

# CDO User's Guide

---

Climate Data Operators  
Version 1.5.2  
August 2011

Uwe Schulzweida, Luis Kornblueh – *MPI for Meteorology*  
Ralf Quast – *Brockmann Consult*

---

# 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. Operators	8
1.2.3. Combining operators	8
1.2.4. Operator parameter	9
1.3. Horizontal grids	9
1.3.1. Grid area weights	9
1.3.2. Grid description	10
1.4. Z-axis description	12
1.5. Time axis	13
1.5.1. Absolute time	13
1.5.2. Relative time	14
1.5.3. Conversion of the time	14
1.6. Parameter table	14
1.7. Missing values	14
1.7.1. Mean and average	15
<b>2. Reference manual</b>	<b>16</b>
2.1. Information	17
2.1.1. INFO - Information and simple statistics	18
2.1.2. SINFO - Short information	19
2.1.3. DIFF - Compare two datasets field by field	20
2.1.4. NINFO - Print the number of parameters, levels or times	21
2.1.5. SHOWINFO - Show variables, levels or times	22
2.1.6. FILEDES - Dataset description	23
2.2. File operations	24
2.2.1. COPY - Copy datasets	25
2.2.2. REPLACE - Replace variables	26
2.2.3. MERGE - Merge datasets	27
2.2.4. SPLIT - Split a dataset	28
2.2.5. SPLITTIME - Split time steps of a dataset	29
2.2.6. SPLITSEL - Split selected time steps	30
2.3. Selection	31
2.3.1. SELVAR - Select fields	32
2.3.2. SELTIME - Select time steps	35
2.3.3. SELBOX - Select a box of a field	37
2.4. Conditional selection	38
2.4.1. COND - Conditional select one field	39
2.4.2. COND2 - Conditional select two fields	39
2.4.3. CONDC - Conditional select a constant	40
2.5. Comparison	41
2.5.1. COMP - Comparison of two fields	42
2.5.2. COMPC - Comparison of a field with a constant	43
2.6. Modification	44
2.6.1. SET - Set field info	46

2.6.2.	SETTIME - Set time	47
2.6.3.	CHANGE - Change field header	49
2.6.4.	SETGRID - Set grid information	50
2.6.5.	SETZAXIS - Set z-axis type	51
2.6.6.	SETGATT - Set global attribute	52
2.6.7.	INVERT - Invert latitudes	53
2.6.8.	INVERTLEV - Invert levels	53
2.6.9.	MASKREGION - Mask regions	54
2.6.10.	MASKBOX - Mask a box	55
2.6.11.	SETBOX - Set a box to constant	56
2.6.12.	ENLARGE - Enlarge fields	57
2.6.13.	SETMISS - Set missing value	58
2.7.	Arithmetic	60
2.7.1.	EXPR - Evaluate expressions	61
2.7.2.	MATH - Mathematical functions	63
2.7.3.	ARITHC - Arithmetic with a constant	64
2.7.4.	ARITH - Arithmetic on two datasets	65
2.7.5.	MONARITH - Monthly arithmetic	66
2.7.6.	YMONARITH - Multi-year monthly arithmetic	67
2.7.7.	YDAYARITH - Multi-year daily arithmetic	68
2.7.8.	ARITHDAYS - Arithmetic with days	69
2.8.	Statistical values	70
2.8.1.	CONSECSTAT - Consecutive timestep periods	75
2.8.2.	ENSSTAT - Statistical values over an ensemble	76
2.8.3.	ENSSTAT2 - Statistical values over an ensemble	78
2.8.4.	ENSVAL - Ensemble validation tools	79
2.8.5.	FLDSTAT - Statistical values over a field	81
2.8.6.	ZONSTAT - Zonal statistical values	83
2.8.7.	MERSTAT - Meridional statistical values	85
2.8.8.	GRIDBOXSTAT - Statistical values over grid boxes	87
2.8.9.	VERTSTAT - Vertical statistical values	88
2.8.10.	TIMSELSTAT - Time range statistical values	89
2.8.11.	TIMSELPCTL - Time range percentile values	90
2.8.12.	RUNSTAT - Running statistical values	91
2.8.13.	RUNPCTL - Running percentile values	92
2.8.14.	TIMSTAT - Statistical values over all time steps	93
2.8.15.	TIMPCTL - Percentile values over all time steps	94
2.8.16.	HOURLSTAT - Hourly statistical values	95
2.8.17.	HOURLPCTL - Hourly percentile values	96
2.8.18.	DAYSTAT - Daily statistical values	97
2.8.19.	DAYPCTL - Daily percentile values	98
2.8.20.	MONSTAT - Monthly statistical values	99
2.8.21.	MONPCTL - Monthly percentile values	100
2.8.22.	YEARSTAT - Yearly statistical values	101
2.8.23.	YEARPCTL - Yearly percentile values	102
2.8.24.	SEASSTAT - Seasonal statistical values	103
2.8.25.	SEASPCTL - Seasonal percentile values	104
2.8.26.	YHOURLSTAT - Multi-year hourly statistical values	105
2.8.27.	YDAYSTAT - Multi-year daily statistical values	106
2.8.28.	YDAYPCTL - Multi-year daily percentile values	107
2.8.29.	YMONSTAT - Multi-year monthly statistical values	108
2.8.30.	YMONPCTL - Multi-year monthly percentile values	109
2.8.31.	YSEASSTAT - Multi-year seasonal statistical values	110
2.8.32.	YSEASPCTL - Multi-year seasonal percentile values	111
2.8.33.	YDRUNSTAT - Multi-year daily running statistical values	112
2.8.34.	YDRUNPCTL - Multi-year daily running percentile values	114
2.9.	Correlation	115
2.9.1.	FLDCOR - Correlation of grid space	116

2.9.2. TIMCOR - Correlation over time . . . . .	116
2.10. Regression . . . . .	117
2.10.1. REGRES - Regression . . . . .	118
2.10.2. DETREND - Detrend time series . . . . .	118
2.10.3. TREND - Trend of time series . . . . .	119
2.10.4. SUBTREND - Subtract a trend . . . . .	119
2.11. EOFs . . . . .	120
2.11.1. EOFs - Empirical Orthogonal Functions . . . . .	121
2.11.2. EOFCOEFF - Principal coefficients of EOFs . . . . .	123
2.12. Interpolation . . . . .	124
2.12.1. REMAPGRID - SCRIP grid interpolation . . . . .	125
2.12.2. GENWEIGHTS - Generate SCRIP grid interpolation weights . . . . .	127
2.12.3. REMAP - SCRIP grid remapping . . . . .	129
2.12.4. REMAPETA - Remap vertical hybrid level . . . . .	130
2.12.5. INTVERT - Vertical interpolation . . . . .	132
2.12.6. INTLEVEL - Linear level interpolation . . . . .	133
2.12.7. INTLEVEL3D - Linear level interpolation from/to 3d vertical coordinates . . . . .	134
2.12.8. INTTIME - Time interpolation . . . . .	135
2.12.9. INTYEAR - Year interpolation . . . . .	136
2.13. Transformation . . . . .	137
2.13.1. SPECTRAL - Spectral transformation . . . . .	138
2.13.2. WIND - Wind transformation . . . . .	140
2.14. Import/Export . . . . .	141
2.14.1. IMPORTBINARY - Import binary data sets . . . . .	142
2.14.2. IMPORTCMSAF - Import CM-SAF HDF5 files . . . . .	143
2.14.3. IMPORTAMSR - Import AMSR binary files . . . . .	144
2.14.4. INPUT - Formatted input . . . . .	145
2.14.5. OUTPUT - Formatted output . . . . .	146
2.15. Miscellaneous . . . . .	148
2.15.1. GRADSDES - GrADS data descriptor file . . . . .	149
2.15.2. FILTER - Time series filtering . . . . .	150
2.15.3. GRIDCELL - Grid cell quantities . . . . .	151
2.15.4. SMOOTH9 - 9 point smoothing . . . . .	152
2.15.5. REPLACEVALUES - Replace variable values . . . . .	152
2.15.6. TIMSORT - Timsort . . . . .	153
2.15.7. VARGEN - Generate a field . . . . .	153
2.15.8. ROTUV - Rotation . . . . .	155
2.15.9. MASTRFU - Mass stream function . . . . .	155
2.15.10.HISTOGRAM - Histogram . . . . .	156
2.15.11. SETHALO - Set the left and right bounds of a field . . . . .	156
2.15.12.WCT - Windchill temperature . . . . .	157
2.15.13.FDNS - Frost days where no snow index per time period . . . . .	157
2.15.14.STRWIN - Strong wind days index per time period . . . . .	157
2.15.15.STRBRE - Strong breeze days index per time period . . . . .	158
2.15.16.STRGAL - Strong gale days index per time period . . . . .	158
2.15.17.HURR - Hurricane days index per time period . . . . .	158
2.16. Climate indices . . . . .	159
2.16.1. ECACDD - Consecutive dry days index per time period . . . . .	161
2.16.2. ECACFD - Consecutive frost days index per time period . . . . .	161
2.16.3. ECACSU - Consecutive summer days index per time period . . . . .	162
2.16.4. ECACWD - Consecutive wet days index per time period . . . . .	162
2.16.5. ECACWDI - Cold wave duration index wrt mean of reference period . . . . .	163
2.16.6. ECACWFI - Cold-spell days index wrt 10th percentile of reference period . . . . .	163
2.16.7. ECAETR - Intra-period extreme temperature range . . . . .	165
2.16.8. ECAFD - Frost days index per time period . . . . .	165
2.16.9. ECAGSL - Thermal Growing season length index . . . . .	166
2.16.10.ECAHD - Heating degree days per time period . . . . .	167
2.16.11.ECAHWDI - Heat wave duration index wrt mean of reference period . . . . .	167

2.16.12.ECAHWFI - Warm spell days index wrt 90th percentile of reference period . . . . .	169
2.16.13.ECAID - Ice days index per time period . . . . .	169
2.16.14.ECAPD - Precipitation days index per time period . . . . .	170
2.16.15.ECAR75P - Moderate wet days wrt 75th percentile of reference period . . . . .	171
2.16.16.ECAR75PTOT - Precipitation percent due to R75p days . . . . .	171
2.16.17.ECAR90P - Wet days wrt 90th percentile of reference period . . . . .	172
2.16.18.ECAR90PTOT - Precipitation percent due to R90p days . . . . .	172
2.16.19.ECAR95P - Very wet days wrt 95th percentile of reference period . . . . .	173
2.16.20.ECAR95PTOT - Precipitation percent due to R95p days . . . . .	173
2.16.21.ECAR99P - Extremely wet days wrt 99th percentile of reference period . . . . .	174
2.16.22.ECAR99PTOT - Precipitation percent due to R99p days . . . . .	174
2.16.23.ECARR1 - Wet days index per time period . . . . .	175
2.16.24.ECARX1DAY - Highest one day precipitation amount per time period . . . . .	175
2.16.25.ECARX5DAY - Highest five-day precipitation amount per time period . . . . .	176
2.16.26.ECASDII - Simple daily intensity index per time period . . . . .	176
2.16.27.ECASU - Summer days index per time period . . . . .	177
2.16.28.ECATG10P - Cold days percent wrt 10th percentile of reference period . . . . .	178
2.16.29.ECATG90P - Warm days percent wrt 90th percentile of reference period . . . . .	178
2.16.30.ECATN10P - Cold nights percent wrt 10th percentile of reference period . . . . .	179
2.16.31.ECATN90P - Warm nights percent wrt 90th percentile of reference period . . . . .	179
2.16.32.ECATR - Tropical nights index per time period . . . . .	180
2.16.33.ECATX10P - Very cold days percent wrt 10th percentile of reference period . . . . .	180
2.16.34.ECATX90P - Very warm days percent wrt 90th percentile of reference period . . . . .	181
<b>A. Grid description examples</b>	<b>183</b>
A.1. Example of a curvilinear grid description . . . . .	183
A.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 NWP model output. 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. There are also some operators with the same name as in PINGO but with a different meaning. Appendix A gives an overview of those operators.

The main **CDO** features are:

- More than 400 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](http://code.zmaw.de/projects/cdo) page (<http://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 1.9.5 or higher. This library is needed to process GRIB2 files with **CDO**.
- HDF5 [gzip](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 gzip 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](http://trac.osgeo.org/proj) 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.

### 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>
```

- The GRIB2 configuration depends on the GRIB\_API installation! Here is an example GRIB2 configuration with a JASPER enabled GRIB\_API version:

```
./configure --with-grib_api=<GRIB_API root directory> \
            --with-jasper=<JASPER 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:

<i>&lt;format&gt;</i>	<i>&lt;nbits&gt;</i>
grb, grb2	1 - 24
nc, nc2, nc4, nc4c	I8/I16/I32/F32/F64
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	<i>&lt;format&gt;</i>
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 10). Available grid names are: *r<NX>x<NY>*, *lon=<LON>/lat=<LAT>*, *n<N>*, *gme<NI>*
- h Help information for the operators.
- 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*
- Q Alphanumeric sorting of netCDF parameter names.
- R Convert GRIB1 data from reduced to regular grid (only with cgribex lib).
- r Generate a relative time axis.
- s Silent mode.
- t *<partab>* Set the default parameter table name or file (see chapter 1.6 on page 14).  
Predefined tables are: *echam4 echam5 echam6 mpiom1 ecmwf remo*
- V Print the version number.
- v Print extra details for some operators.
- z *gzip* SZIP compression of GRIB1 records.
- jpeg* JPEG compression of GRIB2 records.
- zip* Deflate compression of netCDF4 variables.

### 1.2.2. Operators

There are more than 400 operators available. A detailed description of all operators can be found in the [Reference Manual](#) section.

### 1.2.3. Combining operators

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
```

Combining of operators is implemented over POSIX Threads (pthread). Therefore this **CDO** feature is not available on operating systems without POSIX Threads support.

### 1.2.4. Operator parameter

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

- STRING

Unquoted characters without blanks and tabs. The following command select the variables with the names `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.

### 1.3.1. Grid area weights

One single point of a 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:** `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° with an increment of  $(360/\text{<NX>})^\circ$ . The latitudes go from south to north with an increment of  $(180/\text{<NY>})^\circ$ .

**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° with an increment of  $(360/\text{nlon})^\circ$ . 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 this program. 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>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
<b>yfirst, yinc</b>	FLOAT, FLOAT	macros to define yvals with a constant increment

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			
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 B](#).

## 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. Some time range statistic operators create an absolute time axis (e.g. `timsel<STAT>`, `run<STAT>`, `seas<STAT>`, ...).

### 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  $-9e + 33$ . 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.

## 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 unlimited 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 time steps
<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 time steps. 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
  - Parameter identifier and Level
  - Size of the grid and number of Missing values
  - Minimum, Mean and Maximum
- The mean value is computed without the use of area weights!
- infor** 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 infor ifile
```

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

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

## 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 unlimited number of input files. All input files need to have the same structure with the same variables on different time steps. 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
  - parameter identifier
  - horizontal grid size and number
  - number of vertical levels and z-axis number
- Section 2 and 3 gives a short overview of all horizontal and vertical grids. And the last section contains short information of the time axis.
- 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 time steps:

```

-1 : Institut Source Name Time Typ Grid Size Num Levels Num
 1 : MPIMET ECHAM5 GEOSP con F32 2048 1 1 1
 2 : MPIMET ECHAM5 T var F32 2048 1 4 2
 3 : MPIMET ECHAM5 TSURF var F32 2048 1 1 1
Horizontal grids :
 1 : gaussian > size : dim = 2048 nlon = 64 nlat = 32
                        longitude : first = 0 last = 354.375 inc = 5.625
                        latitude : first = 85.7605871 last = -85.7605871
Vertical grids :
 1 : surface : 0
 2 : pressure Pa : 92500 85000 50000 20000
Time axis : 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
- Parameter identifier and Level
- Size of the grid and number of Missing 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

$$Absdiff(t, x) = |i_1(t, x) - i_2(t, x)|$$

$$Reldiff(t, x) = \frac{|i_1(t, x) - i_2(t, x)|}{\max(|i_1(t, x)|, |i_2(t, x)|)}$$

**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 time steps:

	Date	Time	Name	Level	Size	Miss	: S Z	Max_Absdiff	Max_Reldiff
1	: 1987-01-31	12:00:00	SST	0	2048	1361	: F F	0.00010681	4.1660e-07
2	: 1987-02-28	12:00:00	SST	0	2048	1361	: F F	6.1035e-05	2.3742e-07
3	: 1987-03-31	12:00:00	SST	0	2048	1361	: F F	7.6294e-05	3.3784e-07
4	: 1987-04-30	12:00:00	SST	0	2048	1361	: F F	7.6294e-05	3.5117e-07
5	: 1987-05-31	12:00:00	SST	0	2048	1361	: F F	0.00010681	4.0307e-07
6	: 1987-06-30	12:00:00	SST	0	2048	1361	: F F	0.00010681	4.2670e-07
7	: 1987-07-31	12:00:00	SST	0	2048	1361	: F F	9.1553e-05	3.5634e-07
8	: 1987-08-31	12:00:00	SST	0	2048	1361	: F F	7.6294e-05	2.8849e-07
9	: 1987-09-30	12:00:00	SST	0	2048	1361	: F F	7.6294e-05	3.6168e-07
10	: 1987-10-31	12:00:00	SST	0	2048	1361	: F F	9.1553e-05	3.5001e-07
11	: 1987-11-30	12:00:00	SST	0	2048	1361	: F F	6.1035e-05	2.3839e-07
12	: 1987-12-31	12:00:00	SST	0	2048	1361	: F F	9.3553e-05	3.7624e-07

## 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 time steps Prints the number of time steps.

### 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 time steps (format YYYY-MM-DD).
<b>showtime</b>	Show time information Prints time information of all time steps (format hh:mm:ss).
<b>showtimestamp</b>	Show timestamp Prints timestamp of all time steps (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>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>splitmon</b>	Split months
<b>splitseas</b>	Split seasons
<b>splityear</b>	Split years
<b>splitsel</b>	Split time selection



### 2.2.1. COPY - Copy datasets

#### Synopsis

```
<operator> ifiles ofile
```

#### Description

This module contains operators to copy or concatenate datasets. **ifiles** is an unlimited number of input files. All input files need to have the same structure with the same variables on different time steps.

#### Operators

<b>copy</b>	Copy datasets Copies all input datasets to <b>ofile</b> .
<b>cat</b>	Concatenate datasets Concatenates all input datasets and appends the result to the end of <b>ofile</b> . If <b>ofile</b> 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 time steps 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 time steps use:

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

## 2.2.2. REPLACE - Replace variables

### Synopsis

```
replace ifile1 ifile2 ofile
```

### Description

Replaces all common fields of `ifile2` and `ifile1` with those of `ifile1` and write the result to `ofile`. Both input datasets need to have the same number of time steps.

### 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` with `tslm1` from `ifile2` use:

```
cdo replace ifile1 ifile2 ofile
```

### 2.2.3. 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>	Merge datasets with different fields Merges time series of different fields from several input datasets. The number of fields per time step written to <code>ofile</code> is the sum of the field numbers per time step in all input datasets. The time series on all input datasets are required to have different fields and the same number of time steps.
<b>mergetime</b>	Merge datasets sorted by date and time Merges all time steps of all input files sorted by date and time. All input files need to have the same structure with the same variables on different time steps. After this operation every input time step is in <code>ofile</code> and all time steps are sorted by date and time.

#### Example

Assume three datasets with the same number of time steps 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
```

## 2.2.4. SPLIT - Split a dataset

### Synopsis

```
<operator> 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.

### Operators

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

### Environment

<code>CDO_FILE_SUFFIX</code>	This environment variable can be used to 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 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
```

## 2.2.5. SPLITTIME - Split time steps of a dataset

### Synopsis

```
<operator> ifile obase
```

### Description

This module splits `ifile` into time steps 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>splitmon</b>	Split months Splits a file into pieces, one for each different month. <code>xxx</code> will have two digits with the month.
<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.

### Environment

<b>CDO_FILE_SUFFIX</b>	This environment variable can be used to 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 the input GRIB1 dataset has time steps 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
```

## 2.2.6. SPLITSEL - Split selected time steps

### 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 time steps 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 time steps for each output file
<i>noffset</i>	INTEGER	Number of input time steps skipped before the first time step range (optional)
<i>nskip</i>	INTEGER	Number of input time steps skipped between time step ranges (optional)

### Environment

CDO_FILE_SUFFIX	This environment variable can be used to 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.
-----------------	--

## 2.3. 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>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>selleidx</b>	Select levels by index
<b>selgrid</b>	Select grids
<b>selzaxis</b>	Select z-axes
<b>selltype</b>	Select GRIB level types
<b>seltabnum</b>	Select parameter table numbers
<b>sel timestep</b>	Select time steps
<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. SELVAR - Select fields

#### Synopsis

**selparam, params** ifile ofile

**delparam, 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

**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>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 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 names or numbers
<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
```

### 2.3.2. SELTIME - Select time steps

#### Synopsis

```

se 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 smon,month[,nts1[,nts2]] ifile ofile

```

#### Description

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

#### Operators

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

**Parameter**

<i>timesteps</i>	INTEGER	Comma separated list of time steps
<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)
<i>nts1</i>	INTEGER	Number of time steps before the selected month [default: 0]
<i>nts2</i>	INTEGER	Number of time steps after the selected month [default: nts1]

### 2.3.3. 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 rectangular understood field. All input fields need to have the same horizontal grid.

#### Operators

<b>sellonlatbox</b>	Select a longitude/latitude box Selects a longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box.
<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 time step of **ifile2** or only one. The fields in **ofile** inherit the meta data from **ifile2**.

#### Operators

<b>ifthen</b>	<p>If then</p> $o(t, x) = \begin{cases} i_2(t, x) & \text{if } i_1([t, ]x) \neq 0 \quad \wedge \quad i_1([t, ]x) \neq \text{miss} \\ \text{miss} & \text{if } i_1([t, ]x) = 0 \quad \vee \quad i_1([t, ]x) = \text{miss} \end{cases}$
<b>ifnotthen</b>	<p>If not then</p> $o(t, x) = \begin{cases} i_2(t, x) & \text{if } i_1([t, ]x) = 0 \quad \wedge \quad i_1([t, ]x) \neq \text{miss} \\ \text{miss} & \text{if } i_1([t, ]x) \neq 0 \quad \vee \quad i_1([t, ]x) = \text{miss} \end{cases}$

#### 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 time step 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 time step 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

**eqc** Equal constant

$$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) = c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) \neq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$$

**nec** Not equal constant

$$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \neq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) = c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$$

**lec** Less equal constant

$$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \leq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) > c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$$

**ltc** Less than constant

$$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) < c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) \geq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$$

**gec** Greater equal constant

$$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \geq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) < c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$$

**gtc** 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>setpartab</b>	Set parameter table
<b>setcode</b>	Set code number
<b>setparam</b>	Set parameter identifier
<b>setname</b>	Set variable name
<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>setunits</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 time steps
<b>chcode</b>	Change code number
<b>chparam</b>	Change parameter identifier
<b>chname</b>	Change variable name
<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>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>setvrang</b>	Set valid range

## 2.6.1. SET - Set field info

### Synopsis

```
setpartab,table ifile ofile
setcode,code ifile ofile
setparam,param ifile ofile
setname,name 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>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 (format: code[.tabnum] or num[.cat[.dis]])
<i>name</i>	STRING	Variable name
<i>level</i>	FLOAT	New level
<i>ltype</i>	INTEGER	GRIB level type

### Example

To assign the parameter table echam5 to the input dataset use:

```
cdo setpartab,echam5 ifile ofile
```

## 2.6.2. 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 time step to the same given value.
<b>settime</b>	Set time of the day Sets the time in every time step to the same given value.
<b>setday</b>	Set day Sets the day in every time step to the same given value.
<b>setmon</b>	Set month Sets the month in every time step to the same given value.
<b>setyear</b>	Set year Sets the year in every time step 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 time steps Shifts all time steps 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, 360days, 365days, 366days)
<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 time step use:

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

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

```
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
```



### 2.6.3. CHANGE - Change field header

#### Synopsis

```

chcode,oldcode,newcode[,...] ifile ofile
chparam,oldparam,newparam,... ifile ofile
chname,oldname,newname,... 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>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
```

## 2.6.4. 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: curvilinear      Converts regular grid to curvilinear grid unstructured    Converts grid type to unstructured grid dereference     Dereference grid type REFERENCE regular          Converts reduced Gaussian grid to regular Gaussian 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 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
```

### 2.6.5. SETZAXIS - Set z-axis type

#### Synopsis

```
setzaxis,zaxis ifile ofile
```

#### Description

This operator sets the z-axis description of all variables with the same number of level as the new z-axis.

#### Parameter

<i>zaxis</i>	STRING	Z-axis description file or name of the target z-axis
--------------	--------	--

## 2.6.6. 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" ;
}
```

### 2.6.7. INVERT - Invert latitudes

#### Synopsis

```
invertlat ifile ofile
```

#### Description

This operator inverts the latitudes of all fields with a regular lon/lat grid.

#### Example

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

```
cdo invertlat ifile ofile
```

### 2.6.8. INVERTLEV - Invert levels

#### Synopsis

```
invertlev ifile ofile
```

#### Description

This operator inverts the levels of all non hybrid 3D variables.

## 2.6.9. 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. 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
```

## 2.6.10. MASKBOX - Mask a box

### Synopsis

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

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

### Description

Masks a box of the rectangular 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 Masks a longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box.
<b>maskindexbox</b>	Mask an index box Masks 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
```

## 2.6.11. 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 rectangular 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
```



## 2.6.12. 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

*grid*      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
```

## 2.6.13. 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
```

### 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
	$o(t, x) = \begin{cases} \text{miss} & \text{if } i(t, x) < rmin \vee i(t, x) > rmax \\ i(t, x) & \text{if } i(t, x) \geq rmin \wedge i(t, x) \leq rmax \end{cases}$

### Parameter

<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 from script file
<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>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>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>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

### Description

This module arithmetically processes every time step of the input dataset. Each individual assignment statement have to end with a semi-colon. The basic arithmetic operations addition +, subtraction −, multiplication \*, division / and exponentiation ^ can be used. The following intrinsic functions are available:

abs(x)	Absolute value of 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
tan(x)	Tangent of x, where x is specified in radians
asin(x)	Arc-sine of x, where x is specified in radians
acos(x)	Arc-cosine of x, where x is specified in radians
atan(x)	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 from script file Contrary to <a href="#">expr</a> the processing instructions are read from a file.

### Parameter

<i>instr</i>	STRING	Processing instructions (without spaces!)
<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`. One of the input files can contain only one time step or one field. The fields in `ofile` inherit the meta data from `ifile1` or `ifile2`.

### 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 time step with the same month and year. For each field in `ifile1` the corresponding field of the time step 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. YMONARITH - Multi-year monthly arithmetic

### Synopsis

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

### Description

This module performs simple arithmetic of a time series and one time step with the same month of year. For each field in `ifile1` the corresponding field of the time step 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
```

## 2.7.7. YDAYARITH - Multi-year daily arithmetic

### Synopsis

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

### Description

This module performs simple arithmetic of a time series and one time step with the same day of year. For each field in `ifile1` the corresponding field of the time step 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. ARITHDAYS - Arithmetic with days

### Synopsis

`<operator> ifile ofile`

### Description

This module multiplies or divides each time step 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 $o(t, x) = i(t, x) / days\_per\_year$

### Example

Assume an input dataset is a monthly mean time series. To compute the yearly mean from the correct weighted monthly mean use:

```
cdo muldpm ifile tmpfile1
cdo yearsum tmpfile1 tmpfile2
cdo divdpy tmpfile2 ofile
```

Or all in one command line:

```
cdo divdpy -yearsum -muldpm ifile ofile
```

## 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.

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>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>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>ensvar</b>	Ensemble variance
<b>ensstd</b>	Ensemble standard deviation
<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>fldvar</b>	Field variance
<b>fldstd</b>	Field standard deviation
<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>zonvar</b>	Zonal variance
<b>zonstd</b>	Zonal standard deviation
<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>mervar</b>	Meridional variance
<b>merstd</b>	Meridional standard deviation
<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>gridboxvar</b>	Gridbox variance
<b>gridboxstd</b>	Gridbox standard deviation

<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>vertvar</b>	Vertical variance
<b>vertstd</b>	Vertical standard deviation
<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>timselvar</b>	Time range variance
<b>timselstd</b>	Time range standard deviation
<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>runvar</b>	Running variance
<b>runstd</b>	Running standard deviation
<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>timvar</b>	Time variance
<b>timstd</b>	Time standard deviation
<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>hourvar</b>	Hourly variance
<b>hourstd</b>	Hourly standard deviation
<b>hourpctl</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>dayvar</b>	Daily variance
<b>daystd</b>	Daily standard deviation



<b>daypctl</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>monvar</b>	Monthly variance
<b>monstd</b>	Monthly standard deviation
<b>monpctl</b>	Monthly percentiles
<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>yearvar</b>	Yearly variance
<b>yearstd</b>	Yearly standard deviation
<b>yearpctl</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>seasvar</b>	Seasonal variance
<b>seasstd</b>	Seasonal standard deviation
<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>yhourvar</b>	Multi-year hourly variance
<b>yhourstd</b>	Multi-year hourly standard deviation
<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>ydayvar</b>	Multi-year daily variance
<b>ydaystd</b>	Multi-year daily standard deviation
<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>ymonvar</b>	Multi-year monthly variance
<b>ymonstd</b>	Multi-year monthly standard deviation

<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>yseasvar</b>	Multi-year seasonal variance
<b>yseasstd</b>	Multi-year seasonal standard deviation
<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>ydrunvar</b>	Multi-year daily running variance
<b>ydrunstd</b>	Multi-year daily running standard deviation
<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 time steps 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>	<p>Consecutive Sum</p> <p>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.</p>
<b>consects</b>	<p>Consecutive Timesteps</p> <p>In contrast to the operator above <code>consects</code> 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.</p>

### 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

```

ensmin ifiles ofile
ensmax ifiles ofile
enssum ifiles ofile
ensmean ifiles ofile
ensavg ifiles ofile
ensvar ifiles ofile
ensstd 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, a certain percentile over all input files or a skill score is written to `ofile`. All input files need to have the same structure with the same variables. The date information of a time step 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>ensvar</b>	Ensemble variance $o(t, x) = \mathbf{var}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensstd</b>	Ensemble standard deviation $o(t, x) = \mathbf{std}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<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 time step 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 time steps.

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 time step 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 time step 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 time step 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

<code>enscrps</code>	Ensemble CRPS and decomposition
<code>ensbrs</code>	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

```
fldmin ifile ofile
fldmax ifile ofile
fldsum ifile ofile
fldmean ifile ofile
fldavg ifile ofile
fldvar ifile ofile
fldstd 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>fldvar</b>	Field variance 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>fldstd</b>	Field standard deviation 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>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

```

zonmin ifile ofile

zonmax ifile ofile

zonsum ifile ofile

zonmean ifile ofile

zonavg ifile ofile

zonvar ifile ofile

zonstd 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. All input fields need to have the same regular lonlat 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>zonvar</b>	Zonal variance For every latitude the variance over all longitudes is computed.
<b>zonstd</b>	Zonal standard deviation For every latitude the standard deviation over all longitudes is computed.
<b>zonpctl</b>	Zonal percentiles For every latitude the pth percentile over all longitudes is computed.

### Parameter

<i>p</i>	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

```

mermin ifile ofile
mermax ifile ofile
mersum ifile ofile
mermean ifile ofile
meravg ifile ofile
mervar ifile ofile
merstd 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`. All input fields need to have 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>mervar</b>	Meridional variance For every longitude the variance over all latitudes is computed.
<b>merstd</b>	Meridional standard deviation For every longitude the standard deviation over all latitudes is computed.
<b>merpctl</b>	Meridional percentiles For every longitude the <i>p</i> th percentile over all latitudes is computed.

### Parameter

<i>p</i>	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

<code>gridboxmin</code>	Gridbox minimum
<code>gridboxmax</code>	Gridbox maximum
<code>gridboxsum</code>	Gridbox sum
<code>gridboxmean</code>	Gridbox mean
<code>gridboxavg</code>	Gridbox average
<code>gridboxvar</code>	Gridbox variance
<code>gridboxstd</code>	Gridbox standard deviation

### Parameter

<code>nx</code>	INTEGER	Number of grid boxes in x direction
<code>ny</code>	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 mean over all levels is computed.
<b>vertavg</b>	Vertical average For every gridpoint the average over all levels is computed.
<b>vertvar</b>	Vertical variance For every gridpoint the variance over all levels is computed.
<b>vertstd</b>	Vertical standard deviation For every gridpoint the standard deviation over all levels is computed.

### 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 time steps. According to the chosen operator the minimum, maximum, sum, average, variance or standard deviation of the selected time steps is written to `ofile`. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Operators

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

### Parameter

<i>nsets</i>	INTEGER	Number of input time steps for each output time step
<i>noffset</i>	INTEGER	Number of input time steps skipped before the first time step range (optional)
<i>nskip</i>	INTEGER	Number of input time steps skipped between time step 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 time steps 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 date information of a time step in `ofile` is the date of the last contributing time step in `ifile`.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps 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 time steps for each output time step
<i>noffset</i>	INTEGER	Number of input time steps skipped before the first time step range (optional)
<i>nskip</i>	INTEGER	Number of input time steps skipped between time step 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 time steps. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of a selected number of consecutive time steps read from **ifile** is written to **ofile**. The date information in **ofile** is the date of the middle contributing time step in **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>runvar</b>	Running variance $o(t + (nts - 1)/2, x) = \mathbf{std}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runstd</b>	Running standard deviation $o(t + (nts - 1)/2, x) = \mathbf{std}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$

### Parameter

**nts**     INTEGER     Number of time steps

### Environment

**RUNSTAT.DATE**     Sets the date information in ofile to the "first", "last" or "middle" contributing time step in ifile.

### Example

To compute the running mean over 9 time steps use:

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

### 2.8.13. RUNPCTL - Running percentile values

#### Synopsis

```
runpctl,p,nts ifile1 ofile
```

#### Description

This module computes running percentiles over a selected number of time steps in **ifile1**. The date information in **ofile** is the date of the medium contributing time step in **ifile1**.

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

#### Parameter

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

#### Example

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

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

## 2.8.14. TIMSTAT - Statistical values over all time steps

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over all time steps in `ifile`. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of all time steps read from `ifile` is written to `ofile`. The date information of a time step in `ofile` is the date of the last contributing time step in `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>timvar</b>	Time variance $o(1, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timstd</b>	Time standard deviation $o(1, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the mean over all input time steps use:

```
cdo timmean ifile ofile
```

## 2.8.15. TIMPCTL - Percentile values over all time steps

### Synopsis

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

### Description

This operator computes percentiles over all time steps 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 date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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

### 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 90th percentile over all input time steps 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 time steps of the same hour. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same hour is written to `ofile`. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Operators

<b>hourmin</b>	Hourly minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps 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 time steps 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 time steps 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 time steps 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 time steps of the same hour it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourvar</b>	Hourly variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourstd</b>	Hourly standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### 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 time steps 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 date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps 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 time steps of the same day. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same day is written to `ofile`. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Operators

<b>daymin</b>	Daily minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps 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 time steps 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 time steps 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 time steps 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 time steps of the same day it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>dayvar</b>	Daily variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daystd</b>	Daily standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### 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 time steps 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 date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same day 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 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 time steps of the same month. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same month is written to `ofile`. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Operators

<b>monmin</b>	Monthly minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps 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 time steps 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 time steps 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 time steps 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 time steps of the same month it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monvar</b>	Monthly variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monstd</b>	Monthly standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### 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 time steps 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 date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same month 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 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. YEARSTAT - Yearly statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over time steps of the same year. Depending on the chosen operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same year is written to `ofile`. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Operators

<b>yearmin</b>	Yearly minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps 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 time steps 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 time steps 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 time steps 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 time steps of the same year it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearvar</b>	Yearly variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearstd</b>	Yearly standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### 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 divdpy -yearsum -muldpm ifile ofile
```

## 2.8.23. YEARPCTL - Yearly percentile values

### Synopsis

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

### Description

This operator computes percentiles over all time steps 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 date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps 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
```

## 2.8.24. SEASSTAT - Seasonal statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

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

### Operators

<b>seasmin</b>	Seasonal minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps 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 time steps 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 time steps 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 time steps 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 time steps of the same season it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasvar</b>	Seasonal variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasstd</b>	Seasonal standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the seasonal mean of a time series use:

```
cdo seasmean ifile ofile
```

## 2.8.25. SEASPCTL - Seasonal percentile values

### Synopsis

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

### Description

This operator computes percentiles over all time steps 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 date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. Be careful about the first and the last output time step, they may be incorrect values if the seasons have incomplete time steps.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same season 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 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
```



## 2.8.26. 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 $o(0001, x) = \min\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \min\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourmax</b>	Multi-year hourly maximum $o(0001, x) = \max\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \max\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhoursum</b>	Multi-year hourly sum $o(0001, x) = \text{sum}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \text{sum}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourmean</b>	Multi-year hourly mean $o(0001, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhouravg</b>	Multi-year hourly average $o(0001, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourvar</b>	Multi-year hourly variance $o(0001, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourstd</b>	Multi-year hourly standard deviation $o(0001, x) = \text{std}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \text{std}\{i(t, x), \text{day}(i(t)) = 8784\}$

### Example

To compute the hourly mean for all days over all input years use:

```
cdo yhourmean ifile ofile
```

## 2.8.27. 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 $o(001, x) = \min\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \min\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaymax</b>	Multi-year daily maximum $o(001, x) = \max\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \max\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaysum</b>	Multi-year daily sum $o(001, x) = \text{sum}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{sum}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaymean</b>	Multi-year daily mean $o(001, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydayavg</b>	Multi-year daily average $o(001, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydayvar</b>	Multi-year daily variance $o(001, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaystd</b>	Multi-year daily standard deviation $o(001, x) = \text{std}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{std}\{i(t, x), \text{day}(i(t)) = 366\}$

### Example

To compute the daily mean over all input years use:

```
cdo ydaymean ifile ofile
```

## 2.8.28. 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
```

## 2.8.29. 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 $o(01, x) = \mathbf{min}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \mathbf{min}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonmax</b>	Multi-year monthly maximum $o(01, x) = \mathbf{max}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \mathbf{max}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonsum</b>	Multi-year monthly sum $o(01, x) = \mathbf{sum}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \mathbf{sum}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonmean</b>	Multi-year monthly mean $o(01, x) = \mathbf{mean}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \mathbf{mean}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonavg</b>	Multi-year monthly average $o(01, x) = \mathbf{avg}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \mathbf{avg}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonvar</b>	Multi-year monthly variance $o(01, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymonstd</b>	Multi-year monthly standard deviation $o(01, x) = \mathbf{std}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \mathbf{std}\{i(t, x), \text{month}(i(t)) = 12\}$

### Example

To compute the monthly mean over all input years use:

```
cdo ymonmean ifile ofile
```

## 2.8.30. 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
```

## 2.8.31. 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>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>yseasstd</b>	Multi-year seasonal standard deviation $o(1, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$

## Example

To compute the seasonal mean over all input years use:

```
cdo yseasmean ifile ofile
```

## 2.8.32. 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.

$$\begin{aligned} o(1, x) &= \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\} \\ o(2, x) &= \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\} \\ o(3, x) &= \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\} \\ o(4, x) &= \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\} \end{aligned}$$

### 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 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
```

## 2.8.33. 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 time steps in running windows of which the medium time step corresponds to a certain day of year is computed. The date information in an output field is the date of the medium time step in 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  $(nts-1)/2$  time steps after the first time step of the input time series and ends  $(nts-1)/2$  time steps 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 $o(001, x) = \text{sum}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{sum}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<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>ydrunvar</b>	Multi-year daily running variance $o(001, x) = \text{var}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{var}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunstd</b>	Multi-year daily running standard deviation $o(001, x) = \text{std}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{std}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$



## Parameter

*nts*     INTEGER     Number of time steps

## Example

Assume the input data provide a continuous time series of daily measurements. To compute the running multi-year daily mean over all input time steps 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
```

## 2.8.34. 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 time steps in running windows of which the medium time step 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 medium time step in 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  $(nts-1)/2$  time steps after the first time step of the input time series and ends  $(nts-1)/2$  time steps 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.

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

### Parameter

<code>p</code>	FLOAT	Percentile number in 0, ..., 100
<code>nts</code>	INTEGER	Number of time steps

### Environment

<code>CDO_PCTL_NBINS</code>	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 time steps 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

This sections contains modules for correlation in space and time.

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

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

### 2.9.1. FLDCOR - Correlation of 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

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

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 stream. For every timestep  $t$  only those field elements  $x$  belong to the sample, which have  $i_1(t, x) \neq \text{missval}$  and  $i_2(t, x) \neq \text{missval}$ .

### 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

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

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  $i_1(t, x) \neq \text{missval}$  and  $i_2(t, x) \neq \text{missval}$ .

## 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 time steps  $t$  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 time steps  $t$  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 time steps 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 time steps  $t$  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 time steps.

#### 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 gridsizes). 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 time steps 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. Note, that the resulting EOF in `ifile2` is  $e_j$  and thus **not weighted** for consistency.

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>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. Weights are taken into account, which is why EOF output is **not** weighted. Note, that this operator calculates a weighted dot product of the fields in `ifile1` and `ifile2`. Given a set of EOFs  $e_j$  and a time series of data  $z(t)$  with  $p$  entries for each time step 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 (time step) in `ifile1` and `suffix` is the filename extension derived from the file format.

### Environment

`CDO_FILE_SUFFIX` This environment variable can be used to 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:

```
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:

```
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>remapbic</b>	Bicubic interpolation
<b>remapdis</b>	Distance-weighted average remapping
<b>remapnn</b>	Nearest neighbor remapping
<b>remapcon</b>	First order conservative remapping
<b>remapcon2</b>	Second order conservative remapping
<b>remaplaf</b>	Largest area fraction remapping
<b>genbil</b>	Generate bilinear interpolation weights
<b>genbic</b>	Generate bicubic interpolation weights
<b>gendis</b>	Generate distance-weighted average remap weights
<b>gennn</b>	Generate nearest neighbor remap weights
<b>gencon</b>	Generate 1st order conservative remap weights
<b>gencon2</b>	Generate 2nd order conservative remap weights
<b>genlaf</b>	Generate largest area fraction remap weights
<b>remap</b>	SCRIP 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>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 time steps
<b>intntime</b>	Interpolation between time steps
<b>intyear</b>	Interpolation between two years

## 2.12.1. REMAPGRID - SCRIP grid interpolation

### Synopsis

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

### Description

This module contains operators to remap all input fields to a new horizontal grid. Each operator uses a different remapping method. The interpolation is based on an adapted SCRIP library version. For a detailed description of the remapping methods see [\[SCRIP\]](#).

### Operators

<b>remapbil</b>	Bilinear interpolation Performs a bilinear interpolation on all input fields. This interpolation method only works on quadrilateral curvilinear grids.
<b>remapbic</b>	Bicubic interpolation Performs a bicubic interpolation on all input fields. This interpolation method only works on quadrilateral curvilinear grids.
<b>remapdis</b>	Distance-weighted average remapping Performs a distance-weighted average remapping of the four nearest neighbor values on all input fields.
<b>remapnn</b>	Nearest neighbor remapping Performs a nearest neighbor remapping on all input fields.
<b>remapcon</b>	First order conservative remapping Performs a first order conservative remapping on all input fields.
<b>remapcon2</b>	Second order conservative remapping Performs a second order conservative remapping on all input fields.
<b>remaplaf</b>	Largest area fraction remapping Performs a largest area fraction remapping on all input fields.

### Parameter

*grid*     STRING     Target grid description file or name

### Environment

NORMALIZE_OPT	This variable is used to choose the normalization of the conservative remapping. By default NORMALIZE_OPT is set to 'fracarea' and will include the destination area fraction in the output weights; other options are 'none' and 'destarea' (for more information see <a href="#">[SCRIP]</a> ).
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.

### Note

For this module the author has converted the original Fortran 90 SCRIP software to ANSI C99. If there are any problems send a bug report to CDO and not to SCRIP!

**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. GENWEIGHTS - Generate SCRIP grid interpolation weights

### Synopsis

```
<operator> .grid ifile ofile
```

### Description

Interpolation between different horizontal grids can be a very time-consuming process. Especially if the data are on an unstructured or a large grid. In this case the [SCRIP](#) 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 method should be used only if all input fields are on the same grid and a possibly mask (missing values) does not change. This module contains operators to generate SCRIP interpolation weights of the first input field. Each operator is using a different interpolation method.

### Operators

<b>genbil</b>	Generate bilinear interpolation weights Generates bilinear interpolation weights and writes the result to a file. This interpolation method only works on quadrilateral curvilinear grids.
<b>genbic</b>	Generate bicubic interpolation weights Generates bicubic interpolation weights and writes the result to a file. This interpolation method only works on quadrilateral curvilinear grids.
<b>gendis</b>	Generate distance-weighted average remap weights Generates distance-weighted average remapping weights of the four nearest neighbor values and writes the result to a file.
<b>gennn</b>	Generate nearest neighbor remap weights Generates nearest neighbor remapping weights and writes the result to a file.
<b>gencon</b>	Generate 1st order conservative remap weights Generates first order conservative remapping weights and writes the result to a file.
<b>gencon2</b>	Generate 2nd order conservative remap weights Generates second order conservative remapping weights and writes the result to a file.
<b>genlaf</b>	Generate largest area fraction remap weights Generates largest area fraction remapping weights and writes the result to a file.

### Parameter

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

### Environment

<b>NORMALIZE_OPT</b>	This variable is used to choose the normalization of the conservative interpolation. By default NORMALIZE_OPT is set to 'fracarea' and will include the destination area fraction in the output weights; other options are 'none' and 'destarea' (for more information see <a href="#">[SCRIP]</a> ).
<b>REMAP_EXTRAPOLATE</b>	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.

## Note

For this module the author has converted the original Fortran 90 SCRIP software to ANSI C99. If there are any problems send a bug report to CDO and not to SCRIP!

## 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
```



### 2.12.3. REMAP - SCRIP grid remapping

#### Synopsis

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

#### Description

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 module [GENWEIGHTS](#) 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

NORMALIZE_OPT	This variable is used to choose the normalization of the conservative interpolation. By default NORMALIZE_OPT is set to 'fracarea' and will include the destination area fraction in the output weights; other options are 'none' and 'destarea' (for more information see <a href="#">[SCRIP]</a> ).
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.

#### Note

For this module the author has converted the original Fortran 90 SCRIP software to ANSI C99. If there are any problems send a bug report to CDO and not to SCRIP!

#### 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
```

## 2.12.4. 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 [\[INTERA\]](#). All input fields have to be on the same horizontal grid.

### Parameter

<i>vct</i>	STRING	File name of an ASCII dataset with the vertical coordinate table
<i>oro</i>	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

## 2.12.5. INTVERT - Vertical interpolation

### Synopsis

```
ml2pl,plevels ifile ofile
```

```
ml2hl,hlevels ifile ofile
```

### Description

Interpolate 3D variables on hybrid model levels to pressure or height levels. The input file should contain the log. surface pressure or the surface pressure. To interpolate the temperature, the orography (surface geopotential) is also needed. The pressure, temperature, and orography are identified by their code numbers. 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. All input fields have to be on the same horizontal grid.

### Operators

- ml2pl**      Model to pressure level interpolation  
Interpolates 3D variables on hybrid model levels to pressure levels.
- ml2hl**      Model to height level interpolation  
Interpolates 3D variables on hybrid model levels to height levels. The procedure is the same as for the operator [mh2pl](#) except for the pressure levels being calculated from the heights by:  $plevel = 101325 * \exp(hlevel / -7000)$

### 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.

### Note

The netCDF CF convention for vertical hybrid coordinates is not supported, yet!

### 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
```

## 2.12.6. INTLEVEL - Linear level interpolation

### Synopsis

```
intlevel,levels ifile ofile
```

### Description

This operator performs a linear vertical interpolation of non hybrid 3D variables.

### Parameter

<i>levels</i>	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
```

## 2.12.7. 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 vert. 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
```

## 2.12.8. INTTIME - Time interpolation

### Synopsis

```
inttime,date,time[,inc] ifile ofile
intntime,n ifile ofile
```

### Description

This module performs linear interpolation between time steps.

### Operators

<b>inttime</b>	Interpolation between time steps This operator creates a new dataset by linear interpolation between time steps. The user has to define the start date/time with an optional increment.
<b>intntime</b>	Interpolation between time steps This operator performs linear interpolation between time steps. The user has to define the number of time steps from one time step 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 time steps from one time step 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
```

## 2.12.9. INTYEAR - Year interpolation

### Synopsis

```
intyear,years ifile1 ifile2 obase
```

### Description

This operator performs linear interpolation between two years, time step by time step. 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>	This environment variable can be used to 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 time steps 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>dv2uwl</b>	Divergence and vorticity to U and V wind (linear)
<b>uv2dv</b>	U and V wind to divergence and vorticity
<b>uv2dwl</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

```

sp2gp ifile ofile

sp2gpl ifile ofile

gp2sp ifile ofile

gp2spl ifile ofile

sp2sp, trunc ifile ofile

```

### Description

This module transforms fields on Gaussian grids to spectral coefficients and vice versa.

### Operators

<b>sp2gp</b>	<p>Spectral to gridpoint</p> <p>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:</p> $nlat = NINT((trunc * \boxed{3} + 1.) / 2.)$
<b>sp2gpl</b>	<p>Spectral to gridpoint (linear)</p> <p>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:</p> $nlat = NINT((trunc * \boxed{2} + 1.) / 2.)$ <p>Use this operator to convert ERA40 data e.g. from TL159 to N80.</p>
<b>gp2sp</b>	<p>Gridpoint to spectral</p> <p>Convert all Gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:</p> $trunc = (nlat * 2 - 1) / \boxed{3}$
<b>gp2spl</b>	<p>Gridpoint to spectral (linear)</p> <p>Convert all Gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:</p> $trunc = (nlat * 2 - 1) / \boxed{2}$ <p>Use this operator to convert ERA40 data e.g. from N80 to TL159 instead of T106.</p>
<b>sp2sp</b>	<p>Spectral to spectral</p> <p>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.</p>

### Parameter

<i>trunc</i>	INTEGER	New spectral resolution
<i>wnums</i>	INTEGER	Comma separated list of wave numbers

**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.

### 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:  
$$nlat = NINT((trunc * \boxed{3} + 1.) / 2.)$$
- dv2uyl** 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:  
$$nlat = NINT((trunc * \boxed{2} + 1.) / 2.)$$
- 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 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:  
$$trunc = (nlat * 2 - 1) / \boxed{3}$$
- uv2dyl** 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 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:  
$$trunc = (nlat * 2 - 1) / \boxed{2}$$
- 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 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

### 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, FILEHEADER, 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 output.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 <code>ofile</code> . The numbers read are exactly that ones which are written out by <a href="#">output</a> .
<b>inputsrv</b>	SERVICE ASCII input Reads fields with ASCII numbers from standard input and stores them in <code>ofile</code> . 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 <a href="#">outputsrv</a> .
<b>inputtext</b>	EXTRA ASCII input Read fields with ASCII numbers from standard input and stores them in <code>ofile</code> . Each field should have header of 4 integers (EXTRA likely). The numbers read are exactly that ones which are written out by <a href="#">outputtext</a> .

### 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> .
<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 (nelem max = gridsizes)

## 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

## 2.15. 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:

<a href="#">gradsdes1</a>	GrADS data descriptor file (version 1 GRIB map)
<a href="#">gradsdes2</a>	GrADS data descriptor file (version 2 GRIB map)
<a href="#">bandpass</a>	Bandpass filtering
<a href="#">lowpass</a>	Lowpass filtering
<a href="#">highpass</a>	Highpass filtering
<a href="#">gridarea</a>	Grid cell area
<a href="#">gridweights</a>	Grid cell weights
<a href="#">smooth9</a>	9 point smoothing
<a href="#">setvals</a>	Set list of old values to new values
<a href="#">setrtoc</a>	Set range to constant
<a href="#">setrtoc2</a>	Set range to constant others to constant2
<a href="#">timsort</a>	Sort over the time
<a href="#">const</a>	Create a constant field
<a href="#">random</a>	Create a field with random numbers
<a href="#">stdatm</a>	Create values for pressure and temperature for hydrostatic atmosphere
<a href="#">rotuvb</a>	Backward rotation
<a href="#">mastrfu</a>	Mass stream function
<a href="#">histcount</a>	Histogram count
<a href="#">histsum</a>	Histogram sum
<a href="#">histmean</a>	Histogram mean
<a href="#">histfreq</a>	Histogram frequency
<a href="#">sethalo</a>	Set the left and right bounds of a field
<a href="#">wct</a>	Windchill temperature
<a href="#">fdns</a>	Frost days where no snow index per time period
<a href="#">strwin</a>	Strong wind days index per time period
<a href="#">strbre</a>	Strong breeze days index per time period
<a href="#">strgal</a>	Strong gale days index per time period
<a href="#">hurr</a>	Hurricane days index per time period

### 2.15.1. GRADSDES - GrADS data descriptor file

#### Synopsis

`<operator> ifile`

#### Description

Creates a [GrADS](#) data descriptor file. Supported file formats are GRIB1, 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. "gradsdes" is an alias for [gradsdes2](#).

#### Operators

- |                  |   |
|------------------|---|
| <b>gradsdes1</b> | GrADS data descriptor file (version 1 GRIB map)<br>Creates a GrADS data descriptor file. Generated a machine specific version 1 GrADS map file for GRIB1 datasets.  |
| <b>gradsdes2</b> | GrADS data descriptor file (version 2 GRIB map)<br>Creates a GrADS data descriptor file. Generated a machine independent version 2 GrADS map file for GRIB1 datasets. This map file can be used only with GrADS version 1.8 or newer. |

#### Example

To create a GrADS data descriptor file from a GRIB1 dataset use:

```
cdo gradsdes2 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 time steps 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 12:00Z1jan1987 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. 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 fills it with zeros (zero- padding) up to the next time-step-number that is a power of 2. Then it (fast fourier) transforms the time series with  $2^n$  elements 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. This time series is cut to the original number of timesteps from `ifile` and written to `ofile`. 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 avoid effects of zero-padding cut or extend your time series down/up to the nearest power of two
- 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 time steps and `dT` the time increment of `ifile` in years.

### Operators

<b>bandpass</b>	Bandpass filtering Bandpass filtering (pass for frequencies between <code>fmin</code> and <code>fmax</code> ). Suppresses all variability outside the frequency range specified by [ <code>fmin,fmax</code> ].
<b>lowpass</b>	Lowpass filtering Lowpass filtering (pass for frequencies lower than <code>fmax</code> ). Suppresses all variability with frequencies greater than <code>fmax</code> .
<b>highpass</b>	Highpass filtering Highpass filtering (pass for frequencies greater than <code>fmin</code> ). Suppresses all variability with frequencies lower than <code>fmin</code> .

### Parameter

<code>fmin</code>	FLOAT	Minimum frequency per year that passes the filter.
<code>fmax</code>	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  $y \geq 0.5$ .

### 2.15.3. 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

<b>PLANET_RADIUS</b>	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.
----------------------	--

## 2.15.4. 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.

## 2.15.5. 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

**setvals** Set list of old values to new values  
Supply a list of n pairs of old and new values.

**setrtoc** Set range to constant

$$o(t,x) = \begin{cases} c & \text{if } i(t,x) \geq rmin \wedge i(t,x) \leq rmax \\ i(t,x) & \text{if } i(t,x) < rmin \vee i(t,x) > rmax \end{cases}$$

**setrtoc2** 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



### 2.15.6. TIMSORT - Timsort

#### Synopsis

```
timsort ifile ofile
```

#### Description

Sorts the elements in ascending order over all time steps 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 time steps use:

```
cdo timsort ifile ofile
```

### 2.15.7. VARGEN - Generate a field

#### Synopsis

```
const,const,grid ofile
```

```
random,grid[,seed] ofile
```

```
stdatm,levels ofile
```

#### Description

Generates a dataset with one or more fields. The field size is specified by the user given grid description. According to the chosen operator all field elements are constant or filled with random numbers.

## 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>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

$T_0 =$	213K	: offset to get a surface temperature of 288K
$\Delta T =$	75K	: Temperature lapse rate for 10Km
$P_0 =$	1013.25hPa	: surface pressure
$H =$	10000.0m	: scale height
$g =$	$9.80665 \frac{\text{m}}{\text{s}^2}$	: earth gravity
$R =$	$287.05 \frac{\text{J}}{\text{kgK}}$	: gas constant for air

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>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
```

## 2.15.8. ROTUV - 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
```

## 2.15.9. 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 number 272). The input dataset have to be a zonal mean of v-velocity (code number 132) on pressure levels.

### Example

To compute the mass stream function from a zonal mean v-velocity dataset use:

```
cdo mastrfu ifile ofile
```

## 2.15.10. 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)

## 2.15.11. 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

## 2.15.12. 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.

## 2.15.13. FDNS - Frost days where no snow index per time period

### Synopsis

```
fdns ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily minimum temperatures `TN` and `ifile2` be a corresponding series of daily surface snow amounts. Then counted is the number of days where  $TN < 0$  °C and the surface snow amount is less than 1 cm. The temperature `TN` have to be given in units of Kelvin. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`.

## 2.15.14. STRWIN - Strong wind days index per time period

### Synopsis

```
strwin[,v] ifile ofile
```

### Description

Let `ifile` be a time series of daily maximum horizontal wind speeds `VX`, then counted is the number of days where  $VX > v$ . 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 time step in `ofile` is the date of the last contributing time step in `ifile`.

### Parameter

<code>v</code>	FLOAT	Horizontal wind speed threshold (m/s, default $v = 10.5$ m/s)
----------------	-------	---

### 2.15.15. STRBRE - Strong breeze days index per time period

#### Synopsis

```
strbre ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum horizontal wind speeds  $VX$ , then counted is the number of days where  $VX$  is greater than or equal to 10.5 m/s. 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 time step in `ofile` is the date of the last contributing time step in `ifile`.

### 2.15.16. STRGAL - Strong gale days index per time period

#### Synopsis

```
strgal ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum horizontal wind speeds  $VX$ , then counted is the number of days where  $VX$  is greater than or equal to 20.5 m/s. 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 time step in `ofile` is the date of the last contributing time step in `ifile`.

### 2.15.17. HURR - Hurricane days index per time period

#### Synopsis

```
hurr ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum horizontal wind speeds  $VX$ , then counted is the number of days where  $VX$  is greater than or equal to 32.5 m/s. 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 time step in `ofile` is the date of the last contributing time step in `ifile`.

## 2.16. Climate indices

This section contains modules to compute the climate indices of daily temperature and precipitation extremes.

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

<code>eca_cdd</code>	Consecutive dry days index per time period
<code>eca_cfd</code>	Consecutive frost days index per time period
<code>eca_csu</code>	Consecutive summer days index per time period
<code>eca_cwd</code>	Consecutive wet days index per time period
<code>eca_cwdi</code>	Cold wave duration index wrt mean of reference period
<code>eca_cwfi</code>	Cold-spell days index wrt 10th percentile of reference period
<code>eca_etr</code>	Intra-period extreme temperature range
<code>eca_fd</code>	Frost days index per time period
<code>eca_gsl</code>	Growing season length index
<code>eca_hd</code>	Heating degree days per time period
<code>eca_hwdi</code>	Heat wave duration index wrt mean of reference period
<code>eca_hwfi</code>	Warm spell days index wrt 90th percentile of reference period
<code>eca_id</code>	Ice days index per time period
<code>eca_pd</code>	Precipitation days index per time period
<code>eca_r10mm</code>	Heavy precipitation days index per time period
<code>eca_r20mm</code>	Very heavy precipitation days index per time period
<code>eca_r75p</code>	Moderate wet days wrt 75th percentile of reference period
<code>eca_r75ptot</code>	Precipitation percent due to R75p days
<code>eca_r90p</code>	Wet days wrt 90th percentile of reference period
<code>eca_r90ptot</code>	Precipitation percent due to R90p days
<code>eca_r95p</code>	Very wet days wrt 95th percentile of reference period
<code>eca_r95ptot</code>	Precipitation percent due to R95p days
<code>eca_r99p</code>	Extremely wet days wrt 99th percentile of reference period
<code>eca_r99ptot</code>	Precipitation percent due to R99p days
<code>eca_rr1</code>	Wet days index per time period
<code>eca_rx1day</code>	Highest one day precipitation amount per time period
<code>eca_rx5day</code>	Highest five-day precipitation amount per time period

---

<b>eca_sdii</b>	Simple daily intensity index per time period
<b>eca_su</b>	Summer days index per time period
<b>eca_tg10p</b>	Cold days percent wrt 10th percentile of reference period
<b>eca_tg90p</b>	Warm days percent wrt 90th percentile of reference period
<b>eca_tn10p</b>	Cold nights percent wrt 10th percentile of reference period
<b>eca_tn90p</b>	Warm nights percent wrt 90th percentile of reference period
<b>eca_tr</b>	Tropical nights index per time period
<b>eca_tx10p</b>	Very cold days percent wrt 10th percentile of reference period
<b>eca_tx90p</b>	Very warm days percent wrt 90th percentile of reference period



### 2.16.1. ECACDD - Consecutive dry days index per time period

#### Synopsis

```
eca_cdd ifile ofile
```

#### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then counted is the largest number of consecutive days where `RR` is less than 1 mm. A further output variable is the number of dry periods of more than 5 days. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `consecutive_dry_days_index_per_time_period`
- `number_of_cdd_periods_with_more_than_5days_per_time_period`

#### Example

To get the largest number of consecutive dry days of a time series of daily precipitation amounts use:

```
cdo eca_cdd rrfile ofile
```

### 2.16.2. ECACFD - Consecutive frost days index per time period

#### Synopsis

```
eca_cfd ifile ofile
```

#### Description

Let `ifile` be a time series of daily minimum temperatures `TN`, then counted is the largest number of consecutive days where  $TN < 0$  °C. Note that `TN` have to be given in units of Kelvin. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `consecutive_frost_days_index_per_time_period`

#### Example

To get the largest number of consecutive frost days of a time series of daily minimum temperatures use:

```
cdo eca_cfd tnfile ofile
```

### 2.16.3. ECACSU - Consecutive summer days index per time period

#### Synopsis

```
eca_csu[,T] ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum temperatures `TX`, then counted is the largest number of consecutive days where `TX > T`. The number `T` is an optional parameter with default `T = 25 °C`. Note that `TN` have to be given in units of Kelvin, whereas `T` have to be given in degrees Celsius. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `consecutive_summer_days_index_per_time_period`

#### Parameter

<code>T</code>	<code>FLOAT</code>	Temperature threshold ( Celsius, default: <code>T = 25 Celsius</code> )
----------------	--------------------	---

#### Example

To get the largest number of consecutive summer days of a time series of daily minimum temperatures use:

```
cdo eca_csu txfile ofile
```

### 2.16.4. ECACWD - Consecutive wet days index per time period

#### Synopsis

```
eca_cwd ifile ofile
```

#### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then counted is the largest number of consecutive days where `RR` is at least 1 mm. A further output variable is the number of wet periods of more than 5 days. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `consecutive_wet_days_index_per_time_period`
- `number_of_cwd_periods_with_more_than_5days_per_time_period`

#### Example

To get the largest number of consecutive wet days of a time series of daily precipitation amounts use:

```
cdo eca_cwd rrfile ofile
```

### 2.16.5. ECACWDI - Cold wave duration index wrt mean of reference period

#### Synopsis

```
eca_cwdi[,nday[,T]] ifile1 ifile2 ofile
```

#### Description

Let *ifile1* be a time series of daily minimum temperatures *TN*, and let *ifile2* be the mean *TN*<sub>norm</sub> of daily minimum temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least *nday* consecutive days,  $TN < TN_{norm} - T$ . The numbers *nday* and *T* are optional parameters with default *nday* = 6 and *T* = 5 °C. A further output variable is the number of cold waves longer than or equal to *nday* days. *TN*<sub>norm</sub> is calculated as the mean of minimum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both *TN* and *TN*<sub>norm</sub> have to be given in the same units. The date information of a time step in *ofile* is the date of the last contributing time step in *ifile1*. The following variables are created:

- cold\_wave\_duration\_index\_wrt\_mean\_of\_reference\_period
- cold\_waves\_per\_time\_period

#### Parameter

<i>nday</i>	INTEGER	Number of consecutive days (default: <i>nday</i> = 6)
<i>T</i>	FLOAT	Temperature offset ( Celsius, default: <i>T</i> = 5 Celsius)

#### Example

To compute the cold wave duration index of a time series of daily minimum temperatures use:

```
cdo eca_cwdi tnfile tnnormfile ofile
```

### 2.16.6. ECACWFI - Cold-spell days index wrt 10th percentile of reference period

#### Synopsis

```
eca_cwfi[,nday] ifile1 ifile2 ofile
```

#### Description

Let *ifile1* be a time series of daily mean temperatures *TG*, and *ifile2* be the 10th percentile *TG*<sub>n10</sub> of daily mean temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least *nday* consecutive days,  $TG < TG_{n10}$ . The number *nday* is an optional parameter with default *nday* = 6. A further output variable is the number of cold-spell periods longer than or equal to *nday* days. *TG*<sub>n10</sub> is calculated as the 10th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both *TG* and *TG*<sub>n10</sub> have to be given in the same units. The date information of a time step in *ofile* is the date of the last contributing time step in *ifile1*. The following variables are created:

- cold\_spell\_days\_index\_wrt\_10th\_percentile\_of\_reference\_period
- cold\_spell\_periods\_per\_time\_period

## Parameter

*nday*     INTEGER     Number of consecutive days (default: *nday* = 6)

## Example

To compute the number of cold-spell days of a time series of daily mean temperatures use:

```
cdo eca_cwfi tgfile tgn10file ofile
```

### 2.16.7. ECAETR - Intra-period extreme temperature range

#### Synopsis

```
eca_etr ifile1 ifile2 ofile
```

#### Description

Let `ifile1` and `ifile2` be time series of maximum and minimum temperatures TX and TN, respectively. Then the extreme temperature range is the difference of the maximum of TX and the minimum of TN. Note that TX and TN have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time steps in `ifile1` and `ifile2`. The following variables are created:

- `intra-period-extreme-temperature-range`

#### Example

To get the intra-period extreme temperature range for two time series of maximum and minimum temperatures use:

```
cdo eca_etr txfile tnfile ofile
```

### 2.16.8. ECAFD - Frost days index per time period

#### Synopsis

```
eca_fd ifile ofile
```

#### Description

Let `ifile` be a time series of daily minimum temperatures TN, then counted is the number of days where  $TN < 0^{\circ}\text{C}$ . Note that TN have to be given in units of Kelvin. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `frost-days-index-per-time-period`

#### Example

To get the number of frost days of a time series of daily minimum temperatures use:

```
cdo eca_fd tnfile ofile
```

## 2.16.9. ECAGSL - Thermal Growing season length index

### Synopsis

```
eca_gsl[,nday[,T[,fland]]] ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily mean temperatures TG, and `ifile2` be a land-water mask. Within a period of 12 months, the thermal growing season length is officially defined as the number of days between:

- first occurrence of at least `nday` consecutive days with  $TG > T$
- first occurrence of at least `nday` consecutive days with  $TG < T$  within the last 6 months

On northern hemisphere, this period corresponds with the regular year, whereas on southern hemisphere, it starts at July 1<sup>st</sup>. Please note, that this definition may lead to weird results concerning values  $TG = T$ : In the first half of the period, these days do not contribute to the gsl, but they do within the second half. Moreover this definition could lead to discontinuous values in equatorial regions.

The numbers `nday` and `T` are optional parameter with default `nday = 6` and `T = 5 °C`. The number `fland` is an optional parameter with default value `fland = 0.5` and denotes the fraction of a grid point that have to be covered by land in order to be included in the calculation. A further output variable is the start day of year of the growing season. Note that TG have to be given in units of Kelvin, whereas T have to be given in degrees Celsius.

The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `thermal_growing_season_length`
- `day_of_year_of_growing_season_start`

### Parameter

<code>nday</code>	INTEGER	Number of consecutive days (default: <code>nday = 6</code> )
<code>T</code>	FLOAT	Temperature threshold (degree Celsius, default: <code>T = 5 Celsius</code> )
<code>fland</code>	FLOAT	Land fraction threshold (default: <code>fland = 0.5</code> )

### Example

To get the growing season length of a time series of daily mean temperatures use:

```
cdo eca_gsl tgfile maskfile ofile
```

## 2.16.10. ECAHD - Heating degree days per time period

### Synopsis

```
eca_hd[,T1[,T2]] ifile ofile
```

### Description

Let `ifile` be a time series of daily mean temperatures `TG`, then the heating degree days are defined as the sum of `T1 - TG`, where only values `TG < T2` are considered. If `T1` and `T2` are omitted, a temperature of 17 °C is used for both parameters. If only `T1` is given, `T2` is set to `T1`. Note that `TG` have to be given in units of kelvin, whereas `T1` and `T2` have to be given in degrees Celsius. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- heating\_degree\_days\_per\_time\_period

### Parameter

<code>T1</code>	FLOAT	Temperature limit ( Celsius, default: <code>T1 = 17 Celsius</code> )
<code>T2</code>	FLOAT	Temperature limit ( Celsius, default: <code>T2 = T1</code> )

### Example

To compute the heating degree days of a time series of daily mean temperatures use:

```
cdo eca_hd tgfile ofile
```

## 2.16.11. ECAHWDI - Heat wave duration index wrt mean of reference period

### Synopsis

```
eca_hwdi[,nday[,T]] ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily maximum temperatures `TX`, and let `ifile2` be the mean `TXnorm` of daily maximum temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least `nday` consecutive days, `TX > TXnorm + T`. The numbers `nday` and `T` are optional parameters with default `nday = 6` and `T = 5 °C`. A further output variable is the number of heat waves longer than or equal to `nday` days. `TXnorm` is calculated as the mean of maximum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both `TX` and `TXnorm` have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- heat\_wave\_duration\_index\_wrt\_mean\_of\_reference\_period
- heat\_waves\_per\_time\_period

**Parameter**

<i>nday</i>	INTEGER	Number of consecutive days (default: $nday = 6$ )
<i>T</i>	FLOAT	Temperature offset ( Celsius, default: $T = 5$ Celsius)



## 2.16.12. ECAHWFI - Warm spell days index wrt 90th percentile of reference period

### Synopsis

```
eca_hwfi[,nday] ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily mean temperatures `TG`, and `ifile2` be the 90th percentile `TGn90` of daily mean temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least `nday` consecutive days,  $TG > TGn90$ . The number `nday` is an optional parameter with default `nday = 6`. A further output variable is the number of warm-spell periods longer than or equal to `nday` days. `TGn90` is calculated as the 90th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both `TG` and `TGn90` have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `warm_spell_days_index_wrt_90th_percentile_of_reference_period`
- `warm_spell_periods_per_time_period`

### Parameter

`nday`     `INTEGER`     Number of consecutive days (default: `nday = 6`)

### Example

To compute the number of warm-spell days of a time series of daily mean temperatures use:

```
cdo eca_hwfi tgfile tgn90file ofile
```

## 2.16.13. ECAID - Ice days index per time period

### Synopsis

```
eca_id ifile ofile
```

### Description

Let `ifile` be a time series of daily maximum temperatures `TX`, then counted is the number of days where  $TX < 0$  °C. Note that `TX` have to be given in units of Kelvin. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `ice_days_index_per_time_period`

### Example

To get the number of ice days of a time series of daily maximum temperatures use:

```
cdo eca_id txfile ofile
```

## 2.16.14. ECAPD - Precipitation days index per time period

### Synopsis

```
eca_pd,x ifile ofile
eca_r10mm ifile ofile
eca_r20mm ifile ofile
```

### Description

Let `ifile` be a time series of daily precipitation amounts `RR` in [mm] (or alternatively in [kg m<sup>-2</sup>]), then counted is the number of days where `RR` is at least  $x$  mm. `eca_r10mm` and `eca_r20mm` are specific ECA operators with a daily precipitation amount of 10 and 20 mm respectively. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `precipitation_days_index_per_time_period`

### Operators

<code>eca_pd</code>	Precipitation days index per time period Generic ECA operator with daily precipitation sum exceeding $x$ mm.
<code>eca_r10mm</code>	Heavy precipitation days index per time period Specific ECA operator with daily precipitation sum exceeding 10 mm.
<code>eca_r20mm</code>	Very heavy precipitation days index per time period Specific ECA operator with daily precipitation sum exceeding 20 mm.

### Parameter

<code>x</code>	FLOAT	Daily precipitation amount threshold in [mm]
----------------	-------	--

### Note

Please note, that precipitation rates in [mm/s] have to be converted to precipitation amounts (multiply with 86400 s).

### Example

To get the number of days with precipitation greater than 25 mm for a time series of daily precipitation amounts use:

```
cdo eca_pd,25 ifile ofile
```

### 2.16.15. ECAR75P - Moderate wet days wrt 75th percentile of reference period

#### Synopsis

```
eca_r75p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 75th percentile `RRn75` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the percentage of wet days with  $RR > RRn75$ . `RRn75` is calculated as the 75th percentile of all wet days of a given climate reference period. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `moderate_wet_days_wrt_75th_percentile_of_reference_period`

#### Example

To compute the percentage of wet days where the daily precipitation amount is greater than the 75th percentile of the daily precipitation amount at wet days for a given reference period use:

```
cdo eca_r75p rrfile rrn75file ofile
```

### 2.16.16. ECAR75PTOT - Precipitation percent due to R75p days

#### Synopsis

```
eca_r75ptot ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 75th percentile `RRn75` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the ratio of the precipitation sum at wet days with  $RR > RRn75$  to the total precipitation sum. `RRn75` is calculated as the 75th percentile of all wet days of a given climate reference period. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `precipitation_percent_due_to_R75p_days`

## 2.16.17. ECAR90P - Wet days wrt 90th percentile of reference period

### Synopsis

```
eca_r90p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 90th percentile `RRn90` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the percentage of wet days with  $RR > RRn90$ . `RRn90` is calculated as the 90th percentile of all wet days of a given climate reference period. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `wet_days_wrt_90th_percentile_of_reference_period`

### Example

To compute the percentage of wet days where the daily precipitation amount is greater than the 90th percentile of the daily precipitation amount at wet days for a given reference period use:

```
cdo eca_r90p rrfile rrn90file ofile
```

## 2.16.18. ECAR90PTOT - Precipitation percent due to R90p days

### Synopsis

```
eca_r90ptot ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 90th percentile `RRn90` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the ratio of the precipitation sum at wet days with  $RR > RRn90$  to the total precipitation sum. `RRn90` is calculated as the 90th percentile of all wet days of a given climate reference period. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `precipitation_percent_due_to_R90p_days`

## 2.16.19. ECAR95P - Very wet days wrt 95th percentile of reference period

### Synopsis

```
eca_r95p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 95th percentile `RRn95` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the percentage of wet days with  $RR > RRn95$ . `RRn95` is calculated as the 95th percentile of all wet days of a given climate reference period. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `very_wet_days_wrt_95th_percentile_of_reference_period`

### Example

To compute the percentage of wet days where the daily precipitation amount is greater than the 95th percentile of the daily precipitation amount at wet days for a given reference period use:

```
cdo eca_r95p rrfile rrn95file ofile
```

## 2.16.20. ECAR95PTOT - Precipitation percent due to R95p days

### Synopsis

```
eca_r95ptot ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 95th percentile `RRn95` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the ratio of the precipitation sum at wet days with  $RR > RRn95$  to the total precipitation sum. `RRn95` is calculated as the 95th percentile of all wet days of a given climate reference period. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `precipitation_percent_due_to_R95p_days`

## 2.16.21. ECAR99P - Extremely wet days wrt 99th percentile of reference period

### Synopsis

```
eca_r99p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 99th percentile `RRn99` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the percentage of wet days with  $RR > RRn99$ . `RRn99` is calculated as the 99th percentile of all wet days of a given climate reference period. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `extremely_wet_days_wrt_99th_percentile_of_reference_period`

### Example

To compute the percentage of wet days where the daily precipitation amount is greater than the 99th percentile of the daily precipitation amount at wet days for a given reference period use:

```
cdo eca_r99p rrfile rrn99file ofile
```

## 2.16.22. ECAR99PTOT - Precipitation percent due to R99p days

### Synopsis

```
eca_r99ptot ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 99th percentile `RRn99` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the ratio of the precipitation sum at wet days with  $RR > RRn99$  to the total precipitation sum. `RRn99` is calculated as the 99th percentile of all wet days of a given climate reference period. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `precipitation_percent_due_to_R99p_days`

### 2.16.23. ECARR1 - Wet days index per time period

#### Synopsis

```
eca_rr1 ifile ofile
```

#### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then counted is the number of days where `RR` is at least 1 mm. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `wet_days_index_per_time_period`

#### Example

To get the number of wet days of a time series of daily precipitation amounts use:

```
cdo eca_rr1 rrfile ofile
```

### 2.16.24. ECARX1DAY - Highest one day precipitation amount per time period

#### Synopsis

```
eca_rx1day[,mode] ifile ofile
```

#### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then the maximum of `RR` is written to `ofile`. If the optional parameter `mode` is set to 'm', then maximum daily precipitation amounts are determined for each month. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `highest_one_day_precipitation_amount_per_time_period`

#### Parameter

`mode`      **STRING**      Operation mode (optional). If `mode` = 'm' then maximum daily precipitation amounts are determined for each month

#### Example

To get the maximum of a time series of daily precipitation amounts use:

```
cdo eca_rx1day rrfile ofile
```

If you are interested in the maximum daily precipitation for each month, use:

```
cdo eca_rx1day,m rrfile ofile
```

Apart from metadata information, both operations yield the same as:

```
cdo timmax rrfile ofile
cdo monmax rrfile ofile
```

## 2.16.25. ECARX5DAY - Highest five-day precipitation amount per time period

### Synopsis

```
eca_rx5day[,x] ifile ofile
```

### Description

Let `ifile` be a time series of 5-day precipitation totals `RR`, then the maximum of `RR` is written to `ofile`. A further output variable is the number of 5 day period with precipitation totals greater than `x` mm, where `x` is an optional parameter with default `x = 50` mm. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `highest_five_day_precipitation_amount_per_time_period`
- `number_of_5day_heavy_precipitation_periods_per_time_period`

### Parameter

<code>x</code>	FLOAT	Precipitation threshold (mm, default: <code>x = 50</code> mm)
----------------	-------	---

### Example

To get the maximum of a time series of 5-day precipitation totals use:

```
cdo eca_rx5day rrfile ofile
```

Apart from metadata information, the above operation yields the same as:

```
cdo timmax rrfile ofile
```

## 2.16.26. ECASDII - Simple daily intensity index per time period

### Synopsis

```
eca_sdii ifile ofile
```

### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then the mean precipitation amount at wet days ( $RR > 1$  mm) is written to `ofile`. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `simple_daily_intensity_index_per_time_period`

### Example

To get the daily intensity index of a time series of daily precipitation amounts use:

```
cdo eca_sdii rrfile ofile
```



## 2.16.27. ECASU - Summer days index per time period

### Synopsis

```
eca_su[,T] ifile ofile
```

### Description

Let `ifile` be a time series of daily maximum temperatures `TX`, then counted is the number of days where `TX > T`. The number `T` is an optional parameter with default `T = 25 °C`. Note that `TX` have to be given in units of Kelvin, whereas `T` have to be given in degrees Celsius. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `summer_days_index_per_time_period`

### Parameter

<code>T</code>	<code>FLOAT</code>	Temperature threshold (degree Celsius, default: <code>T = 25 Celsius</code> )
----------------	--------------------	---

### Example

To get the number of summer days of a time series of daily maximum temperatures use:

```
cdo eca_su txfile ofile
```

## 2.16.28. ECATG10P - Cold days percent wrt 10th percentile of reference period

### Synopsis

```
eca_tg10p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily mean temperatures TG, and `ifile2` be the 10th percentile TGn10 of daily mean temperatures for any period used as reference. Then calculated is the percentage of time where  $TG < TGn10$ . TGn10 is calculated as the 10th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TG and TGn10 have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- cold\_days\_percent\_wrt\_10th\_percentile\_of\_reference\_period

### Example

To compute the percentage of time where the daily mean temperature is less than the 10th percentile of the daily mean temperature for a given reference period use:

```
cdo eca_tg10p tgfile tgn10file ofile
```

## 2.16.29. ECATG90P - Warm days percent wrt 90th percentile of reference period

### Synopsis

```
eca_tg90p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily mean temperatures TG, and `ifile2` be the 90th percentile TGn90 of daily mean temperatures for any period used as reference. Then calculated is the percentage of time where  $TG > TGn90$ . TGn90 is calculated as the 90th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TG and TGn90 have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- warm\_days\_percent\_wrt\_90th\_percentile\_of\_reference\_period

### Example

To compute the percentage of time where the daily mean temperature is greater than the 90th percentile of the daily mean temperature for a given reference period use:

```
cdo eca_tg90p tgfile tgn90file ofile
```

### 2.16.30. ECATN10P - Cold nights percent wrt 10th percentile of reference period

#### Synopsis

```
eca_tn10p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily minimum temperatures `TN`, and `ifile2` be the 10th percentile `TNn10` of daily minimum temperatures for any period used as reference. Then calculated is the percentage of time where  $TN < TN_{n10}$ . `TNn10` is calculated as the 10th percentile of daily minimum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both `TN` and `TNn10` have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `cold_nights_percent_wrt_10th_percentile_of_reference_period`

#### Example

To compute the percentage of time where the daily minimum temperature is less than the 10th percentile of the daily minimum temperature for a given reference period use:

```
cdo eca_tn10p tnfile tnn10file ofile
```

### 2.16.31. ECATN90P - Warm nights percent wrt 90th percentile of reference period

#### Synopsis

```
eca_tn90p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily minimum temperatures `TN`, and `ifile2` be the 90th percentile `TNn90` of daily minimum temperatures for any period used as reference. Then calculated is the percentage of time where  $TN > TN_{n90}$ . `TNn90` is calculated as the 90th percentile of daily minimum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both `TN` and `TNn90` have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `warm_nights_percent_wrt_90th_percentile_of_reference_period`

#### Example

To compute the percentage of time where the daily minimum temperature is greater than the 90th percentile of the daily minimum temperature for a given reference period use:

```
cdo eca_tn90p tnfile tnn90file ofile
```

## 2.16.32. ECATR - Tropical nights index per time period

### Synopsis

```
eca_tr[,T] ifile ofile
```

### Description

Let `ifile` be a time series of daily minimum temperatures `TN`, then counted is the number of days where `TN > T`. The number `T` is an optional parameter with default `T = 20 °C`. Note that `TN` have to be given in units of Kelvin, whereas `T` have to be given in degrees Celsius. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile`. The following variables are created:

- `tropical_nights_index_per_time_period`

### Parameter

`T`     `FLOAT`     Temperature threshold ( Celsius, default: `T = 20 Celsius`)

### Example

To get the number of tropical nights of a time series of daily minimum temperatures use:

```
cdo eca_tr tnfile ofile
```

## 2.16.33. ECATX10P - Very cold days percent wrt 10th percentile of reference period

### Synopsis

```
eca_tx10p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily maximum temperatures `TX`, and `ifile2` be the 10th percentile `TXn10` of daily maximum temperatures for any period used as reference. Then calculated is the percentage of time where `TX < TXn10`. `TXn10` is calculated as the 10th percentile of daily maximum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both `TX` and `TXn10` have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `very_cold_days_percent_wrt_10th_percentile_of_reference_period`

### Example

To compute the percentage of time where the daily maximum temperature is less than the 10th percentile of the daily maximum temperature for a given reference period use:

```
cdo eca_tx10p txfile txn10file ofile
```

## 2.16.34. ECATX90P - Very warm days percent wrt 90th percentile of reference period

### Synopsis

```
eca_tx90p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily maximum temperatures TX, and `ifile2` be the 90th percentile TX<sub>n90</sub> of daily maximum temperatures for any period used as reference. Then calculated is the percentage of time where TX > TX<sub>n90</sub>. TX<sub>n90</sub> is calculated as the 90th percentile of daily maximum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TX and TX<sub>n90</sub> have to be given in the same units. The date information of a time step in `ofile` is the date of the last contributing time step in `ifile1`. The following variables are created:

- `very_warm_days_percent_wrt_90th_percentile_of_reference_period`

### Example

To compute the percentage of time where the daily maximum temperature is greater than the 90th percentile of the daily maximum temperature for a given reference period use:

```
cdo eca_tx90p txfile txn90file ofile
```

# 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)

# A. Grid description examples

## A.1. Example of a curvilinear grid description

Here is an example for the **CDO** description of a curvilinear grid. xvals/yvals describes the position 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	

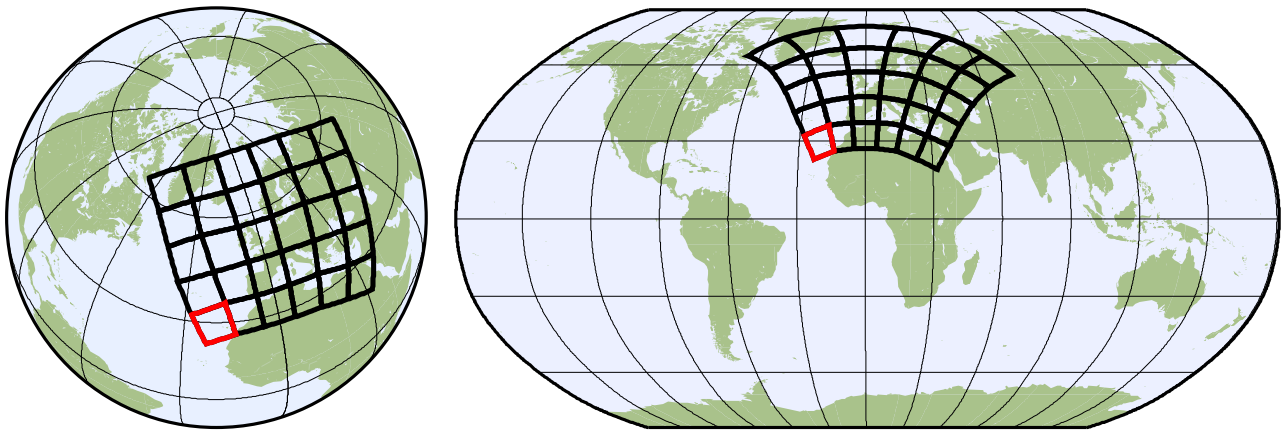


Figure A.1.: Orthographic and Robinson projection of the curvilinear grid, the first grid cell is colored red

## A.2. Example description for an unstructured grid

Here is an example of the **CDO** description for an unstructured grid. xvals/yvals describes the position 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

```

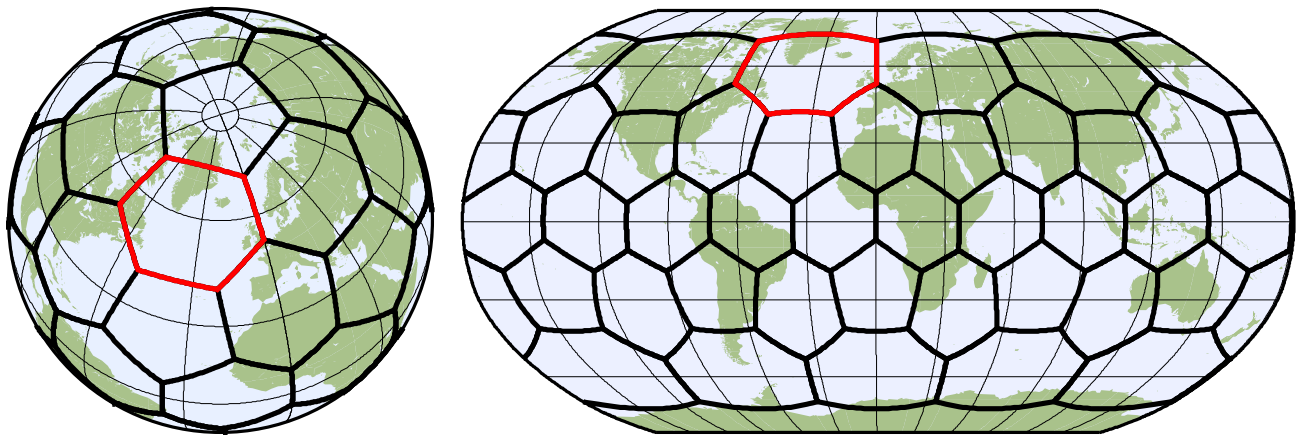


Figure A.2.: Orthographic and Robinson projection of the unstructured grid



# Operator index

## A

abs .....	63
acos .....	63
add .....	65
addc .....	64
asin .....	63
atan2 .....	65

## B

bandpass .....	150
----------------	-----

## C

cat .....	25
chcode .....	49
chlevel .....	49
chlevelc .....	49
chlevelv .....	49
chname .....	49
chparam .....	49
consecsum .....	75
consects .....	75
const .....	153
copy .....	25
cos .....	63

## D

dayavg .....	97
daymax .....	97
daymean .....	97
daymin .....	97
daypctl .....	98
daystd .....	97
daysum .....	97
dayvar .....	97
delcode .....	32
delname .....	32
delparam .....	32
detrend .....	118
diff .....	20
diffn .....	20
div .....	65
divc .....	64
divdpm .....	69
divdpy .....	69
dv2ps .....	140
dv2uv .....	140
dv2uvl .....	140

## E

eca_cdd .....	161
---------------	-----

eca_cfd .....	161
eca_csu .....	162
eca_cwd .....	162
eca_cwdi .....	163
eca_cwfi .....	163
eca_etr .....	165
eca_fd .....	165
eca_gsl .....	166
eca_hd .....	167
eca_hwdi .....	167
eca_hwfi .....	169
eca_id .....	169
eca_pd .....	170
eca_r10mm .....	170
eca_r20mm .....	170
eca_r75p .....	171
eca_r75ptot .....	171
eca_r90p .....	172
eca_r90ptot .....	172
eca_r95p .....	173
eca_r95ptot .....	173
eca_r99p .....	174
eca_r99ptot .....	174
eca_rrl .....	175
eca_rx1day .....	175
eca_rx5day .....	176
eca_sdii .....	176
eca_su .....	177
eca_tg10p .....	178
eca_tg90p .....	178
eca_tn10p .....	179
eca_tn90p .....	179
eca_tr .....	180
eca_tx10p .....	180
eca_tx90p .....	181
enlarge .....	57
ensavg .....	76
ensbrs .....	79
enscrps .....	79
ensmax .....	76
ensmean .....	76
ensmin .....	76
enspctl .....	76
ensrkhistspace .....	78
ensrkhisttime .....	78
ensroc .....	78
ensstd .....	76
enssum .....	76
ensvar .....	76

eof	121
eof3d	121
eofcoeff	123
eofspatial	121
eoftime	121
eq	42
eqc	43
exp	63
expr	61
exprf	61

**F**

fdns	157
fldavg	81
fldcor	116
fldmax	81
fldmean	81
fldmin	81
fldpctl	81
fldstd	81
fldsum	81
fldvar	81

**G**

ge	42
gec	43
genbic	127
genbil	127
gencon	127
gencon2	127
gendis	127
genlaf	127
gennn	127
gp2sp	138
gp2spl	138
gradsdes1	149
gradsdes2	149
gridarea	151
gridboxavg	87
gridboxmax	87
gridboxmean	87
gridboxmin	87
gridboxstd	87
gridboxsum	87
gridboxvar	87
griddes	23
gridweights	151
gt	42
gtc	43

**H**

highpass	150
histcount	156
histfreq	156
histmean	156
histsum	156
houravg	95
hourmax	95

hourmean	95
hourmin	95
hourpctl	96
hourstd	95
hoursum	95
hourvar	95
hurr	158

**I**

ifnotthen	39
ifnotthenc	40
ifthen	39
ifthenc	40
ifthenelse	39
import_amsr	144
import_binary	142
import_cmsaf	143
info	18
infor	18
input	145
inputext	145
inputsrv	145
int	63
intlevel	133
intlevel3d	134
intlevelx3d	134
intntime	135
inttime	135
intyear	136
invertlat	53
invertlev	53

**L**

le	42
lec	43
ln	63
log10	63
lowpass	150
lt	42
ltc	43

**M**

map	18
maskindexbox	55
masklonlatbox	55
maskregion	54
mastrfu	155
max	65
meravg	85
merge	27
mergetime	27
mermax	85
mermean	85
mermin	85
merpctl	85
merstd	85
mersum	85
mervar	85

min	65
ml2hl	132
ml2pl	132
monadd	66
monavg	99
monddiv	66
monmax	99
monmean	99
monmin	99
monmul	66
monpctl	100
monstd	99
monsub	66
monsum	99
monvar	99
mul	65
mulc	64
muldpm	69
muldpy	69

**N**

ndate	21
ne	42
nec	43
nint	63
nlevel	21
nmon	21
npar	21
ntime	21
nyear	21

**O**

output	146
outputext	146
outputf	146
outputint	146
outputsrv	146

**P**

pardes	23
pow	63

**R**

random	153
reci	63
regres	118
remap	129
remapbic	125
remapbil	125
remapcon	125
remapcon2	125
remapdis	125
remapeta	130
remaplaf	125
remapnn	125
replace	26
rotuvb	155
runavg	91

runmax	91
runmean	91
runmin	91
runpctl	92
runstd	91
runsum	91
runvar	91

**S**

seasavg	103
seasmax	103
seasmean	103
seasmin	103
seaspctl	104
seasstd	103
seassum	103
seasvar	103
selcode	32
seldate	35
selday	35
selgrid	32
selhour	35
selindexbox	37
sellevel	32
sellevidx	32
sellonlatbox	37
selltype	32
selmon	35
selname	32
selparam	32
selseas	35
selsmon	35
selstdname	32
seltabnum	32
seltime	35
selimestep	35
selyear	35
selzaxis	32
setcalendar	47
setcindexbox	56
setclonlatbox	56
setcode	46
setctomiss	58
setdate	47
setday	47
setgatt	52
setgatts	52
setgrid	50
setgridarea	50
setgridtype	50
sethalo	156
setlevel	46
setltype	46
setmisstoc	58
setmissval	58
setmon	47
setname	46
setparam	46

setpartab .....	46
setreftime .....	47
setrtoc .....	152
setrtoc2 .....	152
setrtomiss .....	58
settaxis .....	47
settime .....	47
settunits .....	47
setvals .....	152
setvrange .....	58
setyear .....	47
setzaxis .....	51
shifttime .....	47
showcode .....	22
showdate .....	22
showformat .....	22
showlevel .....	22
showltype .....	22
showmon .....	22
showname .....	22
showstdname .....	22
showtime .....	22
showtimestamp .....	22
showyear .....	22
sin .....	63
sinfo .....	19
sinfo .....	19
smooth9 .....	152
sp2gp .....	138
sp2gpl .....	138
sp2sp .....	138
splitcode .....	28
splitday .....	29
splitgrid .....	28
splithour .....	29
splitlevel .....	28
splitmon .....	29
splitname .....	28
splitparam .....	28
splitseas .....	29
splitsel .....	30
splittabnum .....	28
splityear .....	29
splitzaxis .....	28
sqr .....	63
sqrt .....	63
stdatm .....	153
strbre .....	158
strgal .....	158
strwin .....	157
sub .....	65
subc .....	64
subtrend .....	119

## T

tan .....	63
timavg .....	93
timcor .....	116

timmax .....	93
timmean .....	93
timmin .....	93
timpctl .....	94
timselavg .....	89
timselmax .....	89
timselmean .....	89
timselmin .....	89
timselpctl .....	90
timselstd .....	89
timselsum .....	89
timselvar .....	89
timsort .....	153
timstd .....	93
timsum .....	93
timvar .....	93
trend .....	119

## U

uv2dv .....	140
uv2dvl .....	140

## V

vct .....	23
vertavg .....	88
vertmax .....	88
vertmean .....	88
vertmin .....	88
vertstd .....	88
vertsum .....	88
vertvar .....	88

## W

wct .....	157
-----------	-----

## Y

ydayadd .....	68
ydayavg .....	106
ydaydiv .....	68
ydaymax .....	106
ydaymean .....	106
ydaymin .....	106
ydaymul .....	68
ydaypctl .....	107
ydaystd .....	106
ydaysub .....	68
ydaysum .....	106
ydayvar .....	106
ydrunavg .....	112
ydrunmax .....	112
ydrunmean .....	112
ydrunmin .....	112
ydrunpctl .....	114
ydrunstd .....	112
ydrunsum .....	112
ydrunvar .....	112
yearavg .....	101
yearmax .....	101

yearmean .....	101
yearmin .....	101
yearpctl .....	102
yearstd .....	101
yearsum .....	101
yearvar .....	101
yhouravg .....	105
yhourmax .....	105
yhourmean .....	105
yhourmin .....	105
yhourstd .....	105
yhoursum .....	105
yhourvar .....	105
ymonadd .....	67
ymonavg .....	108
ymondiv .....	67
ymonmax .....	108
ymonmean .....	108
ymonmin .....	108
ymonmul .....	67
ymonpctl .....	109
ymonstd .....	108
ymonsub .....	67
ymonsum .....	108
ymonvar .....	108
yseasavg .....	110
yseasmax .....	110
yseasmean .....	110
yseasmin .....	110
yseaspctl .....	111
yseasstd .....	110
yseassum .....	110
yseasvar .....	110

## Z

zaxisdes .....	23
zonavg .....	83
zonmax .....	83
zonmean .....	83
zonmin .....	83
zonpctl .....	83
zonstd .....	83
zonsum .....	83
zonvar .....	83