^{i(t, x)}$
<b>ln</b>	Natural logarithm $o(t, x) = \ln(i(t, x))$
<b>log10</b>	Base 10 logarithm $o(t, x) = \log_{10}(i(t, x))$
<b>sin</b>	Sine $o(t, x) = \sin(i(t, x))$
<b>cos</b>	Cosine $o(t, x) = \cos(i(t, x))$
<b>tan</b>	Tangent $o(t, x) = \tan(i(t, x))$
<b>asin</b>	Arc sine $o(t, x) = \arcsin(i(t, x))$
<b>acos</b>	Arc cosine $o(t, x) = \arccos(i(t, x))$
<b>reci</b>	Reciprocal value $o(t, x) = 1/i(t, x)$

### Example

To calculate the square root for all field elements use:

```
cdo sqrt ifile ofile
```

### 2.7.3. ARITHC - Arithmetic with a constant

#### Synopsis

`<operator>,c ifile ofile`

#### Description

This module performs simple arithmetic with all field elements of a dataset and a constant. The fields in `ofile` inherit the meta data from `ifile`.

#### Operators

<b>addc</b>	Add a constant $o(t, x) = i(t, x) + c$
<b>subc</b>	Subtract a constant $o(t, x) = i(t, x) - c$
<b>mulc</b>	Multiply with a constant $o(t, x) = i(t, x) * c$
<b>divc</b>	Divide by a constant $o(t, x) = i(t, x) / c$

#### Parameter

<code>c</code>	FLOAT	Constant
----------------	-------	----------

#### Example

To sum all input fields with the constant -273.15 use:

```
cdo addc,-273.15 ifile ofile
```



## 2.7.4. ARITH - Arithmetic on two datasets

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module performs simple arithmetic of two datasets. The number of fields in `ifile1` should be the same as in `ifile2`. The fields in `ofile` inherit the meta data from `ifile1`. One of the input files can contain only one timestep or one variable.

### Operators

<b>add</b>	Add two fields $o(t, x) = i_1(t, x) + i_2(t, x)$
<b>sub</b>	Subtract two fields $o(t, x) = i_1(t, x) - i_2(t, x)$
<b>mul</b>	Multiply two fields $o(t, x) = i_1(t, x) * i_2(t, x)$
<b>div</b>	Divide two fields $o(t, x) = i_1(t, x) / i_2(t, x)$
<b>min</b>	Minimum of two fields $o(t, x) = \min(i_1(t, x), i_2(t, x))$
<b>max</b>	Maximum of two fields $o(t, x) = \max(i_1(t, x), i_2(t, x))$
<b>atan2</b>	Arc tangent of two fields The <i>atan2</i> operator calculates the arc tangent of two fields. The result is in radians, which is between -PI and PI (inclusive). $o(t, x) = \text{atan2}(i_1(t, x), i_2(t, x))$

### Example

To sum all fields of the first input file with the corresponding fields of the second input file use:

```
cdo add ifile1 ifile2 ofile
```

## 2.7.5. MONARITH - Monthly arithmetic

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module performs simple arithmetic of a time series and one timestep with the same month and year. For each field in ifile1 the corresponding field of the timestep in ifile2 with the same month and year is used. The header information in ifile1 have to be the same as in ifile2. Usually ifile2 is generated by an operator of the module [MONSTAT](#).

### Operators

<b>monadd</b>	Add monthly time series Adds a time series and a monthly time series.
<b>monsub</b>	Subtract monthly time series Subtracts a time series and a monthly time series.
<b>monmul</b>	Multiply monthly time series Multiplies a time series and a monthly time series.
<b>monddiv</b>	Divide monthly time series Divides a time series and a monthly time series.

### Example

To subtract a monthly time average from a time series use:

```
cdo monsub ifile -monavg ifile ofile
```

## 2.7.6. YHOURARITH - Multi-year hourly arithmetic

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module performs simple arithmetic of a time series and one timestep with the same hour and day of year. For each field in ifile1 the corresponding field of the timestep in ifile2 with the same hour and day of year is used. The header information in ifile1 have to be the same as in ifile2. Usually ifile2 is generated by an operator of the module [YHOURSTAT](#).

### Operators

<b>yhouradd</b>	Add multi-year hourly time series Adds a time series and a multi-year hourly time series.
<b>yhoursub</b>	Subtract multi-year hourly time series Subtracts a time series and a multi-year hourly time series.
<b>yhourmul</b>	Multiply multi-year hourly time series Multiplies a time series and a multi-year hourly time series.
<b>yhourdiv</b>	Divide multi-year hourly time series Divides a time series and a multi-year hourly time series.

### Example

To subtract a multi-year hourly time average from a time series use:

```
cdo yhoursub ifile -yhouravg ifile ofile
```

## 2.7.7. YDAYARITH - Multi-year daily arithmetic

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module performs simple arithmetic of a time series and one timestep with the same day of year. For each field in `ifile1` the corresponding field of the timestep in `ifile2` with the same day of year is used. The header information in `ifile1` have to be the same as in `ifile2`. Usually `ifile2` is generated by an operator of the module [YDAYSTAT](#).

### Operators

<b>ydayadd</b>	Add multi-year daily time series Adds a time series and a multi-year daily time series.
<b>ydaysub</b>	Subtract multi-year daily time series Subtracts a time series and a multi-year daily time series.
<b>ydaymul</b>	Multiply multi-year daily time series Multiplies a time series and a multi-year daily time series.
<b>ydaydiv</b>	Divide multi-year daily time series Divides a time series and a multi-year daily time series.

### Example

To subtract a multi-year daily time average from a time series use:

```
cdo ydaysub ifile -ydayavg ifile ofile
```

## 2.7.8. YMONARITH - Multi-year monthly arithmetic

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module performs simple arithmetic of a time series and one timestep with the same month of year. For each field in ifile1 the corresponding field of the timestep in ifile2 with the same month of year is used. The header information in ifile1 have to be the same as in ifile2. Usually ifile2 is generated by an operator of the module [YMONSTAT](#).

### Operators

<b>ymonadd</b>	Add multi-year monthly time series Adds a time series and a multi-year monthly time series.
<b>ymonsub</b>	Subtract multi-year monthly time series Subtracts a time series and a multi-year monthly time series.
<b>ymonmul</b>	Multiply multi-year monthly time series Multiplies a time series and a multi-year monthly time series.
<b>ymonddiv</b>	Divide multi-year monthly time series Divides a time series and a multi-year monthly time series.

### Example

To subtract a multi-year monthly time average from a time series use:

```
cdo ymonsub ifile -ymonavg ifile ofile
```

### 2.7.9. YSEASARITH - Multi-year seasonal arithmetic

#### Synopsis

```
<operator> ifile1 ifile2 ofile
```

#### Description

This module performs simple arithmetic of a time series and one timestep with the same season. For each field in ifile1 the corresponding field of the timestep in ifile2 with the same season is used. The header information in ifile1 have to be the same as in ifile2. Usually ifile2 is generated by an operator of the module [YSEASSTAT](#).

#### Operators

<b>yseasadd</b>	Add multi-year seasonal time series Adds a time series and a multi-year seasonal time series.
<b>yseassub</b>	Subtract multi-year seasonal time series Subtracts a time series and a multi-year seasonal time series.
<b>yseasmul</b>	Multiply multi-year seasonal time series Multiplies a time series and a multi-year seasonal time series.
<b>yseasdiv</b>	Divide multi-year seasonal time series Divides a time series and a multi-year seasonal time series.

#### Example

To subtract a multi-year seasonal time average from a time series use:

```
cdo yseassub ifile -yseasavg ifile ofile
```

### 2.7.10. ARITHDAYS - Arithmetic with days

#### Synopsis

```
<operator> ifile ofile
```

#### Description

This module multiplies or divides each timestep of a dataset with the corresponding days per month or days per year. The result of these functions depends on the used calendar of the input data.

#### Operators

<b>muldpm</b>	Multiply with days per month $o(t, x) = i(t, x) * days\_per\_month$
<b>divdpm</b>	Divide by days per month $o(t, x) = i(t, x) / days\_per\_month$
<b>muldpy</b>	Multiply with days per year $o(t, x) = i(t, x) * days\_per\_year$
<b>divdpy</b>	Divide by days per year $o(t, x) = i(t, x) / days\_per\_year$

## 2.8. Statistical values

This section contains modules to compute statistical values of datasets. In this program there is the different notion of "mean" and "average" to distinguish two different kinds of treatment of missing values. While computing the mean, only the not missing values are considered to belong to the sample with the side effect of a probably reduced sample size. Computing the average is just adding the sample members and divide the result by the sample size. For example, the mean of 1, 2, miss and 3 is  $(1+2+3)/3 = 2$ , whereas the average is  $(1+2+miss+3)/4 = miss/4 = miss$ . If there are no missing values in the sample, the average and the mean are identical.

This program is using the verification time to identify the time range for time-statistics. The time bounds are never used!

In this section the abbreviations as in the following table are used:

<b>sum</b>	$\sum_{i=1}^n x_i$
<b>mean</b> resp. <b>avg</b>	$n^{-1} \sum_{i=1}^n x_i$
<b>mean</b> resp. <b>avg</b> weighted by $\{w_i, i = 1, \dots, n\}$	$\left( \sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i x_i$
Variance <b>var</b>	$n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2$
<b>var1</b>	$(n-1)^{-1} \sum_{i=1}^n (x_i - \bar{x})^2$
<b>var</b> weighted by $\{w_i, i = 1, \dots, n\}$	$\left( \sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left( x_i - \left( \sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right)^2$
Standard deviation <b>std</b>	$\sqrt{n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2}$
<b>std1</b>	$\sqrt{(n-1)^{-1} \sum_{i=1}^n (x_i - \bar{x})^2}$
<b>std</b> weighted by $\{w_i, i = 1, \dots, n\}$	$\sqrt{\left( \sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left( x_i - \left( \sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right)^2}$
Cumulative Ranked Probability Score <b>crps</b>	$\int_{-\infty}^{\infty} [H(x_1) - cdf(\{x_2 \dots x_n\}) _r]^2 dr$
with $cdf(X) _r$ being the cumulative distribution function of $\{x_i, i = 2 \dots n\}$ at $r$	
and $H(x)$ the Heavyside function jumping at $x$ .	

Here is a short overview of all operators in this section:

<b>consecsum</b>	Consecutive Sum
<b>consects</b>	Consecutive Timesteps
<b>ensmin</b>	Ensemble minimum
<b>ensmax</b>	Ensemble maximum
<b>enssum</b>	Ensemble sum
<b>ensmean</b>	Ensemble mean
<b>ensavg</b>	Ensemble average
<b>ensstd</b>	Ensemble standard deviation
<b>ensstd1</b>	Ensemble standard deviation
<b>ensvar</b>	Ensemble variance
<b>ensvar1</b>	Ensemble variance
<b>enspctl</b>	Ensemble percentiles
<b>ensrkhistspace</b>	Ranked Histogram averaged over time
<b>ensrkhisttime</b>	Ranked Histogram averaged over space
<b>ensroc</b>	Ensemble Receiver Operating characteristics
<b>enscrps</b>	Ensemble CRPS and decomposition
<b>ensbrs</b>	Ensemble Brier score
<b>fldmin</b>	Field minimum
<b>fldmax</b>	Field maximum
<b>fldsum</b>	Field sum
<b>fldmean</b>	Field mean
<b>fldavg</b>	Field average
<b>fldstd</b>	Field standard deviation
<b>fldstd1</b>	Field standard deviation
<b>fldvar</b>	Field variance
<b>fldvar1</b>	Field variance
<b>fldpctl</b>	Field percentiles
<b>zonmin</b>	Zonal minimum
<b>zonmax</b>	Zonal maximum
<b>zonsum</b>	Zonal sum
<b>zonmean</b>	Zonal mean
<b>zonavg</b>	Zonal average
<b>zonstd</b>	Zonal standard deviation
<b>zonstd1</b>	Zonal standard deviation
<b>zonvar</b>	Zonal variance
<b>zonvar1</b>	Zonal variance
<b>zonpctl</b>	Zonal percentiles
<b>mermin</b>	Meridional minimum
<b>mermax</b>	Meridional maximum
<b>mersum</b>	Meridional sum
<b>mermean</b>	Meridional mean
<b>meravg</b>	Meridional average
<b>merstd</b>	Meridional standard deviation
<b>merstd1</b>	Meridional standard deviation
<b>mervar</b>	Meridional variance
<b>mervar1</b>	Meridional variance
<b>merpctl</b>	Meridional percentiles



<b>gridboxmin</b>	Gridbox minimum
<b>gridboxmax</b>	Gridbox maximum
<b>gridboxsum</b>	Gridbox sum
<b>gridboxmean</b>	Gridbox mean
<b>gridboxavg</b>	Gridbox average
<b>gridboxstd</b>	Gridbox standard deviation
<b>gridboxstd1</b>	Gridbox standard deviation
<b>gridboxvar</b>	Gridbox variance
<b>gridboxvar1</b>	Gridbox variance
<b>vertmin</b>	Vertical minimum
<b>vertmax</b>	Vertical maximum
<b>vertsum</b>	Vertical sum
<b>vertmean</b>	Vertical mean
<b>vertavg</b>	Vertical average
<b>vertstd</b>	Vertical standard deviation
<b>vertstd1</b>	Vertical standard deviation
<b>vertvar</b>	Vertical variance
<b>vertvar1</b>	Vertical variance
<b>timselmin</b>	Time range minimum
<b>timselmax</b>	Time range maximum
<b>timselsum</b>	Time range sum
<b>timselmean</b>	Time range mean
<b>timselavg</b>	Time range average
<b>timselstd</b>	Time range standard deviation
<b>timselstd1</b>	Time range standard deviation
<b>timselvar</b>	Time range variance
<b>timselvar1</b>	Time range variance
<b>timselfctl</b>	Time range percentiles
<b>runmin</b>	Running minimum
<b>runmax</b>	Running maximum
<b>runsum</b>	Running sum
<b>runmean</b>	Running mean
<b>runavg</b>	Running average
<b>runstd</b>	Running standard deviation
<b>runstd1</b>	Running standard deviation
<b>runvar</b>	Running variance
<b>runvar1</b>	Running variance
<b>runpctl</b>	Running percentiles
<b>timmin</b>	Time minimum
<b>timmax</b>	Time maximum
<b>timsum</b>	Time sum
<b>timmean</b>	Time mean
<b>timavg</b>	Time average
<b>timstd</b>	Time standard deviation
<b>timstd1</b>	Time standard deviation
<b>timvar</b>	Time variance
<b>timvar1</b>	Time variance
<b>timpctl</b>	Time percentiles

<b>hourmin</b>	Hourly minimum
<b>hourmax</b>	Hourly maximum
<b>hoursum</b>	Hourly sum
<b>hourmean</b>	Hourly mean
<b>houravg</b>	Hourly average
<b>hourstd</b>	Hourly standard deviation
<b>hourstd1</b>	Hourly standard deviation
<b>hourvar</b>	Hourly variance
<b>hourvar1</b>	Hourly variance
<b>hourctl</b>	Hourly percentiles
<b>daymin</b>	Daily minimum
<b>daymax</b>	Daily maximum
<b>daysum</b>	Daily sum
<b>daymean</b>	Daily mean
<b>dayavg</b>	Daily average
<b>daystd</b>	Daily standard deviation
<b>daystd1</b>	Daily standard deviation
<b>dayvar</b>	Daily variance
<b>dayvar1</b>	Daily variance
<b>dayctl</b>	Daily percentiles
<b>monmin</b>	Monthly minimum
<b>monmax</b>	Monthly maximum
<b>monsum</b>	Monthly sum
<b>monmean</b>	Monthly mean
<b>monavg</b>	Monthly average
<b>monstd</b>	Monthly standard deviation
<b>monstd1</b>	Monthly standard deviation
<b>monvar</b>	Monthly variance
<b>monvar1</b>	Monthly variance
<b>monctl</b>	Monthly percentiles
<b>yearmonmean</b>	Yearly mean from monthly data
<b>yearmin</b>	Yearly minimum
<b>yearmax</b>	Yearly maximum
<b>yearsum</b>	Yearly sum
<b>yearmean</b>	Yearly mean
<b>yearavg</b>	Yearly average
<b>yearstd</b>	Yearly standard deviation
<b>yearstd1</b>	Yearly standard deviation
<b>yearvar</b>	Yearly variance
<b>yearvar1</b>	Yearly variance
<b>yearctl</b>	Yearly percentiles

<b>seasmin</b>	Seasonal minimum
<b>seasmax</b>	Seasonal maximum
<b>seassum</b>	Seasonal sum
<b>seasmean</b>	Seasonal mean
<b>seasavg</b>	Seasonal average
<b>seasstd</b>	Seasonal standard deviation
<b>seasstd1</b>	Seasonal standard deviation
<b>seasvar</b>	Seasonal variance
<b>seasvar1</b>	Seasonal variance
<b>seaspctl</b>	Seasonal percentiles
<b>yhourmin</b>	Multi-year hourly minimum
<b>yhourmax</b>	Multi-year hourly maximum
<b>yhoursum</b>	Multi-year hourly sum
<b>yhourmean</b>	Multi-year hourly mean
<b>yhouravg</b>	Multi-year hourly average
<b>yhourstd</b>	Multi-year hourly standard deviation
<b>yhourstd1</b>	Multi-year hourly standard deviation
<b>yhourvar</b>	Multi-year hourly variance
<b>yhourvar1</b>	Multi-year hourly variance
<b>ydaymin</b>	Multi-year daily minimum
<b>ydaymax</b>	Multi-year daily maximum
<b>ydaysum</b>	Multi-year daily sum
<b>ydaymean</b>	Multi-year daily mean
<b>ydayavg</b>	Multi-year daily average
<b>ydaystd</b>	Multi-year daily standard deviation
<b>ydaystd1</b>	Multi-year daily standard deviation
<b>ydayvar</b>	Multi-year daily variance
<b>ydayvar1</b>	Multi-year daily variance
<b>ydaypctl</b>	Multi-year daily percentiles
<b>ymonmin</b>	Multi-year monthly minimum
<b>ymonmax</b>	Multi-year monthly maximum
<b>ymonsum</b>	Multi-year monthly sum
<b>ymonmean</b>	Multi-year monthly mean
<b>ymonavg</b>	Multi-year monthly average
<b>ymonstd</b>	Multi-year monthly standard deviation
<b>ymonstd1</b>	Multi-year monthly standard deviation
<b>ymonvar</b>	Multi-year monthly variance
<b>ymonvar1</b>	Multi-year monthly variance
<b>ymonpctl</b>	Multi-year monthly percentiles
<b>yseasmin</b>	Multi-year seasonal minimum
<b>yseasmax</b>	Multi-year seasonal maximum
<b>yseassum</b>	Multi-year seasonal sum
<b>yseasmean</b>	Multi-year seasonal mean
<b>yseasavg</b>	Multi-year seasonal average
<b>yseasstd</b>	Multi-year seasonal standard deviation
<b>yseasstd1</b>	Multi-year seasonal standard deviation
<b>yseasvar</b>	Multi-year seasonal variance
<b>yseasvar1</b>	Multi-year seasonal variance

---

<b>yseaspctl</b>	Multi-year seasonal percentiles
<b>ydrunmin</b>	Multi-year daily running minimum
<b>ydrunmax</b>	Multi-year daily running maximum
<b>ydrunsum</b>	Multi-year daily running sum
<b>ydrunmean</b>	Multi-year daily running mean
<b>ydrunavg</b>	Multi-year daily running average
<b>ydrunstd</b>	Multi-year daily running standard deviation
<b>ydrunstd1</b>	Multi-year daily running standard deviation
<b>ydrunvar</b>	Multi-year daily running variance
<b>ydrunvar1</b>	Multi-year daily running variance
<b>ydrunpctl</b>	Multi-year daily running percentiles

## 2.8.1. CONSECSTAT - Consecutive timestep periods

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes periods over all timesteps in *ifile* where a certain property is valid. The property can be chosen by creating a mask from the original data, which is the expected input format for operators of this module. Depending on the operator full information about each period or just its length and ending date are computed.

### Operators

<b>consecsum</b>	<b>Consecutive Sum</b> This operator computes periods of consecutive timesteps similar to a <a href="#">runsum</a> , but periods are finished, when the mask value is 0. That way multiple periods can be found. Timesteps from the input are preserved. Missing values are handled like 0, i.e. finish periods of consecutive timesteps.
<b>consects</b>	<b>Consecutive Timesteps</b> In contrast to the operator above <i>consects</i> only computes the length of each period together with its last timestep. To be able to perform statistical analysis like min, max or mean, everything else is set to missing value.

### Example

For a given time series of daily temperatures, the periods of summer days can be calculated with inplace masking the input field:

```
cdo consects -gtc,20.0 ifile1 ofile
```

## 2.8.2. ENSSTAT - Statistical values over an ensemble

### Synopsis

`<operator> ifiles ofile`

`enspctl,p ifiles ofile`

### Description

This module computes statistical values over an ensemble of input files. Depending on the chosen operator the minimum, maximum, sum, average, variance, standard deviation or a certain percentile over all input files is written to ofile. All input files need to have the same structure with the same variables. The date information of a timestep in ofile is the date of the first input file.

### Operators

<b>ensmin</b>	Ensemble minimum $o(t, x) = \mathbf{min}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensmax</b>	Ensemble maximum $o(t, x) = \mathbf{max}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>enssum</b>	Ensemble sum $o(t, x) = \mathbf{sum}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensmean</b>	Ensemble mean $o(t, x) = \mathbf{mean}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensavg</b>	Ensemble average $o(t, x) = \mathbf{avg}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensstd</b>	Ensemble standard deviation Divisor is n. $o(t, x) = \mathbf{std}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensstd1</b>	Ensemble standard deviation Divisor is (n-1). $o(t, x) = \mathbf{std1}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensvar</b>	Ensemble variance Divisor is n. $o(t, x) = \mathbf{var}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensvar1</b>	Ensemble variance Divisor is (n-1). $o(t, x) = \mathbf{var1}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>enspctl</b>	Ensemble percentiles $o(t, x) = \mathbf{pth\ percentile}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$

### Parameter

`p`      `FLOAT`      Percentile number in 0, ..., 100

**Example**

To compute the ensemble mean over 6 input files use:

```
cdo ensmean ifile1 ifile2 ifile3 ifile4 ifile5 ifile6 ofile
```

Or shorter with filename substitution:

```
cdo ensmean ifile[1-6] ofile
```

To compute the 50th percentile (median) over 6 input files use:

```
cdo enspctl,50 ifile1 ifile2 ifile3 ifile4 ifile5 ifile6 ofile
```

### 2.8.3. ENSSTAT2 - Statistical values over an ensemble

#### Synopsis

```
<operator> obsfile ensfiles ofile
```

#### Description

This module computes statistical values over the ensemble of ensfiles using obsfile as a reference. Depending on the operator a ranked Histogram or a roc-curve over all Ensembles ensfiles with reference to obsfile is written to ofile. The date and grid information of a timestep in ofile is the date of the first input file. Thus all input files are required to have the same structure in terms of the gridsize, variable definitions and number of timesteps.

All Operators in this module use obsfile as the reference (for instance an observation) whereas ensfiles are understood as an ensemble consisting of n (where n is the number of ensfiles) members.

The operators ensrkhistspace and ensrkhisttime compute Ranked Histograms. Therefor the vertical axis is utilized as the Histogram axis, which prohibits the use of files containing more than one level. The histogram axis has nensfiles+1 bins with level 0 containing for each grid point the number of observations being smaller as all ensembles and level nensfiles+1 indicating the number of observations being larger than all ensembles.

ensrkhistspace computes a ranked histogram at each timestep reducing each horizontal grid to a 1x1 grid and keeping the time axis as in obsfile. Contrary ensrkhisttime computes a histogram at each grid point keeping the horizontal grid for each variable and reducing the time-axis. The time information is that from the last timestep in obsfile.

#### Operators

<b>ensrkhistspace</b>	Ranked Histogram averaged over time
<b>ensrkhisttime</b>	Ranked Histogram averaged over space
<b>ensroc</b>	Ensemble Receiver Operating characteristics

#### Example

To compute a rank histogram over 5 input files ensfile1-ensfile5 given an observation in obsfile use:

```
cdo ensrkhisttime obsfile ensfile1 ensfile2 ensfile3 ensfile4 ensfile5 ofile
```

Or shorter with filename substitution:

```
cdo ensrkhisttime obsfile ensfile[1-5] ofile
```



## 2.8.4. ENSVAL - Ensemble validation tools

### Synopsis

```
enscrps rfile ifiles ofilebase
ensbrs,x rfile ifiles ofilebase
```

### Description

This module computes ensemble validation scores and their decomposition such as the Brier and cumulative ranked probability score (CRPS). The first file is used as a reference it can be a climatology, observation or reanalysis against which the skill of the ensembles given in `ifiles` is measured. Depending on the operator a number of output files is generated each containing the skill score and its decomposition corresponding to the operator. The output is averaged over horizontal fields using appropriate weights for each level and timestep in `rfile`.

All input files need to have the same structure with the same variables. The date information of a timestep in `ofile` is the date of the first input file. The output files are named as `<ofilebase>.<type>.<filesuffix>` where `<type>` depends on the operator and `<filesuffix>` is determined from the output file type. There are three output files for operator `enscrps` and four output files for operator `ensbrs`.

The CRPS and its decomposition into Reliability and the potential CRPS are calculated by an appropriate averaging over the field members (note, that the CRPS does *\*not\** average linearly). In the three output files `<type>` has the following meaning: `crps` for the CRPS, `reli` for the reliability and `crpspot` for the potential crps. The relation  $CRPS = CRPS_{pot} + RELI$

holds.

The Brier score of the Ensemble given by `ifiles` with respect to the reference given in `rfile` and the threshold `x` is calculated. In the four output files `<type>` has the following meaning: `brs` for the Brier score wrt threshold `x`; `brsreli` for the Brier score reliability wrt threshold `x`; `brsreso` for the Brier score resolution wrt threshold `x`; `brsunct` for the Brier score uncertainty wrt threshold `x`. In analogy to the CRPS the following relation holds:  $BRS(x) = RELI(x) - RESO(x) + UNCT(x)$ .

The implementation of the decomposition of the CRPS and Brier Score follows Hans Hersbach (2000): Decomposition of the Continuous Ranked Probability Score for Ensemble Prediction Systems, in: Weather and Forecasting (15) pp. 559-570.

The CRPS code decomposition has been verified against the CRAN - ensemble validation package from R. Differences occur when grid-cell area is not uniform as the implementation in R does not account for that.

### Operators

<b>enscrps</b>	Ensemble CRPS and decomposition
<b>ensbrs</b>	Ensemble Brier score Ensemble Brier Score and Decomposition

### Example

To compute the field averaged Brier score at `x=5` over an ensemble with 5 members `ensfile1-5` w.r.t. the reference `rfile` and write the results to files `obase.brs.<suff>`, `obase.brsreli<suff>`, `obase.brsreso<suff>`, `obase.brsunct<suff>` where `<suff>` is determined from the output file type, use

```
cdo ensbrs,5 rfile ensfile1 ensfile2 ensfile3 ensfile4 ensfile5 obase
```

or shorter using file name substitution:

```
cdo ensbrs,5 rfile ensfile[1-5] obase
```

## 2.8.5. FLDSTAT - Statistical values over a field

### Synopsis

```
<operator> ifile ofile
fldpctl,p ifile ofile
```

### Description

This module computes statistical values of the input fields. According to the chosen operator the field minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to ofile.

### Operators

<b>fldmin</b>	Field minimum For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{min}\{i(t, x'), x_1 < x' \leq x_n\}$
<b>fldmax</b>	Field maximum For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{max}\{i(t, x'), x_1 < x' \leq x_n\}$
<b>fldsum</b>	Field sum For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{sum}\{i(t, x'), x_1 < x' \leq x_n\}$
<b>fldmean</b>	Field mean For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{mean}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldavg</b>	Field average For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{avg}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldstd</b>	Field standard deviation Divisor is n. For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{std}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldstd1</b>	Field standard deviation Divisor is (n-1). For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{std1}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldvar</b>	Field variance Divisor is n. For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{var}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldvar1</b>	Field variance Divisor is (n-1). For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{var1}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldpctl</b>	Field percentiles For every gridpoint $x_1, \dots, x_n$ of the same field it is: $o(t, 1) = \mathbf{pth\ percentile}\{i(t, x'), x_1 < x' \leq x_n\}$

**Parameter**

$p$       FLOAT      Percentile number in 0, ..., 100

**Example**

To compute the field mean of all input fields use:

```
cdo fldmean ifile ofile
```

To compute the 90th percentile of all input fields use:

```
cdo fldpctl,90 ifile ofile
```

## 2.8.6. ZONSTAT - Zonal statistical values

### Synopsis

```
<operator> ifile ofile
zonpctl,p ifile ofile
```

### Description

This module computes zonal statistical values of the input fields. According to the chosen operator the zonal minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to ofile. This operator requires all variables on the same regular lon/lat grid.

### Operators

<b>zonmin</b>	Zonal minimum For every latitude the minimum over all longitudes is computed.
<b>zonmax</b>	Zonal maximum For every latitude the maximum over all longitudes is computed.
<b>zonsum</b>	Zonal sum For every latitude the sum over all longitudes is computed.
<b>zonmean</b>	Zonal mean For every latitude the mean over all longitudes is computed.
<b>zonavg</b>	Zonal average For every latitude the average over all longitudes is computed.
<b>zonstd</b>	Zonal standard deviation For every latitude the standard deviation over all longitudes is computed. Divisor is n.
<b>zonstd1</b>	Zonal standard deviation For every latitude the standard deviation over all longitudes is computed. Divisor is (n-1).
<b>zonvar</b>	Zonal variance For every latitude the variance over all longitudes is computed. Divisor is n.
<b>zonvar1</b>	Zonal variance For every latitude the variance over all longitudes is computed. Divisor is (n-1).
<b>zonpctl</b>	Zonal percentiles For every latitude the pth percentile over all longitudes is computed.

### Parameter

*p*     FLOAT     Percentile number in 0, ..., 100

### Example

To compute the zonal mean of all input fields use:

```
cdo zonmean ifile ofile
```

To compute the 50th meridional percentile (median) of all input fields use:

```
cdo zonpctl,50 ifile ofile
```

## 2.8.7. MERSTAT - Meridional statistical values

### Synopsis

```
<operator> ifile ofile
merpctl,p ifile ofile
```

### Description

This module computes meridional statistical values of the input fields. According to the chosen operator the meridional minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to ofile. This operator requires all variables on the same regular lon/lat grid.

### Operators

<b>mermin</b>	Meridional minimum For every longitude the minimum over all latitudes is computed.
<b>mermax</b>	Meridional maximum For every longitude the maximum over all latitudes is computed.
<b>mersum</b>	Meridional sum For every longitude the sum over all latitudes is computed.
<b>mermean</b>	Meridional mean For every longitude the area weighted mean over all latitudes is computed.
<b>meravg</b>	Meridional average For every longitude the area weighted average over all latitudes is computed.
<b>merstd</b>	Meridional standard deviation For every longitude the standard deviation over all latitudes is computed. Divisor is n.
<b>merstd1</b>	Meridional standard deviation For every longitude the standard deviation over all latitudes is computed. Divisor is (n-1).
<b>mervar</b>	Meridional variance For every longitude the variance over all latitudes is computed. Divisor is n.
<b>mervar1</b>	Meridional variance For every longitude the variance over all latitudes is computed. Divisor is (n-1).
<b>merpctl</b>	Meridional percentiles For every longitude the pth percentile over all latitudes is computed.

### Parameter

*p*      FLOAT      Percentile number in 0, ..., 100

### Example

To compute the meridional mean of all input fields use:

```
cdo mermean ifile ofile
```

To compute the 50th meridional percentile (median) of all input fields use:

```
cdo merpctl,50 ifile ofile
```

## 2.8.8. GRIDBOXSTAT - Statistical values over grid boxes

### Synopsis

```
<operator>,nx,ny ifile ofile
```

### Description

This module computes statistical values over surrounding grid boxes. According to the chosen operator the minimum, maximum, sum, average, variance, or standard deviation of the neighboring grid boxes is written to ofile. All gridbox operators only works on quadrilateral curvilinear grids.

### Operators

<b>gridboxmin</b>	Gridbox minimum Minimum value of the selected grid boxes.
<b>gridboxmax</b>	Gridbox maximum Maximum value of the selected grid boxes.
<b>gridboxsum</b>	Gridbox sum Sum of the selected grid boxes.
<b>gridboxmean</b>	Gridbox mean Mean of the selected grid boxes.
<b>gridboxavg</b>	Gridbox average Average of the selected grid boxes.
<b>gridboxstd</b>	Gridbox standard deviation Standard deviation of the selected grid boxes. Divisor is n.
<b>gridboxstd1</b>	Gridbox standard deviation Standard deviation of the selected grid boxes. Divisor is (n-1).
<b>gridboxvar</b>	Gridbox variance Variance of the selected grid boxes. Divisor is n.
<b>gridboxvar1</b>	Gridbox variance Variance of the selected grid boxes. Divisor is (n-1).

### Parameter

<i>nx</i>	INTEGER	Number of grid boxes in x direction
<i>ny</i>	INTEGER	Number of grid boxes in y direction

### Example

To compute the mean over 10x10 grid boxes of the input field use:

```
cdo gridboxmean,10,10 ifile ofile
```

## 2.8.9. VERTSTAT - Vertical statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over all levels of the input variables. According to chosen operator the vertical minimum, maximum, sum, average, variance or standard deviation is written to ofile.

### Operators

<b>vertmin</b>	Vertical minimum For every gridpoint the minimum over all levels is computed.
<b>vertmax</b>	Vertical maximum For every gridpoint the maximum over all levels is computed.
<b>vertsum</b>	Vertical sum For every gridpoint the sum over all levels is computed.
<b>vertmean</b>	Vertical mean For every gridpoint the layer weighted mean over all levels is computed.
<b>vertavg</b>	Vertical average For every gridpoint the layer weighted average over all levels is computed.
<b>vertstd</b>	Vertical standard deviation For every gridpoint the standard deviation over all levels is computed. Divisor is n.
<b>vertstd1</b>	Vertical standard deviation For every gridpoint the standard deviation over all levels is computed. Divisor is (n-1).
<b>vertvar</b>	Vertical variance For every gridpoint the variance over all levels is computed. Divisor is n.
<b>vertvar1</b>	Vertical variance For every gridpoint the variance over all levels is computed. Divisor is (n-1).

### Example

To compute the vertical sum of all input variables use:

```
cdo vertsum ifile ofile
```



## 2.8.10. TIMSELSTAT - Time range statistical values

### Synopsis

`<operator>,nsets[,noffset[,nskip]] ifile ofile`

### Description

This module computes statistical values for a selected number of timesteps. According to the chosen operator the minimum, maximum, sum, average, variance or standard deviation of the selected timesteps is written to ofile. The time stamp in ofile is from the middle contributing timestep of ifile.

### Operators

<b>timselmin</b>	Time range minimum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselmax</b>	Time range maximum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselsum</b>	Time range sum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselmean</b>	Time range mean For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselavg</b>	Time range average For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselstd</b>	Time range standard deviation Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselstd1</b>	Time range standard deviation Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselvar</b>	Time range variance Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselvar1</b>	Time range variance Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same selected time range it is: $o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

**Parameter**

<i>nsets</i>	INTEGER	Number of input timesteps for each output timestep
<i>noffset</i>	INTEGER	Number of input timesteps skipped before the first timestep range (optional)
<i>nskip</i>	INTEGER	Number of input timesteps skipped between timestep ranges (optional)

**Example**

Assume an input dataset has monthly means over several years. To compute seasonal means from monthly means the first two month have to be skipped:

```
cdo timselmean,3,2 ifile ofile
```

**2.8.11. TIMSELPCTL - Time range percentile values****Synopsis**

```
timselpctl,p,nsets[,noffset[,nskip]] ifile1 ifile2 ifile3 ofile
```

**Description**

This operator computes percentile values over a selected number of timesteps in *ifile1*. The algorithm uses histograms with minimum and maximum bounds given in *ifile2* and *ifile3*, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files *ifile2* and *ifile3* should be the result of corresponding [timselmin](#) and [timselmax](#) operations, respectively. The time stamp in *ofile* is from the middle contributing timestep of *ifile1*.

For every adjacent sequence  $t_1, \dots, t_n$  of timesteps of the same selected time range it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

**Parameter**

<i>p</i>	FLOAT	Percentile number in 0, ..., 100
<i>nsets</i>	INTEGER	Number of input timesteps for each output timestep
<i>noffset</i>	INTEGER	Number of input timesteps skipped before the first timestep range (optional)
<i>nskip</i>	INTEGER	Number of input timesteps skipped between timestep ranges (optional)

**Environment**

<code>CDO_PCTL_NBINS</code>	Sets the number of histogram bins. The default number is 101.
-----------------------------	---

## 2.8.12. RUNSTAT - Running statistical values

### Synopsis

`<operator>,nts ifile ofile`

### Description

This module computes running statistical values over a selected number of timesteps. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of a selected number of consecutive timesteps read from ifile is written to ofile. The time stamp in ofile is from the middle contributing timestep of ifile.

### Operators

<b>runmin</b>	Running minimum $o(t + (nts - 1)/2, x) = \mathbf{min}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runmax</b>	Running maximum $o(t + (nts - 1)/2, x) = \mathbf{max}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runsum</b>	Running sum $o(t + (nts - 1)/2, x) = \mathbf{sum}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runmean</b>	Running mean $o(t + (nts - 1)/2, x) = \mathbf{mean}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runavg</b>	Running average $o(t + (nts - 1)/2, x) = \mathbf{avg}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runstd</b>	Running standard deviation Divisor is n. $o(t + (nts - 1)/2, x) = \mathbf{std}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runstd1</b>	Running standard deviation Divisor is (n-1). $o(t + (nts - 1)/2, x) = \mathbf{std1}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runvar</b>	Running variance Divisor is n. $o(t + (nts - 1)/2, x) = \mathbf{var}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runvar1</b>	Running variance Divisor is (n-1). $o(t + (nts - 1)/2, x) = \mathbf{var1}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$

### Parameter

`nts`     **INTEGER**     Number of timesteps

### Environment

`CDO_TIMESTAT_DATE`     Sets the time stamp in ofile to the "first", "middle" or "last" contributing timestep of ifile.

## Example

To compute the running mean over 9 timesteps use:

```
cdo runmean,9 ifile ofile
```

## 2.8.13. RUNPCTL - Running percentile values

### Synopsis

```
runpctl,p,nts ifile ofile
```

### Description

This module computes running percentiles over a selected number of timesteps in *ifile*. The time stamp in *ofile* is from the middle contributing timestep of *ifile*.

$$o(t + (nts - 1)/2, x) = \mathbf{pth\ percentile}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$$

### Parameter

<i>p</i>	FLOAT	Percentile number in 0, ..., 100
<i>nts</i>	INTEGER	Number of timesteps

## Example

To compute the running 50th percentile (median) over 9 timesteps use:

```
cdo runpctl,50,9 ifile ofile
```

## 2.8.14. TIMSTAT - Statistical values over all timesteps

### Synopsis

`<operator> ifile ofile`

### Description

This module computes statistical values over all timesteps in `ifile`. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of all timesteps read from `ifile` is written to `ofile`. The time stamp in `ofile` is from the middle contributing timestep of `ifile`.

### Operators

<b>timmin</b>	Time minimum $o(1, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timmax</b>	Time maximum $o(1, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timsum</b>	Time sum $o(1, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timmean</b>	Time mean $o(1, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timavg</b>	Time average $o(1, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timstd</b>	Time standard deviation Divisor is n. $o(1, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timstd1</b>	Time standard deviation Divisor is (n-1). $o(1, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timvar</b>	Time variance Divisor is n. $o(1, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timvar1</b>	Time variance Divisor is (n-1). $o(1, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the mean over all input timesteps use:

```
cdo timmean ifile ofile
```

## 2.8.15. TIMPCTL - Percentile values over all timesteps

### Synopsis

```
timpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator computes percentiles over all timesteps in `ifile1`. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` should be the result of corresponding `timmin` and `timmax` operations, respectively. The time stamp in `ofile` is from the middle contributing timestep of `ifile1`.

$$o(1,x) = \text{pth percentile}\{i(t',x), t_1 < t' \leq t_n\}$$

### Parameter

`p`      FLOAT      Percentile number in 0, ..., 100

### Environment

`CDO_PCTL_NBINS`      Sets the number of histogram bins. The default number is 101.

### Example

To compute the 90th percentile over all input timesteps use:

```
cdo timmin ifile minfile
cdo timmax ifile maxfile
cdo timpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo timpctl,90 ifile -timmin ifile -timmax ifile ofile
```

## 2.8.16. HOURSTAT - Hourly statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over timesteps of the same hour. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of timesteps of the same hour is written to ofile. The time stamp in ofile is from the middle contributing timestep of ifile.

### Operators

<b>hourmin</b>	Hourly minimum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourmax</b>	Hourly maximum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hoursum</b>	Hourly sum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourmean</b>	Hourly mean For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>houravg</b>	Hourly average For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourstd</b>	Hourly standard deviation Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourstd1</b>	Hourly standard deviation Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourvar</b>	Hourly variance Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourvar1</b>	Hourly variance Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is: $o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the hourly mean of a time series use:

```
cdo hourmean ifile ofile
```

## 2.8.17. HOURPCTL - Hourly percentile values

### Synopsis

```
hourpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator computes percentiles over all timesteps of the same hour in `ifile1`. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` should be the result of corresponding [hourmin](#) and [hourmax](#) operations, respectively. The time stamp in `ofile` is from the middle contributing timestep of `ifile1`.

For every adjacent sequence  $t_1, \dots, t_n$  of timesteps of the same hour it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

<code>p</code>	FLOAT	Percentile number in 0, ..., 100
----------------	-------	----------------------------------

### Environment

<code>CDO_PCTL_NBINS</code>	Sets the number of histogram bins. The default number is 101.
-----------------------------	---

### Example

To compute the hourly 90th percentile of a time series use:

```
cdo hourmin ifile minfile
cdo hourmax ifile maxfile
cdo hourpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo hourpctl,90 ifile -hourmin ifile -hourmax ifile ofile
```



## 2.8.18. DAYSTAT - Daily statistical values

### Synopsis

`<operator> ifile ofile`

### Description

This module computes statistical values over timesteps of the same day. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of timesteps of the same day is written to ofile. The time stamp in ofile is from the middle contributing timestep of ifile.

### Operators

<b>daymin</b>	Daily minimum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daymax</b>	Daily maximum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daysum</b>	Daily sum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daymean</b>	Daily mean For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>dayavg</b>	Daily average For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daystd</b>	Daily standard deviation Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daystd1</b>	Daily standard deviation Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$
<b>dayvar</b>	Daily variance Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>dayvar1</b>	Daily variance Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is: $o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the daily mean of a time series use:

```
cdo daymean ifile ofile
```

## 2.8.19. DAYPCTL - Daily percentile values

### Synopsis

```
daypctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator computes percentiles over all timesteps of the same day in ifile1. The algorithm uses histograms with minimum and maximum bounds given in ifile2 and ifile3, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable CDO\_PCTL\_NBINS. The files ifile2 and ifile3 should be the result of corresponding [daymin](#) and [daymax](#) operations, respectively. The time stamp in ofile is from the middle contributing timestep of ifile1.

For every adjacent sequence  $t_1, \dots, t_n$  of timesteps of the same day it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

$p$       FLOAT      Percentile number in 0, ..., 100

### Environment

CDO\_PCTL\_NBINS      Sets the number of histogram bins. The default number is 101.

### Example

To compute the daily 90th percentile of a time series use:

```
cdo daymin ifile minfile
cdo daymax ifile maxfile
cdo daypctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo daypctl,90 ifile -daymin ifile -daymax ifile ofile
```

## 2.8.20. MONSTAT - Monthly statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over timesteps of the same month. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of timesteps of the same month is written to ofile. The time stamp in ofile is from the middle contributing timestep of ifile.

### Operators

<b>monmin</b>	Monthly minimum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monmax</b>	Monthly maximum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monsum</b>	Monthly sum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monmean</b>	Monthly mean For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monavg</b>	Monthly average For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monstd</b>	Monthly standard deviation Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monstd1</b>	Monthly standard deviation Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monvar</b>	Monthly variance Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monvar1</b>	Monthly variance Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same month it is: $o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the monthly mean of a time series use:

```
cdo monmean ifile ofile
```

## 2.8.21. MONPCTL - Monthly percentile values

### Synopsis

```
monpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator computes percentiles over all timesteps of the same month in *ifile1*. The algorithm uses histograms with minimum and maximum bounds given in *ifile2* and *ifile3*, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files *ifile2* and *ifile3* should be the result of corresponding [monmin](#) and [monmax](#) operations, respectively. The time stamp in *ofile* is from the middle contributing timestep of *ifile1*.

For every adjacent sequence  $t_1, \dots, t_n$  of timesteps of the same month it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

*p*      FLOAT      Percentile number in 0, ..., 100

### Environment

`CDO_PCTL_NBINS`      Sets the number of histogram bins. The default number is 101.

### Example

To compute the monthly 90th percentile of a time series use:

```
cdo monmin ifile minfile
cdo monmax ifile maxfile
cdo monpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo monpctl,90 ifile -monmin ifile -monmax ifile ofile
```

## 2.8.22. YEARMONSTAT - Yearly mean from monthly data

### Synopsis

```
yearmonmean ifile ofile
```

### Description

This operator computes the yearly mean of a monthly time series. Each month is weighted with the number of days per month. The time stamp in `ofile` is from the middle contributing timestep of `ifile`.

For every adjacent sequence  $t_1, \dots, t_n$  of timesteps of the same year it is:  
$$o(t, x) = \text{mean}\{i(t', x), t_1 < t' \leq t_n\}$$

### Environment

CDO_TIMESTAT_DATE	Sets the date information in <code>ofile</code> to the "first", "middle" or "last" contributing timestep of <code>ifile</code> .
-------------------	--

### Example

To compute the yearly mean of a monthly time series use:

```
cdo yearmonmean ifile ofile
```

## 2.8.23. YEARSTAT - Yearly statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over timesteps of the same year. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of timesteps of the same year is written to ofile. The time stamp in ofile is from the middle contributing timestep of ifile.

### Operators

<b>yearmin</b>	Yearly minimum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearmax</b>	Yearly maximum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearsum</b>	Yearly sum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearmean</b>	Yearly mean For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearavg</b>	Yearly average For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearstd</b>	Yearly standard deviation Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearstd1</b>	Yearly standard deviation Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearvar</b>	Yearly variance Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearvar1</b>	Yearly variance Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is: $o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

### Note

The operators yearmean and yearavg compute only arithmetical means!

## Example

To compute the yearly mean of a time series use:

```
cdo yearmean ifile ofile
```

To compute the yearly mean from the correct weighted monthly mean use:

```
cdo yearmonmean ifile ofile
```

## 2.8.24. YEARPCTL - Yearly percentile values

### Synopsis

```
yearpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator computes percentiles over all timesteps of the same year in *ifile1*. The algorithm uses histograms with minimum and maximum bounds given in *ifile2* and *ifile3*, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files *ifile2* and *ifile3* should be the result of corresponding [yearmin](#) and [yearmax](#) operations, respectively. The time stamp in *ofile* is from the middle contributing timestep of *ifile1*.

For every adjacent sequence  $t_1, \dots, t_n$  of timesteps of the same year it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

<i>p</i>	FLOAT	Percentile number in 0, ..., 100
----------	-------	----------------------------------

### Environment

<code>CDO_PCTL_NBINS</code>	Sets the number of histogram bins. The default number is 101.
-----------------------------	---

## Example

To compute the yearly 90th percentile of a time series use:

```
cdo yearmin ifile minfile
cdo yearmax ifile maxfile
cdo yearpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo yearpctl,90 ifile -yearmin ifile -yearmax ifile ofile
```

## 2.8.25. SEASSTAT - Seasonal statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over timesteps of the same season. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of timesteps of the same season is written to ofile. The time stamp in ofile is from the middle contributing timestep of ifile. Be careful about the first and the last output timestep, they may be incorrect values if the seasons have incomplete timesteps.

### Operators

<b>seasmin</b>	Seasonal minimum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasmax</b>	Seasonal maximum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seassum</b>	Seasonal sum For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasmean</b>	Seasonal mean For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasavg</b>	Seasonal average For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasstd</b>	Seasonal standard deviation Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasstd1</b>	Seasonal standard deviation Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasvar</b>	Seasonal variance Divisor is n. For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasvar1</b>	Seasonal variance Divisor is (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is: $o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the seasonal mean of a time series use:

```
cdo seasmean ifile ofile
```



## 2.8.26. SEASPCTL - Seasonal percentile values

### Synopsis

```
seaspctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator computes percentiles over all timesteps in *ifile1* of the same season. The algorithm uses histograms with minimum and maximum bounds given in *ifile2* and *ifile3*, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files *ifile2* and *ifile3* should be the result of corresponding [seasmin](#) and [seasmax](#) operations, respectively. The time stamp in *ofile* is from the middle contributing timestep of *ifile1*. Be careful about the first and the last output timestep, they may be incorrect values if the seasons have incomplete timesteps.

For every adjacent sequence  $t_1, \dots, t_n$  of timesteps of the same season it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

<i>p</i>	FLOAT	Percentile number in 0, ..., 100
----------	-------	----------------------------------

### Environment

<code>CDO_PCTL_NBINS</code>	Sets the number of histogram bins. The default number is 101.
-----------------------------	---

### Example

To compute the seasonal 90th percentile of a time series use:

```
cdo seasmin ifile minfile
cdo seasmax ifile maxfile
cdo seaspctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo seaspctl,90 ifile -seasmin ifile -seasmax ifile ofile
```

## 2.8.27. YHOURSTAT - Multi-year hourly statistical values

### Synopsis

*<operator>* ifile ofile

### Description

This module computes statistical values of each hour and day of year. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of each hour and day of year in ifile is written to ofile. The date information in an output field is the date of the last contributing input field.

### Operators

<b>yhourmin</b>	Multi-year hourly minimum $o(0001, x) = \mathbf{min}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{min}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourmax</b>	Multi-year hourly maximum $o(0001, x) = \mathbf{max}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{max}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhoursum</b>	Multi-year hourly sum $o(0001, x) = \mathbf{sum}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{sum}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourmean</b>	Multi-year hourly mean $o(0001, x) = \mathbf{mean}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{mean}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhouravg</b>	Multi-year hourly average $o(0001, x) = \mathbf{avg}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{avg}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourstd</b>	Multi-year hourly standard deviation Divisor is n. $o(0001, x) = \mathbf{std}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{std}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourstd1</b>	Multi-year hourly standard deviation Divisor is (n-1). $o(0001, x) = \mathbf{std1}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{std1}\{i(t, x), \text{day}(i(t)) = 8784\}$

<b>yhourvar</b>	Multi-year hourly variance Divisor is n. $o(0001, x) = \mathbf{var}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{var}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourvar1</b>	Multi-year hourly variance Divisor is (n-1). $o(0001, x) = \mathbf{var1}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{var1}\{i(t, x), \text{day}(i(t)) = 8784\}$

## 2.8.28. YDAYSTAT - Multi-year daily statistical values

### Synopsis

*<operator>* ifile ofile

### Description

This module computes statistical values of each day of year. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of each day of year in ifile is written to ofile. The date information in an output field is the date of the last contributing input field.

### Operators

<b>ydaymin</b>	Multi-year daily minimum $o(001, x) = \mathbf{min}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{min}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaymax</b>	Multi-year daily maximum $o(001, x) = \mathbf{max}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{max}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaysum</b>	Multi-year daily sum $o(001, x) = \mathbf{sum}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{sum}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaymean</b>	Multi-year daily mean $o(001, x) = \mathbf{mean}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{mean}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydayavg</b>	Multi-year daily average $o(001, x) = \mathbf{avg}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{avg}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaystd</b>	Multi-year daily standard deviation Divisor is n. $o(001, x) = \mathbf{std}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{std}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaystd1</b>	Multi-year daily standard deviation Divisor is (n-1). $o(001, x) = \mathbf{std1}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{std1}\{i(t, x), \text{day}(i(t)) = 366\}$

<b>ydayvar</b>	Multi-year daily variance Divisor is n. $o(001, x) = \mathbf{var}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{var}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydayvar1</b>	Multi-year daily variance Divisor is (n-1). $o(001, x) = \mathbf{var1}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \mathbf{var1}\{i(t, x), \text{day}(i(t)) = 366\}$

## Example

To compute the daily mean over all input years use:

```
cdo ydaymean ifile ofile
```

## 2.8.29. YDAYPCTL - Multi-year daily percentile values

### Synopsis

```
ydaypctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator writes a certain percentile of each day of year in ifile1 to ofile. The algorithm uses histograms with minimum and maximum bounds given in ifile2 and ifile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO\_PCTL\_NBINS to a different value. The files ifile2 and ifile3 should be the result of corresponding [ydaymin](#) and [ydaymax](#) operations, respectively. The date information in an output field is the date of the last contributing input field.

$$\begin{aligned} o(001, x) &= \text{pth percentile}\{i(t, x), \text{day}(i(t)) = 001\} \\ &\vdots \\ o(366, x) &= \text{pth percentile}\{i(t, x), \text{day}(i(t)) = 366\} \end{aligned}$$

### Parameter

*p*      FLOAT      Percentile number in 0, ..., 100

### Environment

CDO\_PCTL\_NBINS      Sets the number of histogram bins. The default number is 101.

### Example

To compute the daily 90th percentile over all input years use:

```
cdo ydaymin ifile minfile
cdo ydaymax ifile maxfile
cdo ydaypctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo ydaypctl,90 ifile -ydaymin ifile -ydaymax ifile ofile
```

## 2.8.30. YMONSTAT - Multi-year monthly statistical values

### Synopsis

`<operator> ifile ofile`

### Description

This module computes statistical values of each month of year. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of each month of year in `ifile` is written to `ofile`. The date information in an output field is the date of the last contributing input field.

### Operators

<b>ymonmin</b>	Multi-year monthly minimum $o(01, x) = \min\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \min\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonmax</b>	Multi-year monthly maximum $o(01, x) = \max\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \max\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonsum</b>	Multi-year monthly sum $o(01, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonmean</b>	Multi-year monthly mean $o(01, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonavg</b>	Multi-year monthly average $o(01, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonstd</b>	Multi-year monthly standard deviation Divisor is n. $o(01, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonstd1</b>	Multi-year monthly standard deviation Divisor is (n-1). $o(01, x) = \text{std1}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{std1}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonvar</b>	Multi-year monthly variance Divisor is n.

$$o(01, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 01\}$$

$$\vdots$$

$$o(12, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 12\}$$

**ymonvar1**      Multi-year monthly variance  
 Divisor is (n-1).

$$o(01, x) = \mathbf{var1}\{i(t, x), \text{month}(i(t)) = 01\}$$

$$\vdots$$

$$o(12, x) = \mathbf{var1}\{i(t, x), \text{month}(i(t)) = 12\}$$

### Example

To compute the monthly mean over all input years use:

```
cdo ymonmean ifile ofile
```



## 2.8.31. YMONPCTL - Multi-year monthly percentile values

### Synopsis

```
ymonpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator writes a certain percentile of each month of year in ifile1 to ofile. The algorithm uses histograms with minimum and maximum bounds given in ifile2 and ifile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO\_PCTL\_NBINS to a different value. The files ifile2 and ifile3 should be the result of corresponding [ymonmin](#) and [ymonmax](#) operations, respectively. The date information in an output field is the date of the last contributing input field.

$$\begin{aligned} o(01, x) &= \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 01\} \\ &\vdots \\ o(12, x) &= \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 12\} \end{aligned}$$

### Parameter

*p*      FLOAT      Percentile number in 0, ..., 100

### Environment

CDO\_PCTL\_NBINS      Sets the number of histogram bins. The default number is 101.

### Example

To compute the monthly 90th percentile over all input years use:

```
cdo ymonmin ifile minfile
cdo ymonmax ifile maxfile
cdo ymonpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo ymonpctl,90 ifile -ymonmin ifile -ymonmax ifile ofile
```

## 2.8.32. YSEASSTAT - Multi-year seasonal statistical values

### Synopsis

`<operator> ifile ofile`

### Description

This module computes statistical values of each season. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of each season in `ifile` is written to `ofile`. The date information in an output field is the date of the last contributing input field.

### Operators

<b>yseasmin</b>	Multi-year seasonal minimum $o(1, x) = \min\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \min\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \min\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \min\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasmax</b>	Multi-year seasonal maximum $o(1, x) = \max\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \max\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \max\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \max\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseassum</b>	Multi-year seasonal sum $o(1, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasmean</b>	Multi-year seasonal mean $o(1, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasavg</b>	Multi-year seasonal average $o(1, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasstd</b>	Multi-year seasonal standard deviation $o(1, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasstd1</b>	Multi-year seasonal standard deviation $o(1, x) = \text{std1}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{std1}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{std1}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{std1}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$

<b>yseasvar</b>	Multi-year seasonal variance $o(1, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasvar1</b>	Multi-year seasonal variance $o(1, x) = \text{var1}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{var1}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{var1}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{var1}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$

## Example

To compute the seasonal mean over all input years use:

```
cdo yseasmean ifile ofile
```

## 2.8.33. YSEASPCTL - Multi-year seasonal percentile values

### Synopsis

```
yseaspctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This operator writes a certain percentile of each season in ifile1 to ofile. The algorithm uses histograms with minimum and maximum bounds given in ifile2 and ifile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO\_PCTL\_NBINS to a different value. The files ifile2 and ifile3 should be the result of corresponding [yseasmin](#) and [yseasmax](#) operations, respectively. The date information in an output field is the date of the last contributing input field.

$$o(1, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$$

$$o(2, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$$

$$o(3, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$$

$$o(4, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$$

### Parameter

*p*      FLOAT      Percentile number in 0, ..., 100

### Environment

CDO\_PCTL\_NBINS      Sets the number of histogram bins. The default number is 101.

### Example

To compute the seasonal 90th percentile over all input years use:

```
cdo yseasmin ifile minfile
cdo yseasmax ifile maxfile
cdo yseaspctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo yseaspctl,90 ifile -yseasmin ifile -yseasmax ifile ofile
```

## 2.8.34. YDRUNSTAT - Multi-year daily running statistical values

### Synopsis

`<operator>,nts ifile ofile`

### Description

This module writes running statistical values for each day of year in ifile to ofile. Depending on the chosen operator, the minimum, maximum, sum, average, variance or standard deviation of all timesteps in running windows of which the medium timestep corresponds to a certain day of year is computed. The date information in an output field is the date of the timestep in the middle of the last contributing running window. Note that the operator have to be applied to a continuous time series of daily measurements in order to yield physically meaningful results. Also note that the output time series begins  $(nts-1)/2$  timesteps after the first timestep of the input time series and ends  $(nts-1)/2$  timesteps before the last one. For input data which are complete but not continuous, such as time series of daily measurements for the same month or season within different years, the operator yields physically meaningful results only if the input time series does include the  $(nts-1)/2$  days before and after each period of interest.

### Operators

<b>ydrunmin</b>	Multi-year daily running minimum $o(001, x) = \min\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \min\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunmax</b>	Multi-year daily running maximum $o(001, x) = \max\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \max\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunsum</b>	Multi-year daily running sum $o(001, x) = \sum\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \sum\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunmean</b>	Multi-year daily running mean $o(001, x) = \text{mean}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{mean}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunavg</b>	Multi-year daily running average $o(001, x) = \text{avg}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{avg}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunstd</b>	Multi-year daily running standard deviation Divisor is n. $o(001, x) = \text{std}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{std}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunstd1</b>	Multi-year daily running standard deviation Divisor is (n-1).

<b>ydrunvar</b>	$o(001, x) = \mathbf{std1}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x); \text{day}[(i(t + (nts - 1)/2)] = 001\}$
	$\vdots$
	$o(366, x) = \mathbf{std1}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x); \text{day}[(i(t + (nts - 1)/2)] = 366\}$
	Multi-year daily running variance Divisor is n.
<b>ydrunvar1</b>	$o(001, x) = \mathbf{var}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x); \text{day}[(i(t + (nts - 1)/2)] = 001\}$
	$\vdots$
	$o(366, x) = \mathbf{var}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x); \text{day}[(i(t + (nts - 1)/2)] = 366\}$
	Multi-year daily running variance Divisor is (n-1).
	$o(001, x) = \mathbf{var1}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x); \text{day}[(i(t + (nts - 1)/2)] = 001\}$
	$\vdots$
	$o(366, x) = \mathbf{var1}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x); \text{day}[(i(t + (nts - 1)/2)] = 366\}$

## Parameter

*nts*    INTEGER    Number of timesteps

## Example

Assume the input data provide a continuous time series of daily measurements. To compute the running multi-year daily mean over all input timesteps for a running window of five days use:

```
cdo ydrunmean,5 ifile ofile
```

Note that except for the standard deviation the results of the operators in this module are equivalent to a composition of corresponding operators from the [YDAYSTAT](#) and [RUNSTAT](#) modules. For instance, the above command yields the same result as:

```
cdo ydaymean -runmean,5 ifile ofile
```

## 2.8.35. YDRUNPCTL - Multi-year daily running percentile values

### Synopsis

```
ydrunpctl,p,nts ifile1 ifile2 ifile3 ofile
```

### Description

This operator writes running percentile values for each day of year in ifile1 to ofile. A certain percentile is computed for all timesteps in running windows of which the medium timestep corresponds to a certain day of year. The algorithm uses histograms with minimum and maximum bounds given in ifile2 and ifile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO\_PCTL\_NBINS to a different value. The files ifile2 and ifile3 should be the result of corresponding [ydrunmin](#) and [ydrunmax](#) operations, respectively. The date information in an output field is the date of the timestep in the middle of the last contributing running window. Note that the operator have to be applied to a continuous time series of daily measurements in order to yield physically meaningful results. Also note that the output time series begins (nts-1)/2 timesteps after the first timestep of the input time series and ends (nts-1)/2 timesteps before the last. For input data which are complete but not continuous, such as time series of daily measurements for the same month or season within different years, the operator only yields physically meaningful results if the input time series does include the (nts-1)/2 days before and after each period of interest.

$$\begin{aligned}
 o(001, x) &= \text{pth percentile}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\} \\
 &\quad \vdots \\
 o(366, x) &= \text{pth percentile}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}
 \end{aligned}$$

### Parameter

<i>p</i>	FLOAT	Percentile number in 0, ..., 100
<i>nts</i>	INTEGER	Number of timesteps

### Environment

CDO_PCTL_NBINS	Sets the number of histogram bins. The default number is 101.
----------------	---

### Example

Assume the input data provide a continuous time series of daily measurements. To compute the running multi-year daily 90th percentile over all input timesteps for a running window of five days use:

```
cdo ydrunmin,5 ifile minfile
cdo ydrunmax,5 ifile maxfile
cdo ydrunpctl,90,5 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo ydrunpctl,90,5 ifile -ydrunmin ifile -ydrunmax ifile ofile
```

## 2.9. Correlation and co.

This sections contains modules for correlation and co. in grid space and over time.  
In this section the abbreviations as in the following table are used:

Covariance <b>covar</b>	$n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2 (y_i - \bar{y})^2$
<b>covar</b> weighted by $\{w_i, i = 1, \dots, n\}$	$\left( \sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left( x_i - \left( \sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right) \left( y_i - \left( \sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j y_j \right)$

Here is a short overview of all operators in this section:

<b>fldcor</b>	Correlation in grid space
<b>timcor</b>	Correlation over time
<b>fldcovar</b>	Covariance in grid space
<b>timcovar</b>	Covariance over time

### 2.9.1. FLDCOR - Correlation in grid space

#### Synopsis

```
fldcor ifile1 ifile2 ofile
```

#### Description

The correlation coefficient is a quantity that gives the quality of a least squares fitting to the original data. This operator correlates all gridpoints of two fields for each timestep. With

$$S(t) = \{x, i_1(t, x) \neq missval \wedge i_2(t, x) \neq missval\}$$

it is

$$o(t, 1) = \frac{\sum_{x \in S(t)} i_1(t, x) i_2(t, x) w(x) - \overline{i_1(t, x)} \overline{i_2(t, x)} \sum_{x \in S(t)} w(x)}{\sqrt{\left( \sum_{x \in S(t)} i_1(t, x)^2 w(x) - \overline{i_1(t, x)}^2 \sum_{x \in S(t)} w(x) \right) \left( \sum_{x \in S(t)} i_2(t, x)^2 w(x) - \overline{i_2(t, x)}^2 \sum_{x \in S(t)} w(x) \right)}}$$

where  $w(x)$  are the area weights obtained by the input streams. For every timestep  $t$  only those field elements  $x$  belong to the sample, which have  $i_1(t, x) \neq missval$  and  $i_2(t, x) \neq missval$ .

### 2.9.2. TIMCOR - Correlation over time

#### Synopsis

```
timcor ifile1 ifile2 ofile
```

#### Description

The correlation coefficient is a quantity that gives the quality of a least squares fitting to the original data. This operator correlates each gridpoint of two fields over all timesteps. With

$$S(x) = \{t, i_1(t, x) \neq missval \wedge i_2(t, x) \neq missval\}$$

it is

$$o(1, x) = \frac{\sum_{t \in S(x)} i_1(t, x) i_2(t, x) - n \overline{i_1(t, x)} \overline{i_2(t, x)}}{\sqrt{\left( \sum_{t \in S(x)} i_1(t, x)^2 - n \overline{i_1(t, x)}^2 \right) \left( \sum_{t \in S(x)} i_2(t, x)^2 - n \overline{i_2(t, x)}^2 \right)}}$$

For every gridpoint  $x$  only those timesteps  $t$  belong to the sample, which have  $i_1(t, x) \neq missval$  and  $i_2(t, x) \neq missval$ .



### 2.9.3. FLDCOVAR - Covariance in grid space

#### Synopsis

```
fldcovar ifile1 ifile2 ofile
```

#### Description

This operator calculates the covariance of two fields over all gridpoints for each timestep. With

$$S(t) = \{x, i_1(t, x) \neq missval \wedge i_2(t, x) \neq missval\}$$

it is

$$o(t, 1) = \left( \sum_{x \in S(t)} w(x) \right)^{-1} \sum_{x \in S(t)} w(x) \left( i_1(t, x) - \frac{\sum_{x \in S(t)} w(x) i_1(t, x)}{\sum_{x \in S(t)} w(x)} \right) \left( i_2(t, x) - \frac{\sum_{x \in S(t)} w(x) i_2(t, x)}{\sum_{x \in S(t)} w(x)} \right)$$

where  $w(x)$  are the area weights obtained by the input streams. For every timestep  $t$  only those field elements  $x$  belong to the sample, which have  $i\_1(t, x) \neq missval$  and  $i\_2(t, x) \neq missval$ .

### 2.9.4. TIMCOVAR - Covariance over time

#### Synopsis

```
timcovar ifile1 ifile2 ofile
```

#### Description

This operator calculates the covariance of two fields at each gridpoint over all timesteps. With

$$S(x) = \{t, i_1(t, x) \neq missval \wedge i_2(t, x) \neq missval\}$$

it is

$$o(1, x) = n^{-1} \sum_{t \in S(x)} \left( i_1(t, x) - \overline{i_1(t, x)} \right)^2 \left( i_2(t, x) - \overline{i_2(t, x)} \right)^2$$

For every gridpoint  $x$  only those timesteps  $t$  belong to the sample, which have  $i\_1(t, x) \neq missval$  and  $i\_2(t, x) \neq missval$ .

## 2.10. Regression

This sections contains modules for linear regression of time series.

Here is a short overview of all operators in this section:

<b>regres</b>	Regression
<b>detrend</b>	Detrend
<b>trend</b>	Trend
<b>subtrend</b>	Subtract trend

### 2.10.1. REGRES - Regression

#### Synopsis

```
regres ifile ofile
```

#### Description

The values of the input file `ifile` are assumed to be distributed as  $N(a + bt, \sigma^2)$  with unknown  $a$ ,  $b$  and  $\sigma^2$ . This operator estimates the parameter  $b$ . For every field element  $x$  only those timesteps  $t$  belong to the sample  $S(x)$ , which have  $i(t, x) \neq \text{miss}$ . It is

$$o(1, x) = \frac{\sum_{t \in S(x)} \left( i(t, x) - \frac{1}{\#S(x)} \sum_{t' \in S(x)} i(t', x) \right) \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)}{\sum_{t \in S(x)} \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)^2}$$

### 2.10.2. DETREND - Detrend time series

#### Synopsis

```
detrend ifile ofile
```

#### Description

Every time series in `ifile` is linearly detrended. For every field element  $x$  only those timesteps  $t$  belong to the sample  $S(x)$ , which have  $i(t, x) \neq \text{miss}$ . With

$$a(x) = \frac{1}{\#S(x)} \sum_{t \in S(x)} i(t, x) - b(x) \left( \frac{1}{\#S(x)} \sum_{t \in S(x)} t \right)$$

and

$$b(x) = \frac{\sum_{t \in S(x)} \left( i(t, x) - \frac{1}{\#S(x)} \sum_{t' \in S(x)} i(t', x) \right) \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)}{\sum_{t \in S(x)} \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)^2}$$

it is

$$o(t, x) = i(t, x) - (a(x) + b(x)t)$$

#### Note

This operator has to keep the fields of all timesteps concurrently in the memory. If not enough memory is available use the operators [trend](#) and [subtrend](#).

#### Example

To detrend the data in `ifile` and to store the detrended data in `ofile` use:

```
cdo detrend ifile ofile
```

### 2.10.3. TREND - Trend of time series

#### Synopsis

```
trend ifile ofile1 ofile2
```

#### Description

The values of the input file `ifile` are assumed to be distributed as  $N(a + bt, \sigma^2)$  with unknown  $a$ ,  $b$  and  $\sigma^2$ . This operator estimates the parameter  $a$  and  $b$ . For every field element  $x$  only those timesteps  $t$  belong to the sample  $S(x)$ , which have  $i(t, x) \neq \text{miss}$ . It is

$$o_1(1, x) = \frac{1}{\#S(x)} \sum_{t \in S(x)} i(t, x) - b(x) \left( \frac{1}{\#S(x)} \sum_{t \in S(x)} t \right)$$

and

$$o_2(1, x) = \frac{\sum_{t \in S(x)} \left( i(t, x) - \frac{1}{\#S(x)} \sum_{t' \in S(x)} i(t', x) \right) \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)}{\sum_{t \in S(x)} \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)^2}$$

Thus the estimation for  $a$  is stored in `ofile1` and that for  $b$  is stored in `ofile2`. To subtract the trend from the data see operator [subtrend](#).

### 2.10.4. SUBTREND - Subtract a trend

#### Synopsis

```
subtrend ifile1 ifile2 ifile3 ofile
```

#### Description

This operator is for subtracting a trend computed by the operator [trend](#). It is

$$o(t, x) = i_1(t, x) - (i_2(1, x) + i_3(1, x) \cdot t)$$

where  $t$  is the timesteps.

#### Example

The typical call for detrending the data in `ifile` and storing the detrended data in `ofile` is:

```
cdo trend ifile afile bfile
cdo subtrend ifile afile bfile ofile
```

The result is identical to a call of the operator [detrend](#):

```
cdo detrend ifile ofile
```

## 2.11. EOFs

This section contains modules to compute Empirical Orthogonal Functions and - once they are computed - their principal coefficients.

An introduction to the theory of principal component analysis as applied here can be found in:

Principal Component Analysis [Peisendorfer]

Details about calculation in the time- and spatial spaces are found in:

Statistical Analysis in Climate Research [vonStorch]

EOFs are defined as the eigen values of the scatter matrix (covariance matrix) of the data. For the sake of simplicity, samples are regarded as **time series of anomalies**

$$(z(t)), t \in \{1, \dots, n\}$$

of (column-) vectors  $z(t)$  with  $p$  entries (where  $p$  is the gridsize). Thus, using the fact, that  $z_j(t)$  are anomalies, i.e.

$$\langle z_j \rangle = n^{-1} \sum_{i=1}^n z_j(i) = 0 \quad \forall 1 \leq j \leq p$$

the scatter matrix  $\mathbf{S}$  can be written as

$$\mathbf{S} = \sum_{t=1}^n \left[ \sqrt{\mathbf{W}} z(t) \right] \left[ \sqrt{\mathbf{W}} z(t) \right]^T$$

where  $\mathbf{W}$  is the diagonal matrix containing the area weight of cell  $p_0$  in  $z$  at  $\mathbf{W}(x, x)$ .

The matrix  $\mathbf{S}$  has a set of orthonormal eigenvectors  $e_j, j = 1, \dots, p$ , which are called *empirical orthogonal functions (EOFs) of the sample  $z$* . (Please note, that  $e_j$  is the eigenvector of  $\mathbf{S}$  and not the weighted eigen-vector which would be  $\mathbf{W}e_j$ .) Let the corresponding eigenvalues be denoted  $\lambda_j$ . The vectors  $e_j$  are spatial patterns which explain a certain amount of variance of the time series  $z(t)$  that is related linearly to  $\lambda_j$ . Thus, the spatial pattern defined by the first eigenvector (the one with the largest eigenvalue) is the pattern which explains a maximum possible amount of variance of the sample  $z(t)$ . The orthonormality of eigenvectors reads as

$$\sum_{x=1}^p \left[ \sqrt{\mathbf{W}(x, x)} e_j(x) \right] \left[ \sqrt{\mathbf{W}(x, x)} e_k(x) \right] = \sum_{x=1}^p \mathbf{W}(x, x) e_j(x) e_k(x) = \begin{cases} 0 & \text{if } j \neq k \\ 1 & \text{if } j = k \end{cases}$$

If all EOFs  $e_j$  with  $\lambda_j \neq 0$  are calculated, the data can be reconstructed from

$$z(t, x) = \sum_{j=1}^p \mathbf{W}(x, x) a_j(t) e_j(x)$$

where  $a_j$  are called the *principal components* or *principal coefficients* or *EOF coefficients* of  $z$ . These coefficients - as readily seen from above - are calculated as the projection of an EOF  $e_j$  onto a time step of the data sample  $z(t_0)$  as

$$a_j(t_0) = \sum_{x=1}^p \left[ \sqrt{\mathbf{W}(x, x)} e_j(x) \right] \left[ \sqrt{\mathbf{W}(x, x)} z(t_0, x) \right] = \left[ \sqrt{\mathbf{W}} z(t_0) \right]^T \left[ \sqrt{\mathbf{W}} e_j \right].$$

Here is a short overview of all operators in this section:

<b>eof</b>	Calculate EOFs in spatial or time space
<b>eoftime</b>	Calculate EOFs in time space
<b>eofspatial</b>	Calculate EOFs in spatial space
<b>eof3d</b>	Calculate 3-Dimensional EOFs in time space
<b>eofcoeff</b>	Calculate principal coefficients of EOFs

## 2.11.1. EOFs - Empirical Orthogonal Functions

### Synopsis

```
<operator>,neof ifile ofile1 ofile2
```

### Description

This module calculates empirical orthogonal functions of the data in `ifile` as the eigen values of the scatter matrix (covariance matrix)  $S$  of the data sample  $z(t)$ . A more detailed description can be found above.

**Please note, that the input data are assumed to be anomalies.**

If operator `eof` is chosen, the EOFs are computed in either time or spatial space, whichever is the fastest. If the user already knows, which computation is faster, the module can be forced to perform a computation in time- or gridspace by using the operators `eoftime` or `eofspatial`, respectively. This can enhance performance, especially for very long time series, where the number of timesteps is larger than the number of grid-points. Data in `ifile` are assumed to be anomalies. If they are not, the behavior of this module is **not well defined**. After execution `ofile1` will contain all eigen-values and `ofile2` the eigenvectors  $e_j$ . All EOFs and eigen-values are computed. However, only the first `neof` EOFs are written to `ofile2`. Nonetheless, `ofile1` contains all eigen-values.

Missing values are not fully supported. Support is only checked for non-changing masks of missing values in time. Although there still will be results, they are not trustworthy, and a warning will occur. In the latter case we suggest to replace missing values by 0 in `ifile`.

### Operators

<code>eof</code>	Calculate EOFs in spatial or time space
<code>eoftime</code>	Calculate EOFs in time space
<code>eofspatial</code>	Calculate EOFs in spatial space
<code>eof3d</code>	Calculate 3-Dimensional EOFs in time space

### Parameter

<code>neof</code>	INTEGER	Number of eigen functions
-------------------	---------	---------------------------

### Environment

<code>CDO_SVD_MODE</code>	Is used to choose the algorithm for eigenvalue calculation. Options are 'jacobi' for a one-sided parallel jacobi-algorithm (only executed in parallel if -P flag is set) and 'danielson_lanczos' for a non-parallel d/l algorithm. The default setting is 'jacobi'.
<code>CDO_WEIGHT_MODE</code>	It is used to set the weight mode. The default is 'on'. Set it to 'off' for a non weighted version.
<code>MAX_JACOBI_ITER</code>	Is the maximum integer number of annihilation sweeps that is executed if the jacobi-algorithm is used to compute the eigen values. The default value is 12.
<code>FNORM_PRECISION</code>	Is the Frobenius norm of the matrix consisting of an annihilation pair of eigenvectors that is used to determine if the eigenvectors have reached a sufficient level of convergence. If all annihilation-pairs of vectors have a norm below this value, the computation is considered to have converged properly. Otherwise, a warning will occur. The default value 1e-12.

## Example

To calculate the first 40 EOFs of a data-set containing anomalies use:

```
cdo eof,40 ifile ofile1 ofile2
```

If the dataset does not contain anomalies, process them first, and use:

```
cdo sub ifile1 -timmean ifile1 anom_file  
cdo eof,40 anom_file ofile1 ofile2
```

## 2.11.2. EOFCOEFF - Principal coefficients of EOFs

### Synopsis

```
eofcoeff ifile1 ifile2 obase
```

### Description

This module calculates the time series of the principal coefficients for given EOF (empirical orthogonal functions) and data. Time steps in ifile1 are assumed to be the EOFs, time steps in ifile2 are assumed to be the time series. Note, that this operator calculates a weighted dot product of the fields in ifile1 and ifile2. For consistency set the environment variable CDO\_WEIGHT\_MODE=off when using eof or eof3d. Given a set of EOFs  $e_j$  and a time series of data  $z(t)$  with  $p$  entries for each timestep from which  $e_j$  have been calculated, this operator calculates the time series of the projections of data onto each EOF

$$o_j(t) = \sum_{x=1}^p W(x, x) z(t, x) e_j(x)$$

where  $W$  is the diagonal matrix containing area weights as above. There will be a separate file  $o_j$  for the principal coefficients of each EOF.

As the EOFs  $e_j$  are uncorrelated, so are their principal coefficients, i.e.

$$\sum_{t=1}^n o_j(t) o_k(t) = \begin{cases} 0 & \text{if } j \neq k \\ \lambda_j & \text{if } j = k \end{cases} \quad \text{with } \sum_{t=1}^n o_j(t) = 0 \forall j \in \{1, \dots, p\}.$$

There will be a separate file containing a time series of principal coefficients with time information from ifile2 for each EOF in ifile1. Output files will be numbered as <obase><neof><suffix> where neof+1 is the number of the EOF (timestep) in ifile1 and suffix is the filename extension derived from the file format.

### Environment

CDO_FILE_SUFFIX	Set the default file suffix. This suffix will be added to the output file names instead of the filename extension derived from the file format. Set this variable to NULL to disable the adding of a file suffix.
-----------------	---

### Example

To calculate principal coefficients of the first 40 EOFs of anom\_file, and write them to files beginning with obase, use:

```
export CDO_WEIGHT_MODE=off
cdo eof,40 anom_file eval_file eof_file
cdo eofcoeff eof_file anom_file obase
```

The principal coefficients of the first EOF will be in the file obase000000.nc (and so forth for higher EOFs,  $n$ th EOF will be in obase<n-1>).

If the dataset ifile does not contain anomalies, process them first, and use:

```
export CDO_WEIGHT_MODE=off
cdo sub ifile -timmean ifile anom_file
cdo eof,40 anom_file eval_file eof_file
cdo eofcoeff eof_file anom_file obase
```



## 2.12. Interpolation

This section contains modules to interpolate datasets. There are several operators to interpolate horizontal fields to a new grid. Some of those operators can handle only 2D fields on a regular rectangular grid. Vertical interpolation of 3D variables is possible from hybrid model levels to height or pressure levels. Interpolation in time is possible between time steps and years.

Here is a short overview of all operators in this section:

<b>remapbil</b>	Bilinear interpolation
<b>genbil</b>	Generate bilinear interpolation weights
<b>remapbic</b>	Bicubic interpolation
<b>genbic</b>	Generate bicubic interpolation weights
<b>remapnn</b>	Nearest neighbor remapping
<b>gennn</b>	Generate nearest neighbor remap weights
<b>remapdis</b>	Distance-weighted average remapping
<b>gendis</b>	Generate distance-weighted average remap weights
<b>remapycon</b>	First order conservative remapping
<b>genycon</b>	Generate 1st order conservative remap weights
<b>remapcon</b>	First order conservative remapping
<b>gencon</b>	Generate 1st order conservative remap weights
<b>remapcon2</b>	Second order conservative remapping
<b>gencon2</b>	Generate 2nd order conservative remap weights
<b>remaplaf</b>	Largest area fraction remapping
<b>genlaf</b>	Generate largest area fraction remap weights
<b>remap</b>	Grid remapping
<b>remapeta</b>	Remap vertical hybrid level
<b>ml2pl</b>	Model to pressure level interpolation
<b>ml2hl</b>	Model to height level interpolation
<b>ap2pl</b>	Model to pressure level interpolation
<b>intlevel</b>	Linear level interpolation
<b>intlevel3d</b>	Linear level interpolation onto a 3d vertical coordinate
<b>intlevelx3d</b>	like intlevel3d but with extrapolation
<b>inttime</b>	Interpolation between timesteps
<b>intntime</b>	Interpolation between timesteps
<b>intyear</b>	Interpolation between two years

## 2.12.1. REMAPBIL - Bilinear interpolation

### Synopsis

```
<operator>,grid ifile ofile
```

### Description

This module contains operators for a bilinear remapping of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see [\[SCRIP\]](#). This interpolation method only works on quadrilateral curvilinear source grids.

### Operators

<b>remapbil</b>	Bilinear interpolation Performs a bilinear interpolation on all input fields.
<b>genbil</b>	Generate bilinear interpolation weights Generates bilinear interpolation weights for the first input field and writes the result to a file. The format of this file is netCDF following the SCRIP convention. Use the operator <a href="#">remap</a> to apply this remapping weights to a data file with the same source grid.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Environment

REMAP_EXTRAPOLATE	This variable is used to switch the extrapolation feature 'on' or 'off'. By default the extrapolation is enabled for circular grids.
-------------------	--

### Example

Say *ifile* contains fields on a quadrilateral curvilinear grid. To remap all fields bilinear to a Gaussian N32 grid, type:

```
cdo remapbil,n32 ifile ofile
```

## 2.12.2. REMAPBIC - Bicubic interpolation

### Synopsis

```
<operator>,grid ifile ofile
```

### Description

This module contains operators for a bicubic remapping of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see [\[SCRIP\]](#). This interpolation method only works on quadrilateral curvilinear source grids.

### Operators

<b>remapbic</b>	Bicubic interpolation Performs a bicubic interpolation on all input fields.
<b>genbic</b>	Generate bicubic interpolation weights Generates bicubic interpolation weights for the first input field and writes the result to a file. The format of this file is netCDF following the SCRIP convention. Use the operator <a href="#">remap</a> to apply this remapping weights to a data file with the same source grid.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Environment

REMAP_EXTRAPOLATE	This variable is used to switch the extrapolation feature 'on' or 'off'. By default the extrapolation is enabled for circular grids.
-------------------	--

### Example

Say *ifile* contains fields on a quadrilateral curvilinear grid. To remap all fields bicubic to a Gaussian N32 grid, type:

```
cdo remapbic,n32 ifile ofile
```

### 2.12.3. REMAPNN - Nearest neighbor remapping

#### Synopsis

`<operator>,grid ifile ofile`

#### Description

This module contains operators for a nearest neighbor remapping of fields between grids in spherical coordinates.

#### Operators

<b>remapnn</b>	Nearest neighbor remapping Performs a nearest neighbor remapping on all input fields.
<b>gennn</b>	Generate nearest neighbor remap weights Generates nearest neighbor remapping weights for the first input field and writes the result to a file. The format of this file is netCDF following the SCRIP convention. Use the operator <a href="#">remap</a> to apply this remapping weights to a data file with the same source grid.

#### Parameter

`grid`     `STRING`     Target grid description file or name

#### Environment

<code>REMAP_EXTRAPOLATE</code>	This variable is used to switch the extrapolation feature 'on' or 'off'. By default the extrapolation is enabled for this remapping method.
<code>CDO_GRIDSEARCH_RADIUS</code>	Grid search radius in degree, default 180 degree.

## 2.12.4. REMAPDIS - Distance-weighted average remapping

### Synopsis

```
remapdis,grid[,neighbors] ifile ofile
```

```
gendis,grid ifile ofile
```

### Description

This module contains operators for a distance-weighted average remapping of the four nearest neighbor values of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see [\[SCRIP\]](#).

### Operators

<b>remapdis</b>	Distance-weighted average remapping Performs a distance-weighted average remapping of the nearest neighbors value on all input fields. The default number of nearest neighbors is 4.
<b>gendis</b>	Generate distance-weighted average remap weights Generates distance-weighted average remapping weights of the four nearest neighbor values for the first input field and writes the result to a file. The format of this file is netCDF following the SCRIP convention. Use the operator <a href="#">remap</a> to apply this remapping weights to a data file with the same source grid.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
<i>neighbors</i>	INTEGER	Number of nearest neighbors

### Environment

REMAP_EXTRAPOLATE	This variable is used to switch the extrapolation feature 'on' or 'off'. By default the extrapolation is enabled for this remapping method.
CDO_GRIDSEARCH_RADIUS	Grid search radius in degree, default 180 degree.

## 2.12.5. REMAPYCON - First order conservative remapping

### Synopsis

```
<operator>,grid ifile ofile
```

### Description

This module contains operators for a first order conservative remapping of fields between grids in spherical coordinates. The operators in this module uses code from the YAC software package to compute the conservative remapping weights. For a detailed description of the interpolation method see [YAC]. The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for the conservative remapping requires that no grid cell occurs more than once.

### Operators

<b>remapycon</b>	First order conservative remapping Performs a first order conservative remapping on all input fields.
<b>genycon</b>	Generate 1st order conservative remap weights Generates first order conservative remapping weights for the first input field and writes the result to a file. The format of this file is netCDF following the SCRIP convention. Use the operator <a href="#">remap</a> to apply this remapping weights to a data file with the same source grid.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Environment

CDO_REMAP_NORM	This variable is used to choose the normalization of the conservative interpolation. By default CDO_REMAP_NORM is set to 'fracarea'. 'fracarea' uses the sum of the non-masked source cell intersected areas to normalize each target cell field value. This results in a reasonable flux value but the flux is not locally conserved. The option 'destarea' uses the total target cell area to normalize each target cell field value. Local flux conservation is ensured, but unreasonable flux values may result.
REMAP_AREA_MIN	This variable is used to set the minimum destination area fraction. The default of this variable is 0.0.

### Example

Say ifile contains fields on a quadrilateral curvilinear grid. To remap all fields conservative to a Gaussian N32 grid, type:

```
cdo remapycon,n32 ifile ofile
```

## 2.12.6. REMAPCON - First order conservative remapping

### Synopsis

```
<operator>,grid ifile ofile
```

### Description

This module contains operators for a first order conservative remapping of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see [SCRIP]. The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for the conservative remapping requires that no grid cell occurs more than once.

### Operators

<b>remapcon</b>	First order conservative remapping Performs a first order conservative remapping on all input fields.
<b>gencon</b>	Generate 1st order conservative remap weights Generates first order conservative remapping weights for the first input field and writes the result to a file. The format of this file is netCDF following the SCRIP convention. Use the operator <a href="#">remap</a> to apply this remapping weights to a data file with the same source grid.

### Parameter

*grid*     STRING     Target grid description file or name

### Environment

CDO_REMAP_NORM	This variable is used to choose the normalization of the conservative interpolation. By default CDO_REMAP_NORM is set to 'fracarea'. 'fracarea' uses the sum of the non-masked source cell intersected areas to normalize each target cell field value. This results in a reasonable flux value but the flux is not locally conserved. The option 'destarea' uses the total target cell area to normalize each target cell field value. Local flux conservation is ensured, but unreasonable flux values may result.
REMAP_AREA_MIN	This variable is used to set the minimum destination area fraction. The default of this variable is 0.0.

### Note

The SCRIP conservative remapping method doesn't work correctly for some grid combinations. Please use [remapycon](#) or [genycon](#) in case of problems.

### Example

Say *ifile* contains fields on a quadrilateral curvilinear grid. To remap all fields conservative to a Gaussian N32 grid, type:

```
cdo remapcon,n32 ifile ofile
```

## 2.12.7. REMAPCON2 - Second order conservative remapping

### Synopsis

```
remapcon2,grid ifile ofile
```

```
gencon2,grid2 ifile ofile
```

### Description

This module contains operators for a second order conservative remapping of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see [SCRIP]. The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for the conservative remapping requires that no grid cell occurs more than once.

### Operators

<b>remapcon2</b>	Second order conservative remapping Performs a second order conservative remapping on all input fields.
<b>gencon2</b>	Generate 2nd order conservative remap weights Generates second order conservative remapping weights for the first input field and writes the result to a file. The format of this file is netCDF following the SCRIP convention. Use the operator <a href="#">remap</a> to apply this remapping weights to a data file with the same source grid.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Environment

CDO_REMAP_NORM	This variable is used to choose the normalization of the conservative interpolation. By default CDO_REMAP_NORM is set to 'fracarea'. 'fracarea' uses the sum of the non-masked source cell intersected areas to normalize each target cell field value. This results in a reasonable flux value but the flux is not locally conserved. The option 'destarea' uses the total target cell area to normalize each target cell field value. Local flux conservation is ensured, but unreasonable flux values may result.
REMAP_AREA_MIN	This variable is used to set the minimum destination area fraction. The default of this variable is 0.0.

### Note

The SCRIP conservative remapping method doesn't work correctly for some grid combinations.

### Example

Say ifile contains fields on a quadrilateral curvilinear grid. To remap all fields conservative (2nd order) to a Gaussian N32 grid, type:

```
cdo remapcon2,n32 ifile ofile
```



## 2.12.8. REMAPLAF - Largest area fraction remapping

### Synopsis

`<operator>,grid ifile ofile`

### Description

This module contains operators for a largest area fraction remapping of fields between grids in spherical coordinates. The operators in this module uses code from the YAC software package to compute the largest area fraction. For a detailed description of the interpolation method see [\[YAC\]](#). The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for this remapping method requires that no grid cell occurs more than once.

### Operators

<b>remaplaf</b>	Largest area fraction remapping Performs a largest area fraction remapping on all input fields.
<b>genlaf</b>	Generate largest area fraction remap weights Generates largest area fraction remapping weights for the first input field and writes the result to a file. The format of this file is netCDF following the SCRIP convention. Use the operator <a href="#">remap</a> to apply this remapping weights to a data file with the same source grid.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Environment

REMAP_AREA_MIN	This variable is used to set the minimum destination area fraction. The default of this variable is 0.0.
----------------	--

## 2.12.9. REMAP - Grid remapping

### Synopsis

```
remap,grid,weights ifile ofile
```

### Description

Interpolation between different horizontal grids can be a very time-consuming process. Especially if the data are on an unstructured and/or a large grid. In this case the interpolation process can be split into two parts. Firstly the generation of the interpolation weights, which is the most time-consuming part. These interpolation weights can be reused for every remapping process with the operator [remap](#). This operator remaps all input fields to a new horizontal grid. The remap type and the interpolation weights of one input grid are read from a netCDF file. More weights are computed if the input fields are on different grids. The netCDF file with the weights should follow the SCRIP convention. Normally these weights come from a previous call to one of the genXXX operators (e.g. [genbil](#)) or were created by the original SCRIP package.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
<i>weights</i>	STRING	Interpolation weights (SCRIP netCDF file)

### Environment

CDO_REMAP_NORM	This variable is used to choose the normalization of the conservative interpolation. By default CDO_REMAP_NORM is set to 'fracarea'. 'fracarea' uses the sum of the non-masked source cell intersected areas to normalize each target cell field value. This results in a reasonable flux value but the flux is not locally conserved. The option 'destarea' uses the total target cell area to normalize each target cell field value. Local flux conservation is ensured, but unreasonable flux values may result.
REMAP_EXTRAPOLATE	This variable is used to switch the extrapolation feature 'on' or 'off'. By default the extrapolation is enabled for remapdis, remapnn and for circular grids.
REMAP_AREA_MIN	This variable is used to set the minimum destination area fraction. The default of this variable is 0.0.
CDO_GRIDSEARCH_RADIUS	Grid search radius in degree, default 180 degree.

### Example

Say ifile contains fields on a quadrilateral curvilinear grid. To remap all fields bilinear to a Gaussian N32 grid use:

```
cdo genbil,n32 ifile remapweights.nc
cdo remap,n32,remapweights.nc ifile ofile
```

The result will be the same as:

```
cdo remapbil,n32 ifile ofile
```

## 2.12.10. REMAPETA - Remap vertical hybrid level

### Synopsis

```
remapeta,vct[,oro] ifile ofile
```

### Description

This operator interpolates between different vertical hybrid levels. This include the preparation of consistent data for the free atmosphere. The procedure for the vertical interpolation is based on the HIRLAM scheme and was adapted from [\[INTERA\]](#). The vertical interpolation is based on the vertical integration of the hydrostatic equation with few adjustments. The basic tasks are the following one:

- at first integration of hydrostatic equation
- extrapolation of surface pressure
- Planetary Boundary-Layer (PBL) profile interpolation
- interpolation in free atmosphere
- merging of both profiles
- final surface pressure correction

The vertical interpolation corrects the surface pressure. This is simply a cut-off or an addition of air mass. This mass correction should not influence the geostrophic velocity field in the middle troposphere. Therefore the total mass above a given reference level is conserved. As reference level the geopotential height of the 400 hPa level is used. Near the surface the correction can affect the vertical structure of the PBL. Therefore the interpolation is done using the potential temperature. But in the free atmosphere above a certain  $n$  ( $n=0.8$  defining the top of the PBL) the interpolation is done linearly. After the interpolation both profiles are merged. With the resulting temperature/pressure correction the hydrostatic equation is integrated again and adjusted to the reference level finding the final surface pressure correction. A more detailed description of the interpolation can be found in [\[INTERA\]](#). This operator requires all variables on the same horizontal grid.

### Parameter

<code>vct</code>	STRING	File name of an ASCII dataset with the vertical coordinate table
<code>oro</code>	STRING	File name with the orography (surf. geopotential) of the target dataset (optional)

### Environment

REMAPETA_PTOP	Sets the minimum pressure level for condensation. Above this level the humidity is set to the constant 1.E-6. The default value is 0 Pa.
---------------	--

### Note

The code numbers or the variable names of the required parameter have to follow the [\[ECHAM\]](#) convention. Presently, the vertical coordinate definition of a netCDF file has also to follow the ECHAM convention. This means:

- the dimension of the full level coordinate and the corresponding variable is called `mlev`,
- the dimension of the half level coordinate and the corresponding variable is called `ilev` (`ilev` must have one element more than `mlev`)
- the hybrid vertical coefficient `a` is given in units of Pa and called `hyai` (`hyam` for level midpoints)
- the hybrid vertical coefficient `b` is given in units of 1 and called `hybi` (`hybm` for level midpoints)

- the `mlev` variable has a `borders` attribute containing the character string 'ilev'

Use the `sinfo` command to test if your vertical coordinate system is recognized as hybrid system.

In case `remapeta` complains about not finding any data on hybrid model levels you may wish to use the `setzaxis` command to generate a `zaxis` description which conforms to the ECHAM convention. See section "1.4 Z-axis description" for an example how to define a hybrid Z-axis.

## Example

To remap between different hybrid model level data use:

```
cdo remapeta,vct ifile ofile
```

Here is an example `vct` file with 19 hybrid model level:

0	0.000000000000000000	0.000000000000000000
1	2000.0000000000000000	0.000000000000000000
2	4000.0000000000000000	0.000000000000000000
3	6046.1093750000000000	0.00033899326808751
4	8267.9296875000000000	0.00335718691349030
5	10609.5117187500000000	0.01307003945112228
6	12851.1015625000000000	0.03407714888453484
7	14698.5000000000000000	0.07064980268478394
8	15861.1289062500000000	0.12591671943664551
9	16116.2382812500000000	0.20119541883468628
10	15356.9218750000000000	0.29551959037780762
11	13621.4609375000000000	0.40540921688079834
12	11101.5585937500000000	0.52493220567703247
13	8127.1445312500000000	0.64610791206359863
14	5125.1406250000000000	0.75969839096069336
15	2549.9689941406250000	0.85643762350082397
16	783.1950683593750000	0.92874687910079956
17	0.000000000000000000	0.97298520803451538
18	0.000000000000000000	0.99228149652481079
19	0.000000000000000000	1.000000000000000000

### 2.12.11. VERTINTML - Vertical interpolation

#### Synopsis

```
ml2pl,plevels ifile ofile
```

```
ml2hl,hlevels ifile ofile
```

#### Description

Interpolate 3D variables on hybrid sigma pressure level to pressure or height levels. The input file should contain the log. surface pressure or the surface pressure. To interpolate the temperature, the surface geopotential is also needed. The pressure, temperature, and surface geopotential are identified by their GRIB1 code number or netCDF CF standard name. Supported parameter tables are: WMO standard table number 2 and ECMWF local table number 128. Use the alias **ml2plx/ml2hlx** or the environment variable EXTRAPOLATE to extrapolate missing values. This operator requires all variables on the same horizontal grid.

#### Operators

- |              |  |
|--------------|--|
| <b>ml2pl</b> | Model to pressure level interpolation<br>Interpolates 3D variables on hybrid sigma pressure level to pressure level.   |
| <b>ml2hl</b> | Model to height level interpolation<br>Interpolates 3D variables on hybrid sigma pressure level to height level. The procedure is the same as for the operator mh2pl except for the pressure levels being calculated from the heights by: $p_{level} = 101325 * \exp(h_{level} / -7000)$ |

#### Parameter

<i>plevels</i>	FLOAT	Pressure levels in pascal
<i>hlevels</i>	FLOAT	Height levels in meter (max level: 65535 m)

#### Environment

EXTRAPOLATE      If set to 1 extrapolate missing values.

#### Example

To interpolate hybrid model level data to pressure levels of 925, 850, 500 and 200 hPa use:

```
cdo ml2pl,92500,85000,50000,20000 ifile ofile
```

### 2.12.12. VERTINTAP - Vertical interpolation

#### Synopsis

```
ap2pl,plevels ifile ofile
```

#### Description

Interpolate 3D variables on hybrid sigma height coordinates to pressure levels. The input file must contain the 3D air pressure. The air pressure is identified by the netCDF CF standard name air\_pressure. This operator requires all variables on the same horizontal grid.

**Parameter**

*plevels*      FLOAT      Pressure levels in pascal

**Note**

This is a specific implementation for netCDF files from the ICON model, it may not work with data from other sources.

**Example**

To interpolate 3D variables on hybrid sigma height level to pressure levels of 925, 850, 500 and 200 hPa use:

```
cdo ap2pl,92500,85000,50000,20000 ifile ofile
```

**2.12.13. INTLEVEL - Linear level interpolation****Synopsis**

`intlevel,levels ifile ofile`

**Description**

This operator performs a linear vertical interpolation of non hybrid 3D variables.

**Parameter**

*levels*      FLOAT      Target levels

**Example**

To interpolate 3D variables on height levels to a new set of height levels use:

```
cdo intlevel,10,50,100,500,1000 ifile ofile
```

## 2.12.14. INTLEVEL3D - Linear level interpolation from/to 3d vertical coordinates

### Synopsis

```
<operator>,icoordinate ifile1 ifile2 ofile
```

### Description

This operator performs a linear vertical interpolation of 3D variables fields with given 3D vertical coordinates.

### Operators

**intlevel3d**      Linear level interpolation onto a 3d vertical coordinate

**intlevelx3d**    like intlevel3d but with extrapolation

### Parameter

<i>icoordinate</i>	STRING	filename for vertical source coordinates variable
<i>ifile2</i>	STRING	target vertical coordinate field (intlevel3d only)

### Example

To interpolate 3D variables from one set of 3d height levels into another one where

- *icoordinate* contains a single 3d variable, which represents the input 3d vertical coordinate
- *ifile1* contains the source data, which the vertical coordinate from *icoordinate* belongs to
- *ifile2* only contains the target 3d height levels

```
cdo intlevel3d,icoordinate ifile1 ifile2 ofile
```

## 2.12.15. INTTIME - Time interpolation

### Synopsis

```
inttime,date,time[,inc] ifile ofile
intntime,n ifile ofile
```

### Description

This module performs linear interpolation between timesteps.

### Operators

<b>inttime</b>	Interpolation between timesteps This operator creates a new dataset by linear interpolation between timesteps. The user has to define the start date/time with an optional increment.
<b>intntime</b>	Interpolation between timesteps This operator performs linear interpolation between timesteps. The user has to define the number of timesteps from one timestep to the next.

### Parameter

<i>date</i>	STRING	Start date (format YYYY-MM-DD)
<i>time</i>	STRING	Start time (format hh:mm:ss)
<i>inc</i>	STRING	Optional increment (seconds, minutes, hours, days, months, years) [default: 0hour]
<i>n</i>	INTEGER	Number of timesteps from one timestep to the next

### Example

Assumed a 6 hourly dataset starts at 1987-01-01 12:00:00. To interpolate this time series to a one hourly dataset use:

```
cdo inttime,1987-01-01,12:00:00,1hour ifile ofile
```



## 2.12.16. INTYEAR - Year interpolation

### Synopsis

```
intyear,years ifile1 ifile2 obase
```

### Description

This operator performs linear interpolation between two years, timestep by timestep. The input files need to have the same structure with the same variables. The output files will be named `<obase><yyyy><suffix>` where `yyyy` will be the year and `suffix` is the filename extension derived from the file format.

### Parameter

<code>years</code>	INTEGER	Comma separated list of years
--------------------	---------	-------------------------------

### Environment

<code>CDO_FILE_SUFFIX</code>	Set the default file suffix. This suffix will be added to the output file names instead of the filename extension derived from the file format. Set this variable to NULL to disable the adding of a file suffix.
------------------------------	---

### Example

Assume there are two monthly mean datasets over a year. The first dataset has 12 timesteps for the year 1985 and the second one for the year 1990. To interpolate the years between 1985 and 1990 month by month use:

```
cdo intyear,1986,1987,1988,1989 ifile1 ifile2 year
```

Example result of `'dir year*'` for netCDF datasets:

```
year1986.nc year1987.nc year1988.nc year1989.nc
```

## 2.13. Transformation

This section contains modules to perform spectral transformations.

Here is a short overview of all operators in this section:

<b>sp2gp</b>	Spectral to gridpoint
<b>sp2gpl</b>	Spectral to gridpoint (linear)
<b>gp2sp</b>	Gridpoint to spectral
<b>gp2spl</b>	Gridpoint to spectral (linear)
<b>sp2sp</b>	Spectral to spectral
<b>dv2uv</b>	Divergence and vorticity to U and V wind
<b>dv2uvl</b>	Divergence and vorticity to U and V wind (linear)
<b>uv2dv</b>	U and V wind to divergence and vorticity
<b>uv2dvl</b>	U and V wind to divergence and vorticity (linear)
<b>dv2ps</b>	D and V to velocity potential and stream function

### 2.13.1. SPECTRAL - Spectral transformation

#### Synopsis

```
<operator> ifile ofile
```

```
sp2sp, trunc ifile ofile
```

#### Description

This module transforms fields on Gaussian grids to spectral coefficients and vice versa.

#### Operators

- sp2gp**      Spectral to gridpoint  
Convert all fields with spectral coefficients to a regular Gaussian grid. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:  
$$\text{nlat} = NINT((\text{trunc} * \boxed{3} + 1.) / 2.)$$
- sp2gpl**      Spectral to gridpoint (linear)  
Convert all fields with spectral coefficients to a regular Gaussian grid. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:  
$$\text{nlat} = NINT((\text{trunc} * \boxed{2} + 1.) / 2.)$$
  
Use this operator to convert ERA40 data e.g. from TL159 to N80.
- gp2sp**      Gridpoint to spectral  
Convert all Gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:  
$$\text{trunc} = (\text{nlat} * 2 - 1) / \boxed{3}$$
- gp2spl**      Gridpoint to spectral (linear)  
Convert all Gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:  
$$\text{trunc} = (\text{nlat} * 2 - 1) / \boxed{2}$$
  
Use this operator to convert ERA40 data e.g. from N80 to TL159 instead of T106.
- sp2sp**      Spectral to spectral  
Change the triangular truncation of all spectral fields. The operator performs downward conversion by cutting the resolution. Upward conversions are achieved by filling in zeros.

#### Parameter

*trunc*      INTEGER      New spectral resolution

#### Example

To transform spectral coefficients from T106 to N80 Gaussian grid use:

```
cdo sp2gp ifile ofile
```

To transform spectral coefficients from TL159 to N80 Gaussian grid use:

```
cdo sp2gpl ifile ofile
```

## 2.13.2. WIND - Wind transformation

### Synopsis

```
<operator> ifile ofile
```

### Description

This module converts relative divergence and vorticity to U and V wind and vice versa. Divergence and vorticity are spherical harmonic coefficients in spectral space and U and V are on a regular Gaussian grid. The Gaussian latitudes needs to be ordered from north to south.

### Operators

- dv2uv**      Divergence and vorticity to U and V wind  
Calculate U and V wind on a Gaussian grid from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need to have the names sd and svo or code numbers 155 and 138. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:  
$$nlat = NINT((trunc * \sqrt{3} + 1.)/2.)$$
- dv2uvl**    Divergence and vorticity to U and V wind (linear)  
Calculate U and V wind on a Gaussian grid from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need to have the names sd and svo or code numbers 155 and 138. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:  
$$nlat = NINT((trunc * \sqrt{2} + 1.)/2.)$$
- uv2dv**      U and V wind to divergence and vorticity  
Calculate spherical harmonic coefficients of relative divergence and vorticity from U and V wind. The U and V wind need to be on a Gaussian grid and need to have the names u and v or the code numbers 131 and 132. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:  
$$trunc = (nlat * 2 - 1) / \sqrt{3}$$
- uv2dvl**    U and V wind to divergence and vorticity (linear)  
Calculate spherical harmonic coefficients of relative divergence and vorticity from U and V wind. The U and V wind need to be on a Gaussian grid and need to have the names u and v or the code numbers 131 and 132. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:  
$$trunc = (nlat * 2 - 1) / \sqrt{2}$$
- dv2ps**      D and V to velocity potential and stream function  
Calculate spherical harmonic coefficients of velocity potential and stream function from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need to have the names sd and svo or code numbers 155 and 138.

### Example

Assume a dataset has at least spherical harmonic coefficients of divergence and vorticity. To transform the spectral divergence and vorticity to U and V wind on a Gaussian grid use:

```
cdo dv2uv ifile ofile
```

## 2.14. Import/Export

This section contains modules to import and export data files which can not read or write directly with CDO.

Here is a short overview of all operators in this section:

<b>import_binary</b>	Import binary data sets
<b>import_cmsaf</b>	Import CM-SAF HDF5 files
<b>import_amsr</b>	Import AMSR binary files
<b>input</b>	ASCII input
<b>inputsrv</b>	SERVICE ASCII input
<b>inputtext</b>	EXTRA ASCII input
<b>output</b>	ASCII output
<b>outputf</b>	Formatted output
<b>outputint</b>	Integer output
<b>outputsrv</b>	SERVICE ASCII output
<b>outputtext</b>	EXTRA ASCII output
<b>outputtab</b>	Table output

### 2.14.1. IMPORTBINARY - Import binary data sets

#### Synopsis

```
import_binary ifile ofile
```

#### Description

This operator imports gridded binary data sets via a GrADS data descriptor file. The GrADS data descriptor file contains a complete description of the binary data as well as instructions on where to find the data and how to read it. The descriptor file is an ASCII file that can be created easily with a text editor. The general contents of a gridded data descriptor file are as follows:

- Filename for the binary data
- Missing or undefined data value
- Mapping between grid coordinates and world coordinates
- Description of variables in the binary data set

A detailed description of the components of a GrADS data descriptor file can be found in [\[GrADS\]](#). Here is a list of the supported components: BYTESWAPPED, CHSUB, DSET, ENDVARS, FILE-HEADER, HEADERBYTES, OPTIONS, TDEF, TITLE, TRAILERBYTES, UNDEF, VARS, XDEF, XYHEADER, YDEF, ZDEF

#### Note

Only 32-bit IEEE floats are supported for standard binary files!

#### Example

To convert a binary data file to netCDF use:

```
cdo -f nc import_binary ifile.ctl ofile.nc
```

Here is an example of a GrADS data descriptor file:

```
DSET ^ifile.bin
OPTIONS sequential
UNDEF -9e+33
XDEF 360 LINEAR -179.5 1
YDEF 180 LINEAR -89.5 1
ZDEF 1 LINEAR 1 1
TDEF 1 LINEAR 00:00Z15jun1989 12hr
VARS 1
  param 1 99 description of the variable
ENDVARS
```

The binary data file ifile.bin contains one parameter on a global 1 degree lon/lat grid written with FORTRAN record length headers (sequential).

## 2.14.2. IMPORTCMSAF - Import CM-SAF HDF5 files

### Synopsis

```
import_cmsaf ifile ofile
```

### Description

This operator imports gridded CM-SAF (Satellite Application Facility on Climate Monitoring) HDF5 files. CM-SAF exploits data from polar-orbiting and geostationary satellites in order to provide climate monitoring products of the following parameters:

**Cloud parameters:** cloud fraction (CFC), cloud type (CTY), cloud phase (CPH), cloud top height, pressure and temperature (CTH,CTP,CTT), cloud optical thickness (COT), cloud water path (CWP).

**Surface radiation components:** Surface albedo (SAL); surface incoming (SIS) and net (SNS) shortwave radiation; surface downward (SDL) and outgoing (SOL) longwave radiation, surface net longwave radiation (SNL) and surface radiation budget (SRB).

**Top-of-atmosphere radiation components:** Incoming (TIS) and reflected (TRS) solar radiative flux at top-of-atmosphere. Emitted thermal radiative flux at top-of-atmosphere (TET).

**Water vapour:** Vertically integrated water vapour (HTW), layered vertically integrated water vapour and layer mean temperature and relative humidity for 5 layers (HLW), temperature and mixing ratio at 6 pressure levels.

Daily and monthly mean products can be ordered via the CM-SAF web page ([www.cmsaf.eu](http://www.cmsaf.eu)). Products with higher spatial and temporal resolution, i.e. instantaneous swath-based products, are available on request ([contact.cmsaf@dwd.de](mailto:contact.cmsaf@dwd.de)). All products are distributed free-of-charge. More information on the data is available on the CM-SAF homepage ([www.cmsaf.eu](http://www.cmsaf.eu)).

Daily and monthly mean products are provided in equal-area projections. CDO reads the projection parameters from the metadata in the HDF5-headers in order to allow spatial operations like remapping. For spatial operations with instantaneous products on original satellite projection, additional files with arrays of latitudes and longitudes are needed. These can be obtained from CM-SAF together with the data.

### Note

To use this operator, it is necessary to build CDO with HDF5 support (version 1.6 or higher). The PROJ.4 library (version 4.6 or higher) is needed for full support of the remapping functionality.

### Example

A typical sequence of commands with this operator could look like this:

```
cdo -f nc remapbil,r360x180 -import_cmsaf cmsaf_product.hdf output.nc
```

(bilinear remapping to a predefined global grid with 1 deg resolution and conversion to netcdf).

If you work with CM-SAF data on original satellite project, an additional file with information on geolocation is required, to perform such spatial operations:

```
cdo -f nc remapbil,r720x360 -setgrid,cmsaf_latlon.h5 -import_cmsaf cmsaf.hdf out.nc
```

Some CM-SAF data are stored as scaled integer values. For some operations, it could be desirable (or necessary) to increase the accuracy of the converted products:

```
cdo -b f32 -f nc fldmean -sellonlatbox,0,10,0,10 -remapbil,r720x360 \  
-import_cmsaf cmsaf_product.hdf output.nc
```

### 2.14.3. IMPORTAMSR - Import AMSR binary files

#### Synopsis

```
import_amsr ifile ofile
```

#### Description

This operator imports gridded binary AMSR (Advanced Microwave Scanning Radiometer) data. The binary data files are available from the AMSR ftp site (<ftp://ftp.ssmi.com/amsre>). Each file consists of twelve (daily) or five (averaged) 0.25 x 0.25 degree grid (1440,720) byte maps. For daily files, six daytime maps in the following order, Time (UTC), Sea Surface Temperature (SST), 10 meter Surface Wind Speed (WSPD), Atmospheric Water Vapor (VAPOR), Cloud Liquid Water (CLOUD), and Rain Rate (RAIN), are followed by six nighttime maps in the same order. Time-Averaged files contain just the geophysical layers in the same order [SST, WSPD, VAPOR, CLOUD, RAIN]. More information to the data is available on the AMSR homepage <http://www.remss.com/amsr>.

#### Example

To convert monthly binary AMSR files to netCDF use:

```
cdo -f nc amsre_yyyymm5 amsre_yyyymm5.nc
```



## 2.14.4. INPUT - Formatted input

### Synopsis

**input**,*grid* ofile

**inputsrv** ofile

**inputtext** ofile

### Description

This module reads time series of one 2D variable from standard input. All input fields need to have the same horizontal grid. The format of the input depends on the chosen operator.

### Operators

<b>input</b>	ASCII input Reads fields with ASCII numbers from standard input and stores them in ofile. The numbers read are exactly that ones which are written out by the <a href="#">output</a> operator.
<b>inputsrv</b>	SERVICE ASCII input Reads fields with ASCII numbers from standard input and stores them in ofile. Each field should have a header of 8 integers (SERVICE likely). The numbers that are read are exactly that ones which are written out by the <a href="#">outputsrv</a> operator.
<b>inputtext</b>	EXTRA ASCII input Read fields with ASCII numbers from standard input and stores them in ofile. Each field should have header of 4 integers (EXTRA likely). The numbers read are exactly that ones which are written out by the <a href="#">outputtext</a> operator.

### Parameter

*grid*     STRING     Grid description file or name

### Example

Assume an ASCII dataset contains a field on a global regular grid with 32 longitudes and 16 latitudes (512 elements). To create a GRIB1 dataset from the ASCII dataset use:

```
cdo -f grb input,r32x16 ofile.grb < my_ascii_data
```

## 2.14.5. OUTPUT - Formatted output

### Synopsis

```
output ifiles
outputf,format[,nelem] ifiles
outputint ifiles
outputsrv ifiles
outputtext ifiles
```

### Description

This module prints all values of all input datasets to standard output. All input fields need to have the same horizontal grid. All input files need to have the same structure with the same variables. The format of the output depends on the chosen operator.

### Operators

<b>output</b>	ASCII output Prints all values to standard output. Each row has 6 elements with the C-style format "%13.6g".
<b>outputf</b>	Formatted output Prints all values to standard output. The format and number of elements for each row have to be specified by the parameters <i>format</i> and <i>nelem</i> . The default for <i>nelem</i> is 1.
<b>outputint</b>	Integer output Prints all values rounded to the nearest integer to standard output.
<b>outputsrv</b>	SERVICE ASCII output Prints all values to standard output. Each field with a header of 8 integers (SERVICE likely).
<b>outputtext</b>	EXTRA ASCII output Prints all values to standard output. Each field with a header of 4 integers (EXTRA likely).

### Parameter

<i>format</i>	STRING	C-style format for one element (e.g. %13.6g)
<i>nelem</i>	INTEGER	Number of elements for each row (default: nelem = 1)

### Example

To print all field elements of a dataset formatted with "%8.4g" and 8 values per line use:

```
cdo outputf,%8.4g,8 ifile
```

Example result of a dataset with one field on 64 grid points:

261.7	262	257.8	252.5	248.8	247.7	246.3	246.1
250.6	252.6	253.9	254.8	252	246.6	249.7	257.9
273.4	266.2	259.8	261.6	257.2	253.4	251	263.7
267.5	267.4	272.2	266.7	259.6	255.2	272.9	277.1
275.3	275.5	276.4	278.4	282	269.6	278.7	279.5
282.3	284.5	280.3	280.3	280	281.5	284.7	283.6
292.9	290.5	293.9	292.6	292.7	292.8	294.1	293.6
293.8	292.6	291.2	292.6	293.2	292.8	291	291.2

## 2.14.6. OUTPUTTAB - Table output

### Synopsis

```
outputtab,params ifiles ofile
```

### Description

This operator prints a table of all input datasets to standard output. *ifiles* is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. All input fields need to have the same horizontal grid.

The contents of the table depends on the chosen paramters. The format of each table parameter is *keyname[:len]*. *len* is the optional length of a table entry. Here is a list of all valid keynames:

Keyname	Type	Description
value	FLOAT	Value of the variable [len:8]
name	STRING	Name of the variable [len:8]
param	STRING	Parameter ID (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]]) [len:11]
code	INTEGER	Code number [len:4]
lon	FLOAT	Longitude coordinate [len:6]
lat	FLOAT	Latitude coordinate [len:6]
lev	FLOAT	Vertical level [len:6]
xind	INTEGER	Grid x index [len:4]
yind	INTEGER	Grid y index [len:4]
timestep	INTEGER	Timestep number [len:6]
date	STRING	Date (format YYYY-MM-DD) [len:10]
time	STRING	Time (format hh:mm:ss) [len:8]
year	INTEGER	Year [len:5]
month	INTEGER	Month [len:2]
day	INTEGER	Day [len:2]
nohead	INTEGER	Disable output of header line

### Parameter

*params*      STRING      Comma separated list of keynames, one for each column of the table

### Example

To print a table with name, date, lon, lat and value information use:

```
cdo outputtab,name,date,lon,lat,value ifile
```

Here is an example output of a time series with the yearly mean temperatur at lon=10/lat=53.5:

#	name	date	lon	lat	value
	tsurf	1991-12-31	10	53.5	8.83903
	tsurf	1992-12-31	10	53.5	8.17439
	tsurf	1993-12-31	10	53.5	7.90489
	tsurf	1994-12-31	10	53.5	10.0216
	tsurf	1995-12-31	10	53.5	9.07798

## 2.15. Miscellaneous

This section contains miscellaneous modules which do not fit to the other sections before.

Here is a short overview of all operators in this section:

<b>gradsdes</b>	GrADS data descriptor file
<b>after</b>	ECHAM standard post processor
<b>bandpass</b>	Bandpass filtering
<b>lowpass</b>	Lowpass filtering
<b>highpass</b>	Highpass filtering
<b>gridarea</b>	Grid cell area
<b>gridweights</b>	Grid cell weights
<b>smooth9</b>	9 point smoothing
<b>setvals</b>	Set list of old values to new values
<b>setrtoc</b>	Set range to constant
<b>setrtoc2</b>	Set range to constant others to constant2
<b>timsort</b>	Sort over the time
<b>const</b>	Create a constant field
<b>random</b>	Create a field with random numbers
<b>topo</b>	Create a field with topography
<b>for</b>	Create a time series
<b>stdatm</b>	Create values for pressure and temperature for hydrostatic atmosphere
<b>rotuvb</b>	Backward rotation
<b>mastrfu</b>	Mass stream function
<b>sealevelpressure</b>	Sea level pressure
<b>adisit</b>	Potential temperature to in-situ temperature
<b>adipot</b>	In-situ temperature to potential temperature
<b>rhopot</b>	Calculates potential density
<b>histcount</b>	Histogram count
<b>histsum</b>	Histogram sum
<b>histmean</b>	Histogram mean
<b>histfreq</b>	Histogram frequency
<b>sethalo</b>	Set the left and right bounds of a field
<b>wct</b>	Windchill temperature
<b>fdns</b>	Frost days where no snow index per time period
<b>strwin</b>	Strong wind days index per time period
<b>strbre</b>	Strong breeze days index per time period
<b>strgal</b>	Strong gale days index per time period
<b>hurr</b>	Hurricane days index per time period

### 2.15.1. GRADSDES - GrADS data descriptor file

#### Synopsis

```
gradsdes[,mapversion] ifile
```

#### Description

Creates a GrADS data descriptor file. Supported file formats are GRIB1, netCDF, SERVICE, EXTRA and IEG. For GRIB1 files the GrADS map file is also generated. For SERVICE and EXTRA files the grid have to be specified with the CDO option '-g <grid>'. This module takes ifile in order to create filenames for the descriptor (ifile.ct1) and the map (ifile.gmp) file.

#### Parameter

**mapversion**      **INTEGER**      Format version of the GrADS map file for GRIB1 datasets. Use 1 for a machine specific version 1 GrADS map file, 2 for a machine independent version 2 GrADS map file and 4 to support GRIB files >2GB. A version 2 map file can be used only with GrADS version 1.8 or newer. A version 4 map file can be used only with GrADS version 2.0 or newer. The default is 4 for files >2GB, otherwise 2.

#### Example

To create a GrADS data descriptor file from a GRIB1 dataset use:

```
cdo gradsdes ifile.grb
```

This will create a descriptor file with the name ifile.ct1 and the map file ifile.gmp.

Assumed the input GRIB1 dataset has 3 variables over 12 timesteps on a Gaussian N16 grid. The contents of the resulting GrADS data description file is approximately:

```
DSET ^ifile.grb
DTYPE GRIB
INDEX ^ifile.gmp
XDEF 64 LINEAR 0.000000 5.625000
YDEF 32 LEVELS -85.761 -80.269 -74.745 -69.213 -63.679 -58.143
               -52.607 -47.070 -41.532 -35.995 -30.458 -24.920
               -19.382 -13.844 -8.307 -2.769 2.769 8.307
               13.844 19.382 24.920 30.458 35.995 41.532
               47.070 52.607 58.143 63.679 69.213 74.745
               80.269 85.761
ZDEF 4 LEVELS 925 850 500 200
TDEF 12 LINEAR 12:00 Z1jan1987 1mo
TITLE ifile.grb T21 grid
OPTIONS yrev
UNDEF -9e+33
VARS 3
geosp 0 129,1,0 surface geopotential (orography) [m^2/s^2]
t      4 130,99,0 temperature [K]
tslm1 0 139,1,0 surface temperature of land [K]
ENDVARS
```

## 2.15.2. AFTERBURNER - ECHAM standard post processor

### Synopsis

```
after ifiles ofile
```

### Description

The "afterburner" is the standard post processor for [ECHAM](#) data which provides the following operations:

- Extract specified variables and levels
- Compute derived variables
- Transform spectral data to Gaussian grid representation
- Vertical interpolation to pressure levels
- Compute temporal means

This operator reads selection parameters as namelist from stdin. Use the UNIX redirection "<namelistfile" to read the namelist from file.

### Namelist

Namelist parameter and there defaults:

```
TYPE=0, CODE=-1, LEVEL=-1, INTERVAL=0, MEAN=0, EXTRAPOLATE=0
```

**TYPE** controls the transformation and vertical interpolation. Transforming spectral data to Gaussian grid representation and vertical interpolation to pressure levels are performed in a chain of steps. The **TYPE** parameter may be used to stop the chain at a certain step. Valid values are:

```
TYPE = 0 : Hybrid    level spectral coefficients
TYPE = 10 : Hybrid   level fourier  coefficients
TYPE = 11 : Hybrid   level zonal mean sections
TYPE = 20 : Hybrid   level gauss grids
TYPE = 30 : Pressure level gauss grids
TYPE = 40 : Pressure level fourier  coefficients
TYPE = 41 : Pressure level zonal mean sections
TYPE = 50 : Pressure level spectral coefficients
TYPE = 60 : Pressure level fourier  coefficients
TYPE = 61 : Pressure level zonal mean sections
TYPE = 70 : Pressure level gauss grids
```

Vorticity, divergence, streamfunction and velocity potential need special treatment in the vertical transformation. They are not available as types 30, 40 and 41. If you select one of these combinations, type is automatically switched to the equivalent types 70, 60 and 61. The type of all other variables will be switched too, because the type is a global parameter.

**CODE** selects the variables by the ECHAM GRIB1 code number (1-255). The default value **-1** processes all detected codes. Derived variables computed by the afterburner:

Code	Name	Longname	Level	Needed Codes/Computation
34	low_cld	low cloud	single	223 on modellevel
35	mid_cld	mid cloud	single	223 on modellevel
36	hih_cld	high cloud	single	223 on modellevel
131	u	u-velocity	atm (ml+pl)	138, 155
132	v	v-velocity	atm (ml+pl)	138, 155
135	omega	vertical velocity	atm (ml+pl)	138, 152, 155
148	stream	streamfunction	atm (ml+pl)	131, 132
149	velopot	velocity potential	atm (ml+pl)	131, 132
151	slp	mean sea level pressure	surface	129, 130, 152
156	geopoth	geopotential height	atm (ml+pl)	129, 130, 133, 152
157	rhumidity	relative humidity	atm (ml+pl)	130, 133, 152
189	scfs	surface solar cloud forcing	surface	176-185
190	tcfs	surface thermal cloud forcing	surface	177-186
191	scf0	top solar cloud forcing	surface	178-187
192	tcf0	top thermal cloud forcing	surface	179-188
259	windspeed	windspeed	atm (ml+pl)	$\sqrt{u^2+v^2}$
260	precip	total precipitation	surface	142+143

**LEVEL** selects the hybrid or pressure levels. The allowed values depends on the parameter **TYPE**. The default value **-1** processes all detected levels.

**INTERVAL** selects the processing interval. The default value **0** process data on monthly intervals. **INTERVAL=1** sets the interval to daily.

**MEAN=1** compute and write monthly or daily mean fields. The default value **0** writes out all timesteps.

**EXTRAPOLATE=0** switch of the extrapolation of missing values during the interpolation from model to pressure level (only available with **MEAN=0** and **TYPE=30**). The default value **1** extrapolate missing values.

Possible combinations of **TYPE**, **CODE** and **MEAN**:

TYPE	CODE	MEAN
0/10/11	130 temperature	0
0/10/11	131 u-velocity	0
0/10/11	132 v-velocity	0
0/10/11	133 specific humidity	0
0/10/11	138 vorticity	0
0/10/11	148 streamfunction	0
0/10/11	149 velocity potential	0
0/10/11	152 LnPs	0
0/10/11	155 divergence	0
>11	all codes	0/1

## Example

To interpolate ECHAM hybrid model level data to pressure levels of 925, 850, 500 and 200 hPa, use:

```
cdo after ifile ofile << EON
    TYPE=30 LEVEL=92500,85000,50000,20000
EON
```

### 2.15.3. FILTER - Time series filtering

#### Synopsis

**bandpass**,*fmin*,*fmax* ifile ofile

**lowpass**,*fmax* ifile ofile

**highpass**,*fmin* ifile ofile

#### Description

This module takes the time series for each gridpoint in *ifile* and (fast fourier) transforms it into the frequency domain. According to the particular operator and its parameters certain frequencies are filtered (set to zero) in the frequency domain and the spectrum is (inverse fast fourier) transformed back into the time domain. To determine the frequency the time-axis of *ifile* is used. (Data should have a constant time increment since this assumption applies for transformation. However, the time increment has to be different from zero.) All frequencies given as parameter are interpreted per year. This is done by the assumption of a 365-day calendar. Consequently if you want to perform multiyear-filtering accurately you have to delete the 29th of February. If your *ifile* has a 360 year calendar the frequency parameters *fmin* respectively *fmax* should be multiplied with a factor of 360/365 in order to obtain accurate results. For the set up of a frequency filter the frequency parameters have to be adjusted to a frequency in the data. Here *fmin* is rounded down and *fmax* is always rounded up. Consequently it is possible to use bandpass with *fmin=fmax* without getting a zero-field for *ofile*. Hints for efficient usage:

- to get reliable results the time-series has to be detrended (cdo detrend)
- the lowest frequency greater zero that can be contained in *ifile* is  $1/(N*dT)$ ,
- the greatest frequency is  $1/(2dT)$  (Nyquist frequency),

with N the number of timesteps and dT the time increment of *ifile* in years.

#### Operators

<b>bandpass</b>	Bandpass filtering Bandpass filtering (pass for frequencies between <i>fmin</i> and <i>fmax</i> ). Suppresses all variability outside the frequency range specified by [ <i>fmin</i> , <i>fmax</i> ].
<b>lowpass</b>	Lowpass filtering Lowpass filtering (pass for frequencies lower than <i>fmax</i> ). Suppresses all variability with frequencies greater than <i>fmax</i> .
<b>highpass</b>	Highpass filtering Highpass filtering (pass for frequencies greater than <i>fmin</i> ). Suppresses all variability with frequencies lower than <i>fmin</i> .

#### Parameter

<i>fmin</i>	FLOAT	Minimum frequency per year that passes the filter.
<i>fmax</i>	FLOAT	Maximum frequency per year that passes the filter.

#### Example

Now assume your data are still hourly for a time period of 5 years but with a 365/366-day- calendar and you want to suppress the variability on timescales greater or equal to one year (we suggest here to use a number x bigger than one (e.g. x=1.5) since there will be dominant frequencies around the peak (if there is one) as well due to the issue that the time series is not of infinite length). Therefore you can use the following:



```
cdo highpass,x -del29feb ifile ofile
```

Accordingly you might use the following to suppress variability on timescales shorter than one year:

```
cdo lowpass,1 -del29feb ifile ofile
```

Finally you might be interested in 2-year variability. If you want to suppress the seasonal cycle as well as say the longer cycles in climate system you might use

```
cdo bandpass,x,y -del29feb ifile ofile
```

with  $x \leq 0.5$  and  $y \geq 0.5$ .

## 2.15.4. GRIDCELL - Grid cell quantities

### Synopsis

```
<operator> ifile ofile
```

### Description

This module reads the grid cell area of the first grid from the input stream. If the grid cell area is missing it will be computed from the grid description. Depending on the chosen operator the grid cell area or weights are written to the output stream.

### Operators

<b>gridarea</b>	Grid cell area Writes the grid cell area to the output stream. If the grid cell area have to be computed it is scaled with the earth radius to square meters.
<b>gridweights</b>	Grid cell weights Writes the grid cell area weights to the output stream.

### Environment

PLANET_RADIUS	This variable is used to scale the computed grid cell areas to square meters. By default PLANET_RADIUS is set to an earth radius of 6371000 meter.
---------------	--

## 2.15.5. SMOOTH9 - 9 point smoothing

### Synopsis

```
smooth9 ifile ofile
```

### Description

Performs a 9 point smoothing on all fields with a quadrilateral curvilinear grid. The result at each grid point is a weighted average of the grid point plus the 8 surrounding points. The center point receives a weight of 1.0, the points at each side and above and below receive a weight of 0.5, and corner points receive a weight of 0.3. All 9 points are multiplied by their weights and summed, then divided by the total weight to obtain the smoothed value. Any missing data points are not included in the sum; points beyond the grid boundary are considered to be missing. Thus the final result may be the result of an averaging with less than 9 points.

## 2.15.6. REPLACEVALUES - Replace variable values

### Synopsis

```
setvals,oldval,newval[...] ifile ofile
```

```
setrtoc,rmin,rmax,c ifile ofile
```

```
setrtoc2,rmin,rmax,c,c2 ifile ofile
```

### Description

This module replaces old variable values with new values, depending on the operator.

### Operators

<b>setvals</b>	Set list of old values to new values Supply a list of n pairs of old and new values.
<b>setrtoc</b>	Set range to constant $o(t,x) = \begin{cases} c & \text{if } i(t,x) \geq rmin \wedge i(t,x) \leq rmax \\ i(t,x) & \text{if } i(t,x) < rmin \vee i(t,x) > rmax \end{cases}$
<b>setrtoc2</b>	Set range to constant others to constant2 $o(t,x) = \begin{cases} c & \text{if } i(t,x) \geq rmin \wedge i(t,x) \leq rmax \\ c2 & \text{if } i(t,x) < rmin \vee i(t,x) > rmax \end{cases}$

### Parameter

<i>oldval,newval,...</i>	FLOAT	Pairs of old and new values
<i>rmin</i>	FLOAT	Lower bound
<i>rmax</i>	FLOAT	Upper bound
<i>c</i>	FLOAT	New value - inside range
<i>c2</i>	FLOAT	New value - outside range

### 2.15.7. TIMSORT - Timsort

#### Synopsis

```
timsort ifile ofile
```

#### Description

Sorts the elements in ascending order over all timesteps for every field position. After sorting it is:

$$o(t_1, x) \leq o(t_2, x) \quad \forall (t_1 < t_2), x$$

#### Example

To sort all field elements of a dataset over all timesteps use:

```
cdo timsort ifile ofile
```

### 2.15.8. VARGEN - Generate a field

#### Synopsis

```
const,const,grid ofile
random,grid[,seed] ofile
topo[,grid] ofile
for,start,end[,inc] ofile
stdatm,levels ofile
```

#### Description

Generates a dataset with one or more fields

#### Operators

<b>const</b>	Create a constant field Creates a constant field. All field elements of the grid have the same value.
<b>random</b>	Create a field with random numbers Creates a field with rectangularly distributed random numbers in the interval [0,1].
<b>topo</b>	Create a field with topography Creates a field with topography data, per default on a global half degree grid.
<b>for</b>	Create a time series Creates a time series with field size 1 and field elements beginning with a start value in time step 1 which is increased from one time step to the next.
<b>stdatm</b>	Create values for pressure and temperature for hydrostatic atmosphere Creates pressure and temperature values for the given list of vertical levels. The formulas are: $P(z) = P_0 \exp \left( -\frac{g}{R} \frac{H}{T_0} \log \left( \frac{\exp(\frac{z}{H}) T_0 + \Delta T}{T_0 + \Delta T} \right) \right)$ $T(z) = T_0 + \Delta T \exp \left( -\frac{z}{H} \right)$

with the following constants

$$\begin{aligned}
 T_0 &= 213\text{K} && : \text{offset to get a surface temperature of } 288\text{K} \\
 \Delta T &= 75\text{K} && : \text{Temperature lapse rate for } 10\text{Km} \\
 P_0 &= 1013.25\text{hPa} && : \text{surface pressure} \\
 H &= 10000.0\text{m} && : \text{scale height} \\
 g &= 9.80665 \frac{\text{m}}{\text{s}^2} && : \text{earth gravity} \\
 R &= 287.05 \frac{\text{J}}{\text{kgK}} && : \text{gas constant for air}
 \end{aligned}$$

This is the solution for the hydrostatic equations and is only valid for the troposphere (constant positive lapse rate). The temperature increase in the stratosphere and other effects of the upper atmosphere are not taken into account.

## Parameter

<i>const</i>	FLOAT	Constant
<i>seed</i>	INTEGER	The seed for a new sequence of pseudo-random numbers [default: 1]
<i>grid</i>	STRING	Target grid description file or name
<i>start</i>	FLOAT	Start value of the loop
<i>end</i>	FLOAT	End value of the loop
<i>inc</i>	FLOAT	Increment of the loop [default: 1]
<i>levels</i>	FLOAT	Target levels in metre above surface

## Example

To create a standard atmosphere dataset on a given horizontal grid:

```
cdo enlarge,gridfile -stdatm,10000,8000,5000,3000,2000,1000,500,200,0 ofile
```

### 2.15.9. ROTUVB - Rotation

#### Synopsis

```
rotuvb,u,v,... ifile ofile
```

#### Description

This is a special operator for datasets with wind components on a rotated grid, e.g. data from the regional model REMO. It performs a backward transformation of velocity components U and V from a rotated spherical system to a geographical system.

#### Parameter

<code>u,v,...</code>	STRING	Pairs of zonal and meridional velocity components (use variable names or code numbers)
----------------------	--------	--

#### Example

To transform the u and v velocity of a dataset from a rotated spherical system to a geographical system use:

```
cdo rotuvb,u,v ifile ofile
```

### 2.15.10. MASTRFU - Mass stream function

#### Synopsis

```
mastrfu ifile ofile
```

#### Description

This is a special operator for the post processing of the atmospheric general circulation model ECHAM. It computes the mass stream function (code=272). The input dataset have to be a zonal mean of v-velocity [m/s] (code=132) on pressure levels.

#### Example

To compute the mass stream function from a zonal mean v-velocity dataset use:

```
cdo mastrfu ifile ofile
```

### 2.15.11. DERIVEPAR - Sea level pressure

#### Synopsis

```
sealevelpressure ifile ofile
```

#### Description

This operator computes the sea level pressure (air\_pressure\_at\_sea\_level). Required input fields are surface\_air\_pressure, surface\_geopotential and air\_temperature on hybrid sigma pressure levels.

## 2.15.12. ADISIT - Potential temperature to in-situ temperature and vice versa

### Synopsis

```
adisit[,pressure] ifile ofile
adipot ifile ofile
```

### Description

#### Operators

- adisit** Potential temperature to in-situ temperature  
 This is a special operator for the post processing of the ocean and sea ice model output. It converts potential temperature adiabatically to in-situ temperature to(t, s, p). Required input fields are sea water potential temperature (name=tho; code=2) and sea water salinity (name=sao; code=5). Pressure is calculated from the level information or can be specified by the optional parameter. Output fields are sea water temperature (name=to; code=20) and sea water salinity (name=s; code=5).
- adipot** In-situ temperature to potential temperature  
 This is a special operator for the post processing of the ocean and sea ice model output. It converts in-situ temperature to potential temperature tho(to, s, p). Required input fields are sea water in-situ temperature (name=t; code=2) and sea water salinity (name=sao; code=5). Pressure is calculated from the level information or can be specified by the optional parameter. Output fields are sea water temperature (name=tho; code=2) and sea water salinity (name=s; code=5).

#### Parameter

*pressure*      FLOAT      Pressure in bar (constant value assigned to all levels)

## 2.15.13. RHOPOT - Calculates potential density

### Synopsis

```
rhopot[,pressure] ifile ofile
```

### Description

This is a special operator for the post processing of the ocean and sea ice model MPIOM. It calculates the sea water potential density (name=rhopot; code=18). Required input fields are sea water in-situ temperature (name=to; code=20) and sea water salinity (name=sao; code=5). Pressure is calculated from the level information or can be specified by the optional parameter.

#### Parameter

*pressure*      FLOAT      Pressure in bar (constant value assigned to all levels)

### Example

To compute the sea water potential density from the potential temperature use this operator in combination with [adisit](#):

```
cdo rhopot -adisit ifile ofile
```

## 2.15.14. HISTOGRAM - Histogram

### Synopsis

```
<operator>,<bounds> ifile ofile
```

### Description

This module creates bins for a histogram of the input data. The bins have to be adjacent and have non-overlapping intervals. The user has to define the bounds of the bins. The first value is the lower bound and the second value the upper bound of the first bin. The bounds of the second bin are defined by the second and third value, aso. Only 2-dimensional input fields are allowed. The output file contains one vertical level for each of the bins requested.

### Operators

<b>histcount</b>	Histogram count Number of elements in the bin range.
<b>histsum</b>	Histogram sum Sum of elements in the bin range.
<b>histmean</b>	Histogram mean Mean of elements in the bin range.
<b>histfreq</b>	Histogram frequency Relative frequency of elements in the bin range.

### Parameter

*bounds*      FLOAT      Comma separated list of the bin bounds (-inf and inf valid)

## 2.15.15. SETHALO - Set the left and right bounds of a field

### Synopsis

```
sethalo,<lhalo>,<rhalo> ifile ofile
```

### Description

This operator sets the left and right bounds of the rectangularly understood fields. Positive numbers of the parameter *lhalo* enlarges the left bound by the given number of columns from the right bound. The parameter *rhalo* does the similar for the right bound. Negative numbers of the parameter *lhalo/rhalo* can be used to remove the given number of columns of the left and right bounds.

### Parameter

<i>lhalo</i>	INTEGER	Left halo
<i>rhalo</i>	INTEGER	Right halo



## 2.15.16. WCT - Windchill temperature

### Synopsis

```
wct ifile1 ifile2 ofile
```

### Description

Let ifile1 and ifile2 be time series of temperature and wind speed records, then a corresponding time series of resulting windchill temperatures is written to ofile. The wind chill temperature calculation is only valid for a temperature of  $T \leq 33$  °C and a wind speed of  $v \geq 1.39$  m/s. Whenever these conditions are not satisfied, a missing value is written to ofile. Note that temperature and wind speed records have to be given in units of °C and m/s, respectively.

## 2.15.17. FDNS - Frost days where no snow index per time period

### Synopsis

```
fdns ifile1 ifile2 ofile
```

### Description

Let ifile1 be a time series of the daily minimum temperature TN and ifile2 be a corresponding series of daily surface snow amounts. Then the number of days where  $TN < 0$  °C and the surface snow amount is less than 1 cm is counted. The temperature TN have to be given in units of Kelvin. The date information of a timestep in ofile is the date of the last contributing timestep in ifile.

## 2.15.18. STRWIN - Strong wind days index per time period

### Synopsis

```
strwin[,v] ifile ofile
```

### Description

Let ifile be a time series of the daily maximum horizontal wind speed VX, then the number of days where  $VX > v$  is counted. The horizontal wind speed  $v$  is an optional parameter with default  $v = 10.5$  m/s. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to  $v$ . Note that both VX and  $v$  have to be given in units of m/s. Also note that the horizontal wind speed is defined as the square root of the sum of squares of the zonal and meridional wind speeds. The date information of a timestep in ofile is the date of the last contributing timestep in ifile.

### Parameter

$v$	FLOAT	Horizontal wind speed threshold (m/s, default $v = 10.5$ m/s)
-----	-------	---

### 2.15.19. STRBRE - Strong breeze days index per time period

#### Synopsis

```
strbre ifile ofile
```

#### Description

Let `ifile` be a time series of the daily maximum horizontal wind speed  $VX$ , then the number of days where  $VX$  is greater than or equal to 10.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 10.5 m/s. Note that  $VX$  is defined as the square root of the sum of squares of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile`.

### 2.15.20. STRGAL - Strong gale days index per time period

#### Synopsis

```
strgal ifile ofile
```

#### Description

Let `ifile` be a time series of the daily maximum horizontal wind speed  $VX$ , then the number of days where  $VX$  is greater than or equal to 20.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 20.5 m/s. Note that  $VX$  is defined as the square root of the sum of square of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile`.

### 2.15.21. HURR - Hurricane days index per time period

#### Synopsis

```
hurr ifile ofile
```

#### Description

Let `ifile` be a time series of the daily maximum horizontal wind speed  $VX$ , then the number of days where  $VX$  is greater than or equal to 32.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 32.5 m/s. Note that  $VX$  is defined as the square root of the sum of squares of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile`.

# Bibliography

- [CDI]  
[Climate Data Interface](#), from the [Max Planck Institute for Meteorologie](#)
- [CM-SAF]  
[Satellite Application Facility on Climate Monitoring](#), from the [German Weather Service \(Deutscher Wetterdienst, DWD\)](#)
- [ECHAM]  
[The atmospheric general circulation model ECHAM5](#), from the [Max Planck Institute for Meteorologie](#)
- [GrADS]  
[Grid Analysis and Display System](#), from the [Center for Ocean-Land-Atmosphere Studies \(COLA\)](#)
- [GRIB]  
[GRIB version 1](#), from the [World Meteorological Organisation \(WMO\)](#)
- [GRIBAPI]  
[GRIB API decoding/encoding](#), from the [European Centre for Medium-Range Weather Forecasts \(ECMWF\)](#)
- [HDF5]  
[HDF version 5](#), from the [HDF Group](#)
- [INTERA]  
[INTERA Software Package](#), from the [Max Planck Institute for Meteorologie](#)
- [MPIOM]  
[Ocean and sea ice model](#), from the [Max Planck Institute for Meteorologie](#)
- [netCDF]  
[NetCDF Software Package](#), from the [UNIDATA Program Center of the University Corporation for Atmospheric Research](#)
- [PINGO]  
[The PINGO package](#), from the [Model & Data group](#) at the [Max Planck Institute for Meteorologie](#)
- [REMO]  
[Regional Model](#), from the [Max Planck Institute for Meteorologie](#)
- [Peisendorfer]  
Rudolph W. Peisendorfer: [Principal Component Analysis](#), Elsevier (1988)
- [PROJ.4]  
[Cartographic Projections Library](#), originally written by Gerald Evenden then of the USGS.
- [SCRIP]  
[SCRIP Software Package](#), from the [Los Alamos National Laboratory](#)
- [gzip]  
[Gzip compression software](#), developed at [University of New Mexico](#).
- [vonStorch]  
Hans von Storch, Walter Zwiers: [Statistical Analysis in Climate Research](#), Cambridge University Press (1999)
- [YAC]  
[YAC - Yet Another Coupler Software Package](#), from [DKRZ](#) and [MPI for Meteorologie](#)

## A. Environment Variables

The following table describes the environment variables that affect **CDO**.

Variable name	Default	Description
CDO_FILE_SUFFIX	None	Default file suffix. This suffix will be added to the output file name instead of the filename extension derived from the file format. NULL will disable the adding of a file suffix.
CDO_HISTORY_INFO	1	Append netCDF global attribute histroy
CDO_PCTL_NBINS	101	Number of histogram bins.
CDO_RESET_HISTORY	0	Set to 1 to reset the netCDF <i>history</i> global attribute.
CDO_REMAP_NORM	fracarea	Choose the normalization for the conservative interpolation
CDO_GRIDSEARCH_RADIUS	180	Grid search radius in degree. Used by the operators setmisstonn, remapdis and remapnn.
CDO_TIMESTAT_DATE	None	Set the date information of a time statistic operator to the "first", "middle" or "last" contributing timestep.
CDO_USE_FFTW	1	Set to 0 to switch off usage of FFTW. Used in the Filter module.
CDO_VERSION_INFO	1	Set to 0 to disable netCDF global attribute CDO

## B. Parallelized operators

Some of the **CDO** operators are parallelized with OpenMP. To use **CDO** with multiple OpenMP threads, you have to set the number of threads with the option '-P'. Here is an example to distribute the bilinear interpolation on 8 OpenMP threads:

```
cdo -P 8 remapbil,targetgrid ifile ofile
```

The following **CDO** operators are parallelized with OpenMP:

Module	Operator	Description
Detrend	detrend	Detrend
Ensstat	ensmin	Ensemble minimum
Ensstat	ensmax	Ensemble maximum
Ensstat	enssum	Ensemble sum
Ensstat	ensmean	Ensemble mean
Ensstat	ensavg	Ensemble average
Ensstat	ensvar	Ensemble variance
Ensstat	ensstd	Ensemble standard deviation
Ensstat	enspctl	Ensemble percentiles
Filter	bandpass	Bandpass filtering
Filter	lowpass	Lowpass filtering
Filter	highpass	Highpass filtering
Fourier	fourier	Fourier transformation
Genweights	genbil	Generate bilinear interpolation weights
Genweights	genbic	Generate bicubic interpolation weights
Genweights	gendis	Generate distance-weighted average remap weights
Genweights	gennn	Generate nearest neighbor remap weights
Genweights	gencon	Generate 1st order conservative remap weights
Genweights	gencon2	Generate 2nd order conservative remap weights
Genweights	genlaf	Generate largest area fraction remap weights
Gridboxstat	gridboxmin	Gridbox minimum
Gridboxstat	gridboxmax	Gridbox maximum
Gridboxstat	gridboxsum	Gridbox sum
Gridboxstat	gridboxmean	Gridbox mean
Gridboxstat	gridboxavg	Gridbox average
Gridboxstat	gridboxvar	Gridbox variance
Gridboxstat	gridboxstd	Gridbox standard deviation
Remapeta	remapeta	Remap vertical hybrid level
Remap	remapbil	Bilinear interpolation
Remap	remapbic	Bicubic interpolation
Remap	remapdis	Distance-weighted average remapping
Remap	remapnn	Nearest neighbor remapping
Remap	remapcon	First order conservative remapping
Remap	remapcon2	Second order conservative remapping
Remap	remaplaf	Largest area fraction remapping

## C. Standard name table

The following CF standard names are supported by **CDO**.

CF standard name	Units	GRIB 1 code	variable name
surface_geopotential	m2 s-2	129	geosp
air_temperature	K	130	ta
specific_humidity	1	133	hus
surface_air_pressure	Pa	134	aps
air_pressure_at_sea_level	Pa	151	psl
geopotential_height	m	156	zg

## D. Grid description examples

### D.1. Example of a curvilinear grid description

Here is an example for the **CDO** description of a curvilinear grid. `xvals/yvals` describe the positions of the 6x5 quadrilateral grid cells. The first 4 values of `xbounds/ybounds` are the corners of the first grid cell.

gridtype	=	curvilinear													
gridsize	=	30													
xsize	=	6													
ysize	=	5													
xvals	=	-21	-11	0	11	21	30	-25	-13	0	13				
		25	36	-31	-16	0	16	31	43	-38	-21				
		0	21	38	52	-51	-30	0	30	51	64				
xbounds	=	-23	-14	-17	-28		-14	-5	-6	-17		-5	5	6	-6
		5	14	17	6		14	23	28	17		23	32	38	28
		-28	-17	-21	-34		-17	-6	-7	-21		-6	6	7	-7
		6	17	21	7		17	28	34	21		28	38	44	34
		-34	-21	-27	-41		-21	-7	-9	-27		-7	7	9	-9
		7	21	27	9		21	34	41	27		34	44	52	41
		-41	-27	-35	-51		-27	-9	-13	-35		-9	9	13	-13
		9	27	35	13		27	41	51	35		41	52	63	51
		-51	-35	-51	-67		-35	-13	-21	-51		-13	13	21	-21
yvals	=	13	35	51	21		35	51	67	51		51	63	77	67
		29	32	32	32	29	26	39	42	42	42				
		39	35	48	51	52	51	48	43	57	61				
		62	61	57	51	65	70	72	70	65	58				
ybounds	=	23	26	36	32		26	27	37	36		27	27	37	37
		27	26	36	37		26	23	32	36		23	19	28	32
		32	36	45	41		36	37	47	45		37	37	47	47
		37	36	45	47		36	32	41	45		32	28	36	41
		41	45	55	50		45	47	57	55		47	47	57	57
		47	45	55	57		45	41	50	55		41	36	44	50
		50	55	64	58		55	57	67	64		57	57	67	67
		57	55	64	67		55	50	58	64		50	44	51	58
		58	64	72	64		64	67	77	72		67	67	77	77
		67	64	72	77		64	58	64	72		58	51	56	64

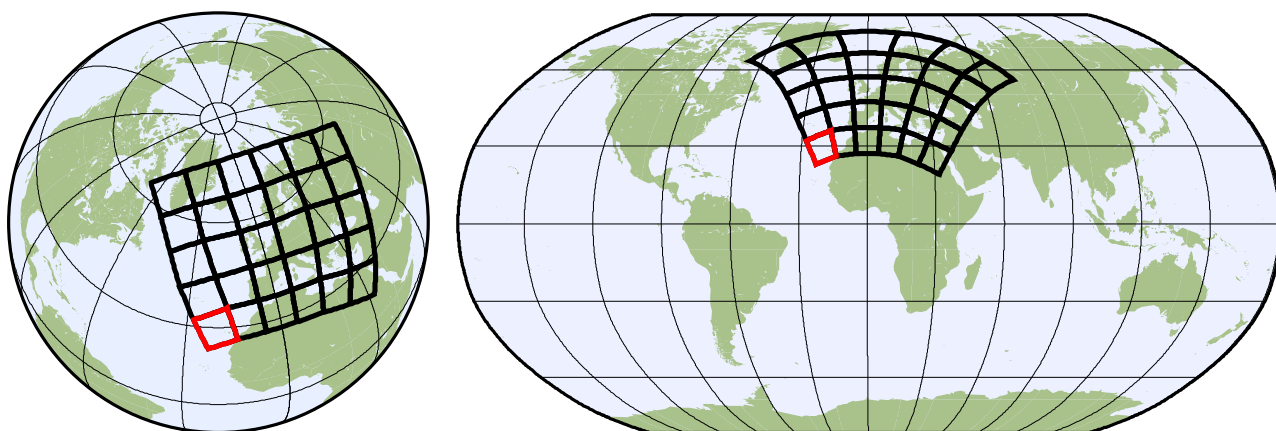


Figure D.1.: Orthographic and Robinson projection of the curvilinear grid, the first grid cell is colored red

## D.2. Example description for an unstructured grid

Here is an example of the **CDO** description for an unstructured grid. xvals/yvals describe the positions of 30 independent hexagonal grid cells. The first 6 values of xbounds/ybounds are the corners of the first grid cell. The grid cell corners have to rotate counterclockwise. The first grid cell is colored red.

```

gridtype = unstructured
gridsize = 30
nvertex = 6
xvals = -36 36 0 -18 18 108 72 54 90 180 144 126 162 -108 -144
        -162 -126 -72 -90 -54 0 72 36 144 108 -144 180 -72 -108 -36
xbounds = 339 0 0 288 288 309 21 51 72 72 0 0
          0 16 21 0 339 344 340 0 -0 344 324 324
          20 36 36 16 0 0 93 123 144 144 72 72
          72 88 93 72 51 56 52 72 72 56 36 36
          92 108 108 88 72 72 165 195 216 216 144 144
          144 160 165 144 123 128 124 144 144 128 108 108
          164 180 180 160 144 144 237 267 288 288 216 216
          216 232 237 216 195 200 196 216 216 200 180 180
          236 252 252 232 216 216 288 304 309 288 267 272
          268 288 288 272 252 252 308 324 324 304 288 288
          345 324 324 36 36 15 36 36 108 108 87 57
          20 15 36 57 52 36 108 108 180 180 159 129
          92 87 108 129 124 108 180 180 252 252 231 201
          164 159 180 201 196 180 252 252 324 324 303 273
          236 231 252 273 268 252 308 303 324 345 340 324
yvals = 58 58 32 0 0 58 32 0 0 58 32 0 0 58 32
        0 0 32 0 0 -58 -58 -32 -58 -32 -58 -32 -58 -32
ybounds = 41 53 71 71 53 41 41 41 53 71 71 53
          11 19 41 53 41 19 -19 -7 11 19 7 -11
          -19 -11 7 19 11 -7 41 41 53 71 71 53
          11 19 41 53 41 19 -19 -7 11 19 7 -11
          -19 -11 7 19 11 -7 41 41 53 71 71 53
          11 19 41 53 41 19 -19 -7 11 19 7 -11
          -19 -11 7 19 11 -7 41 41 53 71 71 53
          11 19 41 53 41 19 -19 -7 11 19 7 -11
          -19 -11 7 19 11 -7 11 19 41 53 41 19
          -19 -7 11 19 7 -11 -19 -11 7 19 11 -7
          -41 -53 -71 -71 -53 -41 -53 -71 -71 -53 -41 -41
          -19 -41 -53 -41 -19 -11 -53 -71 -71 -53 -41 -41
          -19 -41 -53 -41 -19 -11 -53 -71 -71 -53 -41 -41
          -19 -41 -53 -41 -19 -11 -53 -71 -71 -53 -41 -41
          -19 -41 -53 -41 -19 -11 -19 -41 -53 -41 -19 -11

```

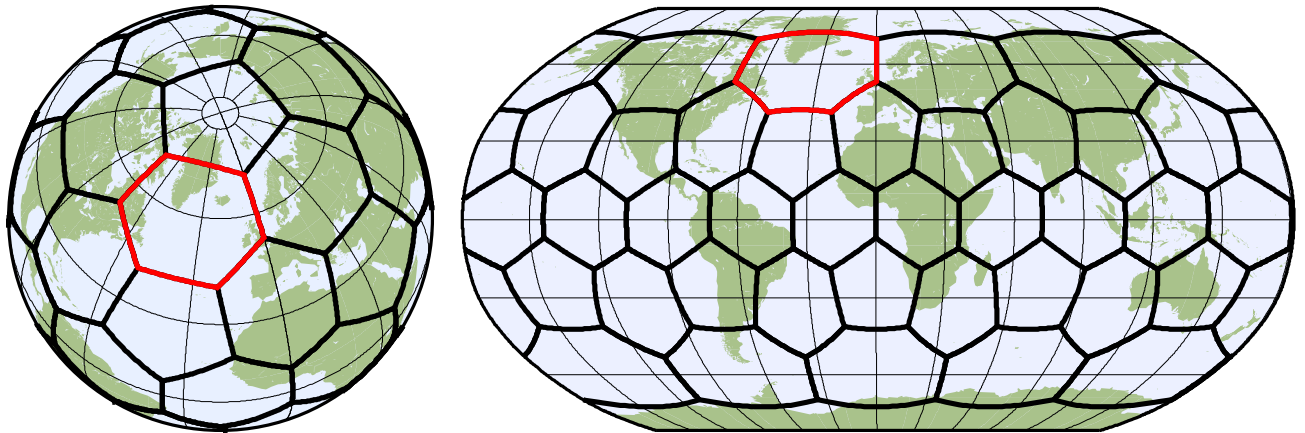


Figure D.2.: Orthographic and Robinson projection of the unstructured grid



# Operator index

## A

abs	71
acos	71
add	73
addc	72
adipot	175
adisit	175
aexpr	69
aexprf	69
after	166
ap2pl	149
asin	71
atan2	73

## B

bandpass	168
----------	-----

## C

cat	27
chcode	56
chlevel	56
chlevelc	56
chlevelv	56
chname	56
chparam	56
chunit	56
collgrid	35
consecsum	85
consects	85
const	171
copy	27
cos	71

## D

dayavg	105
daymax	105
daymean	105
daymin	105
daypctl	106
daystd	105
daystd1	105
daysum	105
dayvar	105
dayvar1	105
delcode	38
delete	37
delname	38
delparam	38
detrend	131
diff	22

diffn	22
distgrid	34
div	73
divc	72
divdpm	78
divdpy	78
duplicate	28
dv2ps	156
dv2uv	156
dv2uvl	156

## E

enlarge	64
ensavg	86
ensbrs	89
enscrps	89
ensmax	86
ensmean	86
ensmin	86
enspctl	86
ensrkhistspace	88
ensrkhisttime	88
ensroc	88
ensstd	86
ensstd1	86
enssum	86
ensvar	86
ensvar1	86
eof	134
eof3d	134
eofcoeff	136
eofspatial	134
eoftime	134
eq	47
eqc	48
exp	71
expr	69
exprf	69

## F

fdns	177
fldavg	91
fldcor	128
fldcovar	129
fldmax	91
fldmean	91
fldmin	91
fldpctl	91
fldstd	91
fldstd1	91

fldsum .....	91
fldvar .....	91
fldvar1 .....	91
for .....	171

**G**

ge .....	47
gec .....	48
genbic .....	139
genbil .....	138
gencon .....	143
gencon2 .....	144
gendis .....	141
genlaf .....	145
genlevelbounds .....	58
gennn .....	140
genycon .....	142
gp2sp .....	155
gp2spl .....	155
gradsdes .....	165
gridarea .....	169
gridboxavg .....	95
gridboxmax .....	95
gridboxmean .....	95
gridboxmin .....	95
gridboxstd .....	95
gridboxstd1 .....	95
gridboxsum .....	95
gridboxvar .....	95
gridboxvar1 .....	95
griddes .....	25
gridweights .....	169
gt .....	47
gtc .....	48

**H**

highpass .....	168
histcount .....	176
histfreq .....	176
histmean .....	176
histsum .....	176
houravg .....	103
hourmax .....	103
hourmean .....	103
hourmin .....	103
hourpctl .....	104
hourstd .....	103
hourstd1 .....	103
hoursum .....	103
hourvar .....	103
hourvar1 .....	103
hurr .....	178

**I**

ifnotthen .....	44
ifnotthenc .....	45
ifthen .....	44
ifthenc .....	45

ifthenelse .....	44
import_amsr .....	160
import_binary .....	158
import_cmsaf .....	159
info .....	20
infor .....	20
input .....	161
inputext .....	161
inputsrv .....	161
int .....	71
intlevel .....	150
intlevel3d .....	151
intlevelx3d .....	151
intntime .....	152
inttime .....	152
intyear .....	153
invertlat .....	60
invertlev .....	60

**L**

le .....	47
lec .....	48
ln .....	71
log10 .....	71
lowpass .....	168
lt .....	47
ltc .....	48

**M**

map .....	20
maskindexbox .....	62
masklonlatbox .....	62
maskregion .....	61
mastrfu .....	173
max .....	73
meravg .....	94
merge .....	29
mergegrid .....	28
mergetime .....	29
mermax .....	94
mermean .....	94
mermin .....	94
merpctl .....	94
merstd .....	94
merstd1 .....	94
mersum .....	94
mervar .....	94
mervar1 .....	94
min .....	73
ml2hl .....	149
ml2pl .....	149
monadd .....	74
monavg .....	107
monddiv .....	74
monmax .....	107
monmean .....	107
monmin .....	107
monmul .....	74

monpctl .....	108
monstd .....	107
monstd1 .....	107
monsub .....	74
monsum .....	107
monvar .....	107
monvar1 .....	107
mul .....	73
mulc .....	72
muldpm .....	78
muldpy .....	78

**N**

ndate .....	23
ne .....	47
nec .....	48
nint .....	71
nlevel .....	23
nmon .....	23
npar .....	23
ntime .....	23
nyear .....	23

**O**

output .....	162
outputext .....	162
outputf .....	162
outputint .....	162
outputsrv .....	162
outputtab .....	163

**P**

pardes .....	25
pow .....	71

**R**

random .....	171
reci .....	71
regres .....	131
remap .....	146
remapbic .....	139
remapbil .....	138
remapcon .....	143
remapcon2 .....	144
remapdis .....	141
remapeta .....	147
remaplaf .....	145
remapnm .....	140
remapycon .....	142
replace .....	28
rhopot .....	175
rotuvb .....	173
runavg .....	99
runmax .....	99
runmean .....	99
runmin .....	99
runpctl .....	100
runstd .....	99

runstd1 .....	99
runsum .....	99
runvar .....	99
runvar1 .....	99

**S**

sealevelpressure .....	174
seasavg .....	112
seasmax .....	112
seasmean .....	112
seasmin .....	112
seaspctl .....	113
seasstd .....	112
seasstd1 .....	112
seassum .....	112
seasvar .....	112
seasvar1 .....	112
selcode .....	38
seldate .....	40
selday .....	40
select .....	37
selgrid .....	38
selhour .....	40
selindexbox .....	42
sellevel .....	38
sellevidx .....	38
sellonlatbox .....	42
selltype .....	38
selmon .....	40
selname .....	38
selparam .....	38
selseas .....	40
selsmon .....	40
selstdname .....	38
seltabnum .....	38
seltime .....	40
seltimestep .....	40
selyear .....	40
selzaxis .....	38
selzaxisname .....	38
setcalendar .....	54
setcindexbox .....	63
setclonlatbox .....	63
setcode .....	53
setctomiss .....	65
setdate .....	54
setday .....	54
setgatt .....	59
setgatts .....	59
setgrid .....	57
setgridarea .....	57
setgridtype .....	57
sethalo .....	176
setlevel .....	53
setltype .....	53
setmisstoc .....	65
setmisstodis .....	65
setmisstonn .....	65

setmissval .....	65
setmon .....	54
setname .....	53
setparam .....	53
setpartab .....	53
setpartabn .....	51
setpartabp .....	51
setreftime .....	54
setrtoc .....	170
setrtoc2 .....	170
setrtomiss .....	65
settaxis .....	54
settime .....	54
setunits .....	54
setunit .....	53
setvals .....	170
setvrange .....	65
setyear .....	54
setzaxis .....	58
shifttime .....	54
showcode .....	24
showdate .....	24
showformat .....	24
showlevel .....	24
showltype .....	24
showmon .....	24
showname .....	24
showstdname .....	24
showtime .....	24
showtimestamp .....	24
showyear .....	24
sin .....	71
sinfo .....	21
sinfo .....	21
smooth9 .....	170
sp2gp .....	155
sp2gpl .....	155
sp2sp .....	155
splitcode .....	30
splitday .....	32
splitgrid .....	30
splithour .....	32
splitlevel .....	30
splitmon .....	32
splitname .....	30
splitparam .....	30
splitseas .....	32
splitsel .....	33
splittabnum .....	30
splityear .....	32
splityearmon .....	32
splitzaxis .....	30
sqr .....	71
sqrt .....	71
stdatm .....	171
strbre .....	178
strgal .....	178
strwin .....	177

sub .....	73
subc .....	72
subtrend .....	132

**T**

tan .....	71
timavg .....	101
timcor .....	128
timcovar .....	129
timmax .....	101
timmean .....	101
timmin .....	101
timpctl .....	102
timselavg .....	97
timselmax .....	97
timselmean .....	97
timselmin .....	97
timselfctl .....	98
timselfstd .....	97
timselfstd1 .....	97
timselfsum .....	97
timselfvar .....	97
timselfvar1 .....	97
timsort .....	171
timstd .....	101
timstd1 .....	101
timsum .....	101
timvar .....	101
timvar1 .....	101
topo .....	171
trend .....	132

**U**

uv2dv .....	156
uv2dvl .....	156

**V**

vct .....	25
vertavg .....	96
vertmax .....	96
vertmean .....	96
vertmin .....	96
vertstd .....	96
vertstd1 .....	96
vertsum .....	96
vertvar .....	96
vertvar1 .....	96

**W**

wct .....	177
-----------	-----

**Y**

ydayadd .....	76
ydayavg .....	116
ydaydiv .....	76
ydaymax .....	116
ydaymean .....	116
ydaymin .....	116
ydaymul .....	76

ydaypctl	118
ydaystd	116
ydaystd1	116
ydaysub	76
ydaysum	116
ydayvar	116
ydayvar1	116
ydrunavg	124
ydrunmax	124
ydrunmean	124
ydrunmin	124
ydrunpctl	126
ydrunstd	124
ydrunstd1	124
ydrunsum	124
ydrunvar	124
ydrunvar1	124
yearavg	110
yearmax	110
yearmean	110
yearmin	110
yearmonmean	109
yearpctl	111
yearstd	110
yearstd1	110
yearsum	110
yearvar	110
yearvar1	110
yhouradd	75
yhouravg	114
yhourdiv	75
yhourmax	114
yhourmean	114
yhourmin	114
yhourmul	75
yhourstd	114
yhourstd1	114
yhoursub	75
yhoursum	114
yhourvar	114
yhourvar1	114
ymonadd	77
ymonavg	119
ymonddiv	77
ymonmax	119
ymonmean	119
ymonmin	119
ymonmul	77
ymonpctl	121
ymonstd	119
ymonstd1	119
ymonsub	77
ymonsum	119
ymonvar	119
ymonvar1	119
yseasadd	78
yseasavg	122
yseasdiv	78

yseasmax	122
yseasmean	122
yseasmin	122
yseasmul	78
yseaspctl	123
yseasstd	122
yseasstd1	122
yseassub	78
yseassum	122
yseasvar	122
yseasvar1	122

## Z

zaxisdes	25
zonavg	93
zonmax	93
zonmean	93
zonmin	93
zonpctl	93
zonstd	93
zonstd1	93
zonsum	93
zonvar	93
zonvar1	93