

CDO User's Guide

Climate Data Operators
Version 1.0.7
March 2007

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

Contents

1. Introduction	6
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. Grid description	9
1.3.1. Predefined grids	9
1.3.2. Grids from data files	10
1.3.3. SCRIP grids	10
1.3.4. PINGO grids	10
1.3.5. CDO grids	11
1.4. Time axis	12
1.4.1. Absolute time	12
1.4.2. Relative time	12
1.4.3. Conversion of the time	12
1.5. Parameter table	13
1.6. Missing values	13
1.6.1. Mean and average	14
2. Reference manual	15
2.1. Information	16
2.1.1. INFO - Information and simple statistics	17
2.1.2. SINFO - Short information	18
2.1.3. DIFF - Compare two datasets field by field	19
2.1.4. NINFO - Print the number of variables, levels or times	20
2.1.5. SHOWINFO - Show variables, levels or times	21
2.1.6. FILEDES - Dataset description	22
2.2. File operations	23
2.2.1. COPY - Copy datasets	24
2.2.2. REPLACE - Replace variables	24
2.2.3. MERGE - Merge datasets	25
2.2.4. SPLIT - Split a dataset	26
2.2.5. SPLITTIME - Split time steps of a dataset	27
2.3. Selection	28
2.3.1. SELECT - Select fields	29
2.3.2. SELTIME - Select time steps	31
2.3.3. SELBOX - Select a box of a field	33
2.4. Conditional selection	34
2.4.1. COND - Conditional select one field	35
2.4.2. COND2 - Conditional select two fields	35
2.4.3. CONDC - Conditional select a constant	36
2.5. Comparison	37
2.5.1. COMP - Comparison of two fields	38

2.5.2. COMPC - Comparison of a field with a constant	39
2.6. Modification	40
2.6.1. SET - Set field info	41
2.6.2. SETTIME - Set time	42
2.6.3. CHANGE - Change field header	44
2.6.4. SETGRID - Set grid type	45
2.6.5. SETZAXIS - Set zaxis type	45
2.6.6. SETGATT - Set global attribute	46
2.6.7. INVERT - Invert fields	47
2.6.8. MASKBOX - Mask a box	48
2.6.9. SETBOX - Set a box to constant	49
2.6.10. ENLARGE - Enlarge fields	50
2.6.11. SETMISS - Set missing value	51
2.7. Arithmetic	52
2.7.1. EXPR - Evaluate expressions	53
2.7.2. MATH - Mathematical functions	54
2.7.3. ARITHC - Arithmetic with a constant	55
2.7.4. ARITH - Arithmetic on two datasets	56
2.7.5. YMONARITH - Multi-year monthly arithmetic	57
2.7.6. ARITHDAYS - Arithmetic with days	58
2.8. Statistical values	59
2.8.1. ENSSTAT - Statistical values over an ensemble	63
2.8.2. FLDSTAT - Statistical values over a field	65
2.8.3. ZONSTAT - Zonal statistical values	67
2.8.4. MERSTAT - Meridional statistical values	69
2.8.5. VERTSTAT - Vertical statistical values	71
2.8.6. SELSTAT - Time range statistical values	72
2.8.7. SELPCTL - Time range percentile values	73
2.8.8. RUNSTAT - Running statistical values	74
2.8.9. RUNPCTL - Running percentile values	75
2.8.10. TIMSTAT - Statistical values over all time steps	76
2.8.11. TIMPCTL - Percentile values over all time steps	77
2.8.12. HOURSTAT - Hourly statistical values	78
2.8.13. HOURPCTL - Hourly percentile values	79
2.8.14. DAYSTAT - Daily statistical values	80
2.8.15. DAYPCTL - Daily percentile values	81
2.8.16. MONSTAT - Monthly statistical values	82
2.8.17. MONPCTL - Monthly percentile values	83
2.8.18. YEARSTAT - Yearly statistical values	84
2.8.19. YEARPCTL - Yearly percentile values	85
2.8.20. SEASSTAT - Seasonal statistical values	86
2.8.21. SEASPCTL - Seasonal percentile values	87
2.8.22. YDAYSTAT - Multi-year daily statistical values	88
2.8.23. YDAYPCTL - Multi-year daily percentile values	89
2.8.24. YMONSTAT - Multi-year monthly statistical values	90
2.8.25. YMONPCTL - Multi-year monthly percentile values	91
2.8.26. YSEASSTAT - Multi-year seasonal statistical values	92
2.8.27. YSEASPCTL - Multi-year seasonal percentile values	93
2.8.28. YDRUNSTAT - Multi-year daily running statistical values	94
2.8.29. YDRUNPCTL - Multi-year daily running percentile values	96
2.9. Regression	97
2.9.1. DETREND - Detrend time series	98
2.9.2. TREND - Trend of time series	99
2.9.3. SUBTREND - Subtract a trend	99
2.10. Interpolation	100
2.10.1. REMAPGRID - SCRIP grid interpolation	101
2.10.2. GENWEIGHTS - Generate SCRIP grid interpolation weights	102
2.10.3. REMAP - SCRIP grid remapping	103

2.10.4. INTGRID - Grid interpolation	104
2.10.5. INTVERT - Vertical interpolation	105
2.10.6. INTTIME - Time interpolation	106
2.10.7. INTYEAR - Year interpolation	107
2.11. Transformation	108
2.11.1. SPECTRAL - Spectral transformation	109
2.11.2. WIND - Wind transformation	110
2.12. Formatted I/O	111
2.12.1. INPUT - Formatted input	112
2.12.2. OUTPUT - Formatted output	113
2.13. Miscellaneous	114
2.13.1. GRADSDES - GrADS data descriptor file	115
2.13.2. TIMSORT - Timsort	116
2.13.3. VARGEN - Generate a field	117
2.13.4. VARDUP - Variable duplication	117
2.13.5. ROTUV - Rotation	118
2.13.6. MASTRFU - Mass stream function	118
2.13.7. HI - Humidity index (C)	119
2.13.8. WCT - Windchill temperature (C)	119
2.14. ECA indices	120
2.14.1. ECACDD - Consecutive dry days index per time period	122
2.14.2. ECACFD - Consecutive frost days index per time period	122
2.14.3. ECACSU - Consecutive summer days index per time period	123
2.14.4. ECACWD - Consecutive wet days index per time period	123
2.14.5. ECACWDI - Cold wave duration index wrt mean of reference period	124
2.14.6. ECACWFI - Cold-spell days index wrt 10th percentile of reference period	124
2.14.7. ECAETR - Intra-period extreme temperature range	125
2.14.8. ECAFD - Frost days index per time period	125
2.14.9. ECAFDNS - Frost days where no snow index per time period	126
2.14.10. ECAGSL - Growing season length index	126
2.14.11. ECAHD - Heating degree days per time period	127
2.14.12. ECAHWDI - Heat wave duration index wrt mean of reference period	127
2.14.13. ECAHWFI - Warm spell days index wrt 90th percentile of reference period	128
2.14.14. ECAID - Ice days index per time period	128
2.14.15. ECAR10MM - Heavy precipitation days index per time period	129
2.14.16. ECAR20MM - Very heavy precipitation days index per time period	129
2.14.17. ECAR75P - Moderate wet days wrt 75th percentile of reference period	130
2.14.18. ECAR75PTOT - Precipitation percent due to R75p days	130
2.14.19. ECAR90P - Wet days wrt 90th percentile of reference period	131
2.14.20. ECAR90PTOT - Precipitation percent due to R90p days	131
2.14.21. ECAR95P - Very wet days wrt 95th percentile of reference period	132
2.14.22. ECAR95PTOT - Precipitation percent due to R95p days	132
2.14.23. ECAR99P - Extremely wet days wrt 99th percentile of reference period	133
2.14.24. ECAR99PTOT - Precipitation percent due to R99p days	133
2.14.25. ECARR1 - Wet days index per time period	134
2.14.26. ECARX1DAY - Highest one day precipitation amount per time period	134
2.14.27. ECARX5DAY - Highest five-day precipitation amount per time period	135
2.14.28. ECASDII - Simple daily intensity index per time period	135
2.14.29. ECASTRWIN - Strong wind days index per time period	136
2.14.30. ECASTRBRE - Strong breeze days index per time period	136
2.14.31. ECASTRGAL - Strong gale days index per time period	136
2.14.32. ECAHURR - Hurricane days index per time period	137
2.14.33. ECASU - Summer days index per time period	137
2.14.34. ECATG10P - Cold days percent wrt 10th percentile of reference period	138
2.14.35. ECATG90P - Warm days percent wrt 90th percentile of reference period	138
2.14.36. ECATN10P - Cold nights percent wrt 10th percentile of reference period	139
2.14.37. ECATN90P - Warm nights percent wrt 90th percentile of reference period	139
2.14.38. ECATR - Tropical nights index per time period	140

2.14.39.ECATX10P - Very cold days percent wrt 10th percentile of reference period	140
2.14.40.ECATX90P - Very warm days percent wrt 90th percentile of reference period	141
A. Hints for PINGO user	143
B. Grid description examples	144
B.1. Example of a curvilinear grid description	144
B.2. Example description for unstructured grid cells	145
Operator index	146

1. Introduction

The Climate Data Operators (**CDO**) software is a collection of many operators for standard processing of climate and forecast 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](#) and [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 data formats SERVICE, EXTRA and IEG are also supported.

There are some limitations for GRIB and netCDF datasets. A GRIB dataset must be consistent, similar to netCDF. That means all time steps must have the same variables, and within a time step each variable may occur only once. NetCDF datasets are supported only with 2-dimensional, 3-dimensional and 4-dimensional variables and the attributes should follow the [GDT](#), [COARDS](#) or [CF Conventions](#).

The user interface and some operators are similar to the [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 300 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
- All 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 to compile the source code. The only requirement is a working ANSI C compiler.

First go to the [download](#) page (<http://www.mpimet.mpg.de/cdo>) to get the latest distribution, if you do not already have it.

To take full advantage of **CDO** features the following additional library should be installed.

- Unidata [netCDF](#) library (<http://www.unidata.ucar.edu/packages/netcdf/index.html>) version 3 or higher. This is needed to read/write netCDF files with **CDO**.

1.1.1. Compilation

Compilation is now done by performing the following steps:

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

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

2. Run the configure script:

```
./configure
```

Or with netCDF support:

```
./configure --with-netcdf=<netCDF 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] [Operators]
```

1.2.1. Options

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

- a Convert from a relative to an absolute time axis.
 -b <nbits> Set the number of bits for the output precision. The valid precisions depends on the file format:

<format>	<nbits>
grb	1 - 32
nc, nc2, srv, ext, ieg	32/64

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

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

File format	<format>
GRIB version 1	grb
netCDF	nc
netCDF version 2	nc2
SERVICE	srv
EXTRA	ext
IEG	ieg

- g <grid> Define the default grid description by name or from file.
 Available grid names are: **t<RES>grid**, **r<NX>x<NY>**, **gme<NI>**
 -h Help information for the operators.
 -m <missval> Set the default missing value (default: **-9e+33**).
 -R Convert GRIB data from reduced to regular grid.
 -r Convert from an absolute to a relative time axis.
 -t <partab> Set the default parameter table name or file.
 Predefined tables are: **echam4 echam5 mpiom1**
 -V Print the version number.
 -v Print extra details for some operators.

1.2.2. Operators

There are more than 300 operators available. A detailed description of all operators can be found in the [Reference Manual](#) section.

1.2.3. Combining operators

All operators with 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
```


1.2.4. Operator parameter

Some operators need one or more parameter.

- **STRING**

Unquoted characters without blanks and tabs. The following command select 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 list of integers can be specified by *first/last[/inc]*. To select the days 5, 6, 7, 8 and 9 use:

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

This is the same as:

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

1.3. 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: `interpolate`, `remapXXX` and `genXXX`)
- Generating variables (operator: `const`, `random`)

As now described, there are several possibilities to define a horizontal grid. Predefined grids are available for global regular, gaussian or icosahedral-hexagonal GME grids.

1.3.1. Predefined grids

The following pre-defined grid names are available: `r<NX>x<NY>`, `t<RES>grid` and `gme<NI>`

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

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

Global gaussian grid: `t<RES>grid`

`t<RES>grid` defines a global gaussian grid. Each valid triangular resolution can be used for `<RES>`. The longitudes starts 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 is the number of intervals on a main triangle side.

1.3.2. 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.

1.3.3. SCRIP grids

SCRIP is a Spherical Coordinate Remapping and Interpolation Package. It is using a common grid description in netCDF. You can use it to describe curvilinear grids or unstructured grid cells. For more information about this format see [[SCRIP](#)]. This grid description format is only available if the program was compiled with netCDF support.

SCRIP grid description example of a curvilinear MPIOM1 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" ;
}
```

1.3.4. PINGO grids

PINGO uses a very simple grid description in ASCII format to describe regular longitude/latitude or global gaussian grids. All PINGO grid description files are supported by **CDO**. For more information about this format see [[PINGO](#)].

PINGO grid description example of a T21 gaussian grid:

```
Grid Description File
(Comments start at non digit characters and end at end of line)
First part: The dimensions.
64 32 = Number of longitudes and latitudes
Second part: The listed longitudes.
2 means equidistant longitudes
```

```

0.000000 5.625000 = Most western and second most western longitude
Third part: The listed latitudes.
32 means all 32 latitudes are given in the following list:
 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

```

1.3.5. CDO grids

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

gridtype	STRING	type of the grid (gaussian, lonlat, curvilinear, cell)
gridsize	INTEGER	size of the grid
xsize	INTEGER	size in x direction (number of longitudes)
ysize	INTEGER	size in y direction (number of latitudes)
xvals	FLOAT ARRAY	x values of the grid
yvals	FLOAT ARRAY	y values of the grid
xnpole	FLOAT	x value of the north pole (rotated grid)
ynpole	FLOAT	y value of the north pole (rotated grid)
nvertex	INTEGER	number of the vertices for all grid cells
xbounds	FLOAT ARRAY	x bounds of each gridbox
ybounds	FLOAT ARRAY	y bounds of each gridbox
xfirst, xinc	FLOAT, FLOAT	macros to define xvals with a constant increment
yfirst, yinc	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 array size for the different grid types.

gridtype	lonlat	gaussian	curvilinear	cell
gridsize	xsize*ysize	xsize*ysize	xsize*ysize	ncell
xsize	nlon	nlon	nlon	gridsize
ysize	nlat	nlat	nlat	gridsize
xvals	xsize	xsize	gridsize	gridsize
yvals	ysize	ysize	gridsize	gridsize
xnpole	0			
ynpole	90			
nvertex	2	2	4	nv
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 the area weights are not needed.

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 an 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. Time axis

A time axis describes the time for every timestep. Two time types are available: absolute time and relative time. **CDO** tries to maintain the actual type of the time axis for all operators. The operators for time range statistic (e.g.: monavg, ymonavg, ...) create an absolute time axis.

1.4.1. 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 relative time axis is represented by the unit of the time: "day as %Y%m%d.%f".

1.4.2. 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, 360 days, 365 days and 366 days calendars. The relative time axis is preferably used by weather forecast 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".

1.4.3. 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'.

1.5. Parameter table

A parameter table is an ASCII formatted file to convert code numbers to variable names. Each variable has one line with the code number, the name and the description with optional units in a blank separated list. It can be used only 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]

1.6. Missing values

All 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.

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

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.

1.6.1. 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 $(1 + 2 + \textit{miss} + 3)/4 = \textit{miss}/4 = \textit{miss}$. 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. Similar 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 <code>ifile</code>
$i(t, x)$	Element number x of the field at timestep t of <code>ifile</code>
$o(t)$	Timestep t of <code>ofile</code>
$o(t, x)$	Element number x of the field at timestep t of <code>ofile</code>

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:

info	Dataset information listed by code number
infov	Dataset information listed by variable name
map	Dataset information and simple map
sinfo	Short dataset information listed by code number
sinfov	Short dataset information listed by variable name
diff	Compare two datasets listed by code number
diffv	Compare two datasets listed by variable name
ncode	Number of codes
nvar	Number of variables
nlevel	Number of levels
nyear	Number of years
nmon	Number of months
ndate	Number of dates
ntime	Number of time steps
showformat	Show file format
showcode	Show codes
showvar	Show variable names
showstdname	Show standard names
showlevel	Show levels
showyear	Show years
showmon	Show months
showdate	Show dates
showtime	Show time steps
vardes	Variable description
griddes	Grid description
vct	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 datasets to standard output. The information displayed depends on the actual operator.

Operators

- info** Dataset information listed by code number
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
 - Code number 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!
- infov** Dataset information listed by variable name
The same as operator [info](#) but using the name instead of the code number to identify the variables.
- 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 rectangular grid.

Example

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

```
cdo info ifile
```

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

-1 :	Date	Time	Code	Level	Size	Miss :	Minimum	Mean	Maximum
1 :	1987-01-31	12:00	139	0	2048	1361 :	232.77	266.65	305.31
2 :	1987-02-28	12:00	139	0	2048	1361 :	233.64	267.11	307.15
3 :	1987-03-31	12:00	139	0	2048	1361 :	225.31	267.52	307.67
4 :	1987-04-30	12:00	139	0	2048	1361 :	215.68	268.65	310.47
5 :	1987-05-31	12:00	139	0	2048	1361 :	215.78	271.53	312.49
6 :	1987-06-30	12:00	139	0	2048	1361 :	212.89	272.80	314.18
7 :	1987-07-31	12:00	139	0	2048	1361 :	209.52	274.29	316.34
8 :	1987-08-31	12:00	139	0	2048	1361 :	210.48	274.41	315.83
9 :	1987-09-30	12:00	139	0	2048	1361 :	210.48	272.37	312.86
10 :	1987-10-31	12:00	139	0	2048	1361 :	219.46	270.53	309.51
11 :	1987-11-30	12:00	139	0	2048	1361 :	230.98	269.85	308.61
12 :	1987-12-31	12:00	139	0	2048	1361 :	241.25	269.94	309.27

2.1.2. SINFO - Short information

Synopsis

```
<operator> ifile
```

Description

This module writes information about the structure of all input datasets to standard output. The information displayed depends on the actual operator.

Operators

- sinfo** Short dataset information listed by code number
Prints short information of a dataset. The information is divided into 4 sections. Section 1 prints one line per variable with the following information:
- institute and source
 - parameter table and code number
 - horizontal grid size and number
 - number of vertical levels and zaxis 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.
- sinfov** Short dataset information listed by variable name
The same as operator [sinfo](#) but using the name instead of the code number and parameter table to identify the variables.

Example

To print short information of a dataset use:

```
cdo sinfo ifile
```

This is the result of an ECHAM5 dataset with 3 variables and 12 time steps:

```

-1 : Institut Source Table Code Time Typ Grid Size Num Levels Num
 1 : MPIMET ECHAM5.3 128 129 constant F32 2048 1 1 1
 2 : MPIMET ECHAM5.3 128 130 variable F32 2048 1 4 2
 3 : MPIMET ECHAM5.3 128 139 variable 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 YYYY-MM-DD hh:mm YYYY-MM-DD hh:mm YYYY-MM-DD hh:mm
1987-01-31 12:00 1987-02-28 12:00 1987-03-31 12:00 1987-04-30 12:00
1987-05-31 12:00 1987-06-30 12:00 1987-07-31 12:00 1987-08-31 12:00
1987-09-30 12:00 1987-10-31 12:00 1987-11-30 12:00 1987-12-31 12:00

```

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 must have the same structure and the fields must have the same header information and dimensions.

Operators

- diff** Compare two datasets listed by code number
Provides statistics on differences between two datasets. For each pair of fields the operator prints one line with the following information:
- date and time
 - code number and level
 - size of the grid and number of missing values
 - occurrence of coefficient pairs with different signs
 - occurrence of zero values
 - maxima of absolute difference of coefficient pairs
 - maxima of relative difference of non-zero coefficient pairs with equal signs
- diffv** Compare two datasets listed by variable name
The same as operator [diff](#). Using the name instead of the code number to identify the variable.

Example

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

```
cdo diff ifile1 ifile2
```

This is an example result of the difference of two datasets with one 2D variable and 12 time steps:

	Date	Time	Code	Level	Size	Miss	:	S	Z	Absdiff	Reldiff
1	: 1987-01-31	12:00	139	0	2048	1361	:	F	F	0.00010681	4.1660e-07
2	: 1987-02-28	12:00	139	0	2048	1361	:	F	F	6.1035e-05	2.3742e-07
3	: 1987-03-31	12:00	139	0	2048	1361	:	F	F	7.6294e-05	3.3784e-07
4	: 1987-04-30	12:00	139	0	2048	1361	:	F	F	7.6294e-05	3.5117e-07
5	: 1987-05-31	12:00	139	0	2048	1361	:	F	F	0.00010681	4.0307e-07
6	: 1987-06-30	12:00	139	0	2048	1361	:	F	F	0.00010681	4.2670e-07
7	: 1987-07-31	12:00	139	0	2048	1361	:	F	F	9.1553e-05	3.5634e-07
8	: 1987-08-31	12:00	139	0	2048	1361	:	F	F	7.6294e-05	2.8849e-07
9	: 1987-09-30	12:00	139	0	2048	1361	:	F	F	7.6294e-05	3.6168e-07
10	: 1987-10-31	12:00	139	0	2048	1361	:	F	F	9.1553e-05	3.5001e-07
11	: 1987-11-30	12:00	139	0	2048	1361	:	F	F	6.1035e-05	2.3839e-07
12	: 1987-12-31	12:00	139	0	2048	1361	:	F	F	9.3553e-05	3.7624e-07

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

Synopsis

`<operator> ifile`

Description

This module prints, according to the actual operator, the number of variables, levels or times of the input dataset.

Operators

ncode	Number of codes Prints the number of variables with different code numbers.
nvar	Number of variables Prints the number of variables with different names.
nlevel	Number of levels Prints the number of levels for each variable.
nyear	Number of years Prints the number of different years.
nmon	Number of months Prints the number of different combinations of years and months.
ndate	Number of dates Prints the number of different dates.
ntime	Number of time steps Prints the number of time steps.

Example

To print the number of variables in a dataset use:

```
cdo nvar ifile
```

To print the number of month in a dataset use:

```
cdo nmon ifile
```

2.1.5. SHOWINFO - Show variables, levels or times

Synopsis

`<operator> ifile`

Description

This module prints, according to the actual operator, the format, variables, levels or times of the input dataset.

Operators

showformat	Show file format Prints the file format of the input dataset.
showcode	Show codes Prints the code number of all different variables.
showvar	Show variable names Prints the name of all different variables.
showstdname	Show standard names Prints the standard name of all different variables.
showlevel	Show levels Prints all levels for each variable.
showyear	Show years Prints all different years.
showmon	Show months Prints all different months.
showdate	Show dates Prints all different dates.
showtime	Show time steps Prints all time steps.

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, according to the actual operator, the description of the variables, the grids or the vertical coordinate table.

Operators

vardes	Variable 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.
griddes	Grid description Prints the description of all grids in a file.
vct	Vertical coordinate table Prints the vertical coordinate table.

Example

Assume an input dataset having three variables with the names geosp, t and tslm1. To print the description of these variables use:

```
cdo vardes ifile
```

Result:

```
129 geosp      surface geopotential (orography) [m^2/s^2]
130 t          temperature [K]
139 tslm1      surface temperature of land [K]
```

Assume all variables of the dataset are on a T21 gaussian 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:

copy	Copy datasets
cat	Concatenate datasets
replace	Replace variables
merge	Merge datasets with different fields
mergetime	Merge datasets sorted by date and time
splitcode	Split codes
splitvar	Split variables
splitlevel	Split levels
splitgrid	Split grids
splitzaxis	Split zaxis
splitrec	Split records
splithour	Split hours
splitday	Split days
splitmon	Split months
splitseas	Split seasons
splityear	Split years

2.2.1. COPY - Copy datasets

Synopsis

```
<operator> ifiles ofile
```

Description

This module contains operators to copy or concatenate datasets. Each input dataset must have the same variables with complete time steps.

Operators

copy	Copy datasets Copies all input datasets to ofile .
cat	Concatenate datasets Concatenates all input datasets and append the result to the end of ofile . If ofile 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 exist 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 variables of **ifile2** and **ifile1** with those of **ifile1** and write the result to **ofile**. Both input datasets must have the same number of time steps.

Example

Assume the first input dataset **ifile1** has three variables with the names **geosp**, **t** and **tslml** and the second input dataset **ifile2** has only the variable **tslml**. To replace the variable **tslml** in **ifile1** with **tslml** 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

merge	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 must have different fields and the same number of time steps.
mergetime	Merge datasets sorted by date and time Merges all time steps of all input files sorted by date and time. After this operation every input time step is in <code>ofile</code> and all time steps are sorted by date and time. Each input file must have the same variables and different time steps.

Example

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

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

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

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

2.2.4. SPLIT - Split a dataset

Synopsis

```
<operator> ifile oprefix
```

Description

This module splits a dataset to several files with names formed from the field header information and **oprefix**.

Operators

splitcode	Split codes Splits a dataset into pieces, one for each different code number. Appends three digits with the code number to oprefix to form the output file names.
splitvar	Split variables Splits a dataset into pieces, one for each variable name. Appends a string with the variable name to oprefix to form the output file names.
splitlevel	Split levels Splits a dataset into pieces, one for each different level. Appends six digits with the level to oprefix to form the output file names.
splitgrid	Split grids Splits a dataset into pieces, one for each different grid. Appends two digits with the grid number to oprefix to form the output file names.
splitzaxis	Split zaxis Splits a dataset into pieces, one for each different zaxis. Appends two digits with the zaxis number to oprefix to form the output file names.
splitrec	Split records Splits a dataset into pieces, one for each record. Appends six digits with the record number to oprefix to form the output file names.

Example

Assume an input GRIB 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 oprefix
```

Description

This module splits time steps of a dataset to several files with names formed from the field header information and **oprefix**.

Operators

splithour	Split hours Splits a file into pieces, one for each different hour. Appends two digits with the hour to oprefix to form the output file names.
splitday	Split days Splits a file into pieces, one for each different day. Appends two digits with the day to oprefix to form the output file names.
splitmon	Split months Splits a file into pieces, one for each different month. Appends two digits with the month to oprefix to form the output file names.
splitseas	Split seasons Splits a file into pieces, one for each different season. Appends three characters with the season to oprefix to form the output file names.
splityear	Split years Splits a file into pieces, one for each different year. Appends four digits with the year to oprefix to form the output file names.

Example

Assume the input GRIB 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
```

2.3. Selection

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

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

selcode	Select codes
delcode	Delete codes
selvar	Select variables
delvar	Delete variables
selstdname	Select standard names
sellevel	Select levels
selgrid	Select grids
selgridname	Select grids by name
selzaxis	Select zaxes
selzaxisname	Select zaxes by name
seltabnum	Select parameter table numbers
selrec	Select records
sel timestep	Select time steps
seltime	Select times
selhour	Select hours
selday	Select days
selmon	Select months
selyear	Select years
selseas	Select seasons
seldate	Select dates
selsmon	Select single month
sellonlatbox	Select a longitude/latitude box
selindexbox	Select an index box

2.3.1. SELECT - Select fields

Synopsis

```

selcode, codes ifile ofile
delcode, codes ifile ofile
selvar, vars ifile ofile
delvar, vars ifile ofile
selstdname, stdnames ifile ofile
sellevel, levels ifile ofile
selgrid, grids ifile ofile
selgridname, gridnames ifile ofile
selzaxis, zaxes ifile ofile
selzaxisname, zaxisnames ifile ofile
seltabnum, tabnums ifile ofile
selrec, records ifile ofile

```

Description

This module selects some fields from `ifile` and writes them to `ofile`. The fields selected depend on the actual operator and the parameters.

Operators

selcode	Select codes Selects all fields with code numbers in a user given list.
delcode	Delete codes Deletes all fields with code numbers in a user given list.
selvar	Select variables Selects all fields with variable names in a user given list.
delvar	Delete variables Deletes all fields with variable names in a user given list.
selstdname	Select standard names Selects all fields with standard names in a user given list.
sellevel	Select levels Selects all fields with levels in a user given list.
selgrid	Select grids Selects all fields with grids in a user given list.
selgridname	Select grids by name Selects all fields with grid names in a user given list.
selzaxis	Select zaxes Selects all fields with zaxes in a user given list.
selzaxisname	Select zaxes by name Selects all fields with zaxis names in a user given list.
seltabnum	Select parameter table numbers Selects all fields with parameter table numbers in a user given list.
selrec	Select records Selects all fields with record numbers in a user given list. This operator can not be used with netCDF data!

Parameter

<i>codes</i>	INTEGER	Comma separated list of code numbers
<i>vars</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>grids</i>	INTEGER	Comma separated list of grid numbers
<i>gridnames</i>	STRING	Comma separated list of grid names
<i>zaxes</i>	INTEGER	Comma separated list of zaxis numbers
<i>zaxisnames</i>	STRING	Comma separated list of zaxis names
<i>tabnums</i>	INTEGER	Comma separated list of parameter table numbers
<i>records</i>	INTEGER	Comma separated list of records

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

seltime,times ifile ofile

selhour,hours ifile ofile

selday,days ifile ofile

selmon,months ifile ofile

selyear,years ifile ofile

selseas,seasons ifile ofile

seldate,date1[,date2] ifile ofile

selsmon,month[,nts1[,nts2]] ifile ofile

```

Description

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

Operators

se timestep	Select time steps Selects all time steps with a time step in a user given list.
seltime	Select times Selects all time steps with a time in a user given list.
selhour	Select hours Selects all time steps with a hour in a user given list.
selday	Select days Selects all time steps with a day in a user given list.
selmon	Select months Selects all time steps with a month in a user given list.
selyear	Select years Selects all time steps with a year in a user given list.
selseas	Select seasons Selects all time steps with a month of a season in a user given list.
seldate	Select dates Selects all time steps with a date in a user given range.
selsmon	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)
<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)
<i>date2</i>	STRING	End date (format YYYY-MM-DDThh:mm)
<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 must have the same horizontal grid.

Operators

sellonlatbox	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.
selindexbox	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 T21 gaussian 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:

ifthen	If then
ifnotthen	If not then
ifthenelse	If then else
ifthenc	If then constant
ifnotthenc	If not then constant

2.4.1. COND - Conditional select one field

Synopsis

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

Description

This module conditional selects field elements from `ifile2` 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".

Operators

ifthen If then

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

ifnotthen If not then

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

Example

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

```
cdo ifthen ifile1 ifile2 ofile
```

2.4.2. COND2 - Conditional select two fields

Synopsis

```
ifthenelse ifile1 ifile2 ifile3 ofile
```

Description

This operator conditional selects field elements from `ifile2` or `ifile3` 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".

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

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 `ifile1` are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false".

Operators

ifthenc	If then constant
	$o(t, x) = \begin{cases} c & \text{if } i(t, x) \neq 0 \wedge i(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = 0 \vee i(t, x) = \text{miss} \end{cases}$
ifnotthenc	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 with 1 if the comparison is true and 0 if the comparison is false.

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 then constant
<code>gec</code>	Greater equal constant
<code>gtc</code>	Greater then 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 with 1 if the comparison is true and 0 if the comparison is false. The type of the comparison depends on the actual operator.

Operators

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

Example

To create a mask with 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 dataset with a constant. The resulting field is a mask with 1 if the comparison is true and 0 if the comparison is false. The type of the comparison depends on the actual 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 then 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 then 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 with 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:

setpartab	Set parameter table
setcode	Set code number
setvar	Set variable name
setlevel	Set level
setdate	Set date
settime	Set time
setday	Set day
setmon	Set month
setyear	Set year
settimeunits	Set time units
settimeaxis	Set time axis
setreftime	Set reference time
setcalendar	Set calendar
shifttime	Shift time steps
chcode	Change code number
chvar	Change variable name
chlevel	Change level
chlevelc	Change level of one code
chlevelv	Change level of one variable
setgrid	Set grid
setgridtype	Set grid type
setzaxis	Set zaxis
setgatt	Set global attribute
setgatts	Set global attributes
invertlat	Invert latitude
invertlon	Invert longitude
invertlatdes	Invert latitude description
invertlondes	Invert longitude description
invertlatdata	Invert latitude data
invertlondata	Invert longitude data
masklonlatbox	Mask a longitude/latitude box
maskindexbox	Mask an index box
setclonlatbox	Set a longitude/latitude box to constant
setcindexbox	Set an index box to constant
enlarge	Enlarge fields
setmissval	Set a new missing value
setctomiss	Set constant to missing value
setmisstoc	Set missing value to constant
setrtomiss	Set range to missing value

2.6.1. SET - Set field info

Synopsis

```
setpartab,table ifile ofile
setcode,code ifile ofile
setvar,name ifile ofile
setlevel,level ifile ofile
```

Description

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

Operators

setpartab	Set parameter table Sets the parameter table for all variables.
setcode	Set code number Sets the code number for all variables to the same given value.
setvar	Set variable name Sets the name of the first variable.
setlevel	Set level Sets the first level of all variables.

Parameter

<i>table</i>	STRING	Parameter table file or name
<i>code</i>	INTEGER	Code number
<i>name</i>	STRING	Variable name
<i>level</i>	FLOAT	New level

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 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 actual operator.

Operators

setdate	Set date Sets the date in every time step to the same given value.
settime	Set time Sets the time in every time step to the same given value.
setday	Set day Sets the day in every time step to the same given value.
setmon	Set month Sets the month in every time step to the same given value.
setyear	Set year Sets the year in every time step to the same given value.
settunits	Set time units Sets the base units of a relative time axis.
settaxis	Set time axis Sets the time axis.
setreftime	Set reference time Sets the reference time of an relative time axis.
setcalendar	Set calendar Sets the calendar of an relative time axis.
shifttime	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 (minutes, hours, days, months, years)
<i>date</i>	STRING	Date (format YYYY-MM-DD)
<i>time</i>	STRING	Time (format HH:MM)
<i>inc</i>	STRING	Optional increment (minutes, hours, days, months, years) [default: 0hour]
<i>calendar</i>	STRING	Calendar (standard, 360days, 365days, 366days)
<i>sval</i>	STRING	Shift value (e.g. -3hour)

Example

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

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

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

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

To shift this time axis by -15 days use:

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

Result of 'cdo showdate ofile':

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

2.6.3. CHANGE - Change field header

Synopsis

```

chcode,oldcode,newcode[,...] ifile ofile
chvar,ovar,nvar,... ifile ofile
chlevel,oldlev,newlev,... ifile ofile
chlevelc,code,oldlev,newlev ifile ofile
chlevelv,var,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 actual operator.

Operators

chcode	Change code number Changes some user given code numbers to new user given values.
chvar	Change variable name Changes some user given variable names to new user given names.
chlevel	Change level Changes some user given levels to new user given values.
chlevelc	Change level of one code Changes one level of a user given code number.
chlevelv	Change level of one variable Changes one level of a user given variable.

Parameter

<i>code</i>	INTEGER	Code number
<i>oldcode,newcode,...</i>	INTEGER	Pairs of old and new code numbers
<i>var</i>	STRING	Variable name
<i>ovar,nvar,...</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 type

Synopsis

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

Description

This module sets the grid description of all fields with the same grid size as the new grid.

Operators

setgrid	Set grid Sets the grid description of all fields.
setgridtype	Set grid type Sets the grid type of all grids to a user given value.

Parameter

<i>grid</i>	STRING	Target grid description file or name
<i>gridtype</i>	STRING	Target grid type (curvilinear or cell)

Example

Assumed a dataset has fields with 2048 gridpoints without or with wrong grid description. To set the grid description of all input fields to a T21 gaussian grid (2048 gridpoints) use:

```
cdo setgrid,t21grid ifile ofile
```

2.6.5. SETZAXIS - Set zaxis type

Synopsis

```
setzaxis,zaxis ifile ofile
```

Description

This operator sets the zaxis description of all variables with the same number of level as the new zaxis.

Parameter

<i>zaxis</i>	STRING	Zaxis description file or name of the target zaxis
--------------	--------	--

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 actual operator the attributes are read from a file or can be specified by a parameter.

Operators

setgatt	Set global attribute Sets one user defined global text attribute.
setgatts	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

From the supported data formats only netCDF can work with 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 fields

Synopsis

```
<operator> ifile ofile
```

Description

This module inverts 2D fields on a rectangular grid. Depending on the actual operator the field, only the data or only the grid description is inverted.

Operators

invertlat	Invert latitude Inverts the latitude of a field.
invertlon	Invert longitude Inverts the longitude of a field.
invertlatdes	Invert latitude description Inverts only the latitude description of a field.
invertlondes	Invert longitude description Inverts only the longitude description of a field.
invertlatdata	Invert latitude data Inverts only the latitude data of a field.
invertlondata	Invert longitude data Inverts only the longitude data of a field.

Example

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

```
cdo invertlat ifile ofile
```

2.6.8. 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 must have the same horizontal grid.

Operators

masklonlatbox	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.
maskindexbox	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 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 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 T21 gaussian grid, the same box can be masked with [maskindexbox](#) by:

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


2.6.9. 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 must have the same horizontal grid.

Operators

setclonlatbox	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.
setcindexbox	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 may be greater then that 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 T21 gaussian grid, the same box can be set with [setcindexbox](#) by:

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

2.6.10. 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 rectangular, 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 must be 1 and the xsize of both grids must have the same size.

Parameter

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

Example

Assumed you want to add two datasets. The first dataset is on a T21 grid (2048 field elements) and the second dataset is only a global mean (1 field element). Before you can add these two datasets the second dataset must be enlarged to the grid size of the first dataset:

```
cdo enlarge,t21grid ifile2 tmpfile
cdo add ifile1 tmpfile ofile
```

Or shorter using operator piping:

```
cdo add ifile1 -enlarge,t21grid ifile2 ofile
```

2.6.11. SETMISS - Set missing value

Synopsis

```
setmissval,miss ifile ofile
setctomiss,c ifile ofile
setmisstoc,c ifile ofile
setrtomiss,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 actual operator.

Operators

setmissval Set a new missing value

$$o(t, x) = \begin{cases} \text{miss} & \text{if } i(t, x) = \text{miss} \\ i(t, x) & \text{if } i(t, x) \neq \text{miss} \end{cases}$$

setctomiss Set constant to missing value

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

setmisstoc Set missing value to constant

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

setrtomiss Set range to missing value

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

Parameter

<i>miss</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 the temperature 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	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	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:

expr	Evaluate expressions
exprf	Evaluate expressions from script file
abs	Absolute value
int	Integer value
nint	Nearest integer value
sqr	Square
sqrt	Square root
exp	Exponential
ln	Natural logarithm
log10	Base 10 logarithm
sin	Sine
cos	Cosine
tan	Tangent
asin	Arc sine
acos	Arc cosine
atan	Arc tangent
addc	Add a constant
subc	Subtract a constant
mulc	Multiply with a constant
divc	Divide by a constant
add	Add two fields
sub	Subtract two fields
mul	Multiply two fields
div	Divide two fields
min	Minimum of two fields
max	Maximum of two fields
atan2	Arc tangent of two fields
ymonadd	Add multi-year monthly time average
ymonsub	Subtract multi-year monthly time average
ymonmul	Multiply multi-year monthly time average
ymonddiv	Divide multi-year monthly time average
muldpm	Multiply with days per month
divdpm	Divide by days per month
muldpy	Multiply with days per year
divdpy	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 must end with a semi-colon. The basic arithmetic operations addition +, subtraction −, multiplication *, division / and exponentiation ^ can be used. The following intrinsic functions are available:

<code>sqrt(x)</code>	Square Root of x
<code>exp(x)</code>	Exponential of x
<code>log(x)</code>	Natural logarithm of x
<code>log10(x)</code>	Base 10 logarithm of x
<code>sin(x)</code>	Sine of x, where x is specified in radians
<code>cos(x)</code>	Cosine of x, where x is specified in radians
<code>tan(x)</code>	Tangent of x, where x is specified in radians
<code>asin(x)</code>	Arc-sine of x, where x is specified in radians
<code>acos(x)</code>	Arc-cosine of x, where x is specified in radians
<code>atan(x)</code>	Arc-tangent of x, where x is specified in radians

Operators

expr	Evaluate expressions The processing instructions are read from the parameter.
exprf	Evaluate expressions from script file Contrary to expr 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

abs	Absolute value $o(t, x) = \text{abs}(i(t, x))$
int	Integer value $o(t, x) = \text{int}(i(t, x))$
nint	Nearest integer value $o(t, x) = \text{nint}(i(t, x))$
sqr	Square $o(t, x) = i(t, x)^2$
sqrt	Square root $o(t, x) = \sqrt{i(t, x)}$
exp	Exponential $o(t, x) = e^{i(t, x)}$
ln	Natural logarithm $o(t, x) = \ln(i(t, x))$
log10	Base 10 logarithm $o(t, x) = \log_{10}(i(t, x))$
sin	Sine $o(t, x) = \sin(i(t, x))$
cos	Cosine $o(t, x) = \cos(i(t, x))$
tan	Tangent $o(t, x) = \tan(i(t, x))$
asin	Arc sine $o(t, x) = \arcsin(i(t, x))$
acos	Arc cosine $o(t, x) = \arccos(i(t, x))$
atan	Arc tangent $o(t, x) = \arctan(i(t, x))$

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 header and date information in `ofile` is the same as in `ifile`.

Operators

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

Parameter

`c` 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 header and date information in `ofile` is the same as in `ifile1`.

Operators

add	Add two fields $o(t, x) = i_1(t, x) + i_2(t, x)$
sub	Subtract two fields $o(t, x) = i_1(t, x) - i_2(t, x)$
mul	Multiply two fields $o(t, x) = i_1(t, x) * i_2(t, x)$
div	Divide two fields $o(t, x) = i_1(t, x) / i_2(t, x)$
min	Minimum of two fields $o(t, x) = \min(i_1(t, x), i_2(t, x))$
max	Maximum of two fields $o(t, x) = \max(i_1(t, x), i_2(t, x))$
atan2	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. YMONARITH - Multi-year monthly arithmetic

Synopsis

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

Description

This module performs simple arithmetic of a time series and a 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` must be the same as in `ifile2`. Usually `ifile2` is generated by a call of the module [YMONSTAT](#).

Operators

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

Example

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

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

2.7.6. 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.

Operators

muldpm	Multiply with days per month $o(t, x) = i(t, x) * days_per_month$
divdpm	Divide by days per month $o(t, x) = i(t, x) / days_per_month$
muldpy	Multiply with days per year $o(t, x) = i(t, x) * days_per_year$
divdpy	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 yearavg tmpfile1 tmpfile2
cdo mulc,12 -divdpy tmpfile2 ofile
```

Or all in one command line:

```
cdo mulc,12 -divdpy -yearavg -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:

sum	$\sum_{i=1}^n x_i$
mean resp. avg	$n^{-1} \sum_{i=1}^n x_i$
mean resp. avg 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 var	$n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2$
var 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 std	$\sqrt{n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2}$
std weighted by $\{w_i, i = 1, \dots, n\}$	$\sqrt{\left(\sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left(x_i - \left(\sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right)^2}$

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

ensmin	Ensemble minimum
ensmax	Ensemble maximum
enssum	Ensemble sum
ensmean	Ensemble mean
ensavg	Ensemble average
ensvar	Ensemble variance
ensstd	Ensemble standard deviation
enspctl	Ensemble percentiles
fldmin	Field minimum
fldmax	Field maximum
fldsum	Field sum
fldmean	Field mean
fldavg	Field average
fldvar	Field variance
fldstd	Field standard deviation
fldpctl	Field percentiles

zonmin	Zonal minimum
zonmax	Zonal maximum
zonsum	Zonal sum
zonmean	Zonal mean
zonavg	Zonal average
zonvar	Zonal variance
zonstd	Zonal standard deviation
zonpctl	Zonal percentiles
mermin	Meridional minimum
mermax	Meridional maximum
mersum	Meridional sum
mermean	Meridional mean
meravg	Meridional average
mervar	Meridional variance
merstd	Meridional standard deviation
merpctl	Meridional percentiles
vertmin	Vertical minimum
vertmax	Vertical maximum
vertsum	Vertical sum
vertmean	Vertical mean
vertavg	Vertical average
vertvar	Vertical variance
vertstd	Vertical standard deviation
selmin	Time range minimum
selmax	Time range maximum
selsum	Time range sum
selmean	Time range mean
selavg	Time range average
selvar	Time range variance
selstd	Time range standard deviation
selpctl	Time range percentiles
runmin	Running minimum
runmax	Running maximum
runsum	Running sum
runmean	Running mean
runavg	Running average
runvar	Running variance
runstd	Running standard deviation
runpctl	Running percentiles
timmin	Time minimum
timmax	Time maximum
tisum	Time sum
timmean	Time mean
timavg	Time average
timvar	Time variance
timstd	Time standard deviation
timpctl	Time percentiles

hourmin	Hourly minimum
hourmax	Hourly maximum
hoursum	Hourly sum
hourmean	Hourly mean
houravg	Hourly average
hourvar	Hourly variance
hourstd	Hourly standard deviation
hourpctl	Hourly percentiles
daymin	Daily minimum
daymax	Daily maximum
daysum	Daily sum
daymean	Daily mean
dayavg	Daily average
dayvar	Daily variance
daystd	Daily standard deviation
daypctl	Daily percentiles
monmin	Monthly minimum
monmax	Monthly maximum
monsum	Monthly sum
monmean	Monthly mean
monavg	Monthly average
monvar	Monthly variance
monstd	Monthly standard deviation
monpctl	Monthly percentiles
yearmin	Yearly minimum
yearmax	Yearly maximum
yearsum	Yearly sum
yearmean	Yearly mean
yearavg	Yearly average
yearvar	Yearly variance
yearstd	Yearly standard deviation
yearpctl	Yearly percentiles
seasmin	Seasonal minimum
seasmax	Seasonal maximum
seassum	Seasonal sum
seasmean	Seasonal mean
seasavg	Seasonal average
seasvar	Seasonal variance
seasstd	Seasonal standard deviation
seaspctl	Seasonal percentiles
ydaymin	Multi-year daily minimum
ydaymax	Multi-year daily maximum
ydaysum	Multi-year daily sum
ydaymean	Multi-year daily mean
ydayavg	Multi-year daily average
ydayvar	Multi-year daily variance
ydaystd	Multi-year daily standard deviation

ydaypctl	Multi-year daily percentiles
ymonmin	Multi-year monthly minimum
ymonmax	Multi-year monthly maximum
ymonsum	Multi-year monthly sum
ymonmean	Multi-year monthly mean
ymonavg	Multi-year monthly average
ymonvar	Multi-year monthly variance
ymonstd	Multi-year monthly standard deviation
ymonpctl	Multi-year monthly percentiles
yseasmin	Multi-year seasonal minimum
yseasmax	Multi-year seasonal maximum
yseassum	Multi-year seasonal sum
yseasmean	Multi-year seasonal mean
yseasavg	Multi-year seasonal average
yseasvar	Multi-year seasonal variance
yseasstd	Multi-year seasonal standard deviation
yseaspctl	Multi-year seasonal percentiles
ydrunmin	Multi-year daily running minimum
ydrunmax	Multi-year daily running maximum
ydrunsum	Multi-year daily running sum
ydrunmean	Multi-year daily running mean
ydrunavg	Multi-year daily running average
ydrunvar	Multi-year daily running variance
ydrunstd	Multi-year daily running standard deviation
ydrunpctl	Multi-year daily running percentiles

2.8.1. 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 actual operator the minimum, maximum, sum, average, variance, standard deviation, or a certain percentile over all input files is written to `ofile`. The date information for a time step in `ofile` is the date of the first input file.

Operators

ensmin	Ensemble minimum $o(t, x) = \mathbf{min}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
ensmax	Ensemble maximum $o(t, x) = \mathbf{max}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
enssum	Ensemble sum $o(t, x) = \mathbf{sum}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
ensmean	Ensemble mean $o(t, x) = \mathbf{mean}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
ensavg	Ensemble average $o(t, x) = \mathbf{avg}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
ensvar	Ensemble variance $o(t, x) = \mathbf{var}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
ensstd	Ensemble standard deviation $o(t, x) = \mathbf{std}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
enspctl	Ensemble percentiles $o(t, x) = \mathbf{pth\ percentile}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$

Parameter

p INTEGER Percentile number in 1, ..., 99

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
```

Or shorter with filename substitution:

```
cdo enspctl,50 ifile[1-6] ofile
```


2.8.2. 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 actual operator the field minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to ofile.

Operators

fldmin	Field minimum For every gridpoint x_1, \dots, x_n of the same field, it is: $o(t, 1) = \mathbf{min}\{i(t, x'), x_1 < x' \leq x_n\}$
fldmax	Field maximum For every gridpoint x_1, \dots, x_n of the same field, it is: $o(t, 1) = \mathbf{max}\{i(t, x'), x_1 < x' \leq x_n\}$
fldsum	Field sum For every gridpoint x_1, \dots, x_n of the same field, it is: $o(t, 1) = \mathbf{sum}\{i(t, x'), x_1 < x' \leq x_n\}$
fldmean	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.
fldavg	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.
fldvar	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.
fldstd	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.
fldpctl	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 INTEGER Percentile number in 1, ..., 99

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
```

2.8.3. 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 actual operator the zonal minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to ofile. All input fields must have the same rectangular grid.

Operators

zonmin	Zonal minimum For every latitude the minimum over all longitudes is computed.
zonmax	Zonal maximum For every latitude the maximum over all longitudes is computed.
zonsum	Zonal sum For every latitude the sum over all longitudes is computed.
zonmean	Zonal mean For every latitude the mean over all longitudes is computed.
zonavg	Zonal average For every latitude the average over all longitudes is computed.
zonvar	Zonal variance For every latitude the variance over all longitudes is computed.
zonstd	Zonal standard deviation For every latitude the standard deviation over all longitudes is computed.
zonpctl	Zonal percentiles For every latitude the pth percentile over all longitudes is computed.

Parameter

p INTEGER Percentile number in 1, ..., 99

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
```

2.8.4. 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 actual operator the meridional minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to `ofile`. All input fields must have the same rectangular grid.

Operators

mermin	Meridional minimum For every longitude the minimum over all latitudes is computed.
mermax	Meridional maximum For every longitude the maximum over all latitudes is computed.
mersum	Meridional sum For every longitude the sum over all latitudes is computed.
mermean	Meridional mean For every longitude the area weighted mean over all latitudes is computed.
meravg	Meridional average For every longitude the area weighted average over all latitudes is computed.
mervar	Meridional variance For every longitude the variance over all latitudes is computed.
merstd	Meridional standard deviation For every longitude the standard deviation over all latitudes is computed.
merpctl	Meridional percentiles For every longitude the <i>p</i> th percentile over all latitudes is computed.

Parameter

p INTEGER Percentile number in 1, ..., 99

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
```

2.8.5. VERTSTAT - Vertical statistical values

Synopsis

```
<operator> ifile ofile
```

Description

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

Operators

vertmin	Vertical minimum For every gridpoint the minimum over all levels is computed.
vertmax	Vertical maximum For every gridpoint the maximum over all levels is computed.
vertsum	Vertical sum For every gridpoint the sum over all levels is computed.
vertmean	Vertical mean For every gridpoint the mean over all levels is computed.
vertavg	Vertical average For every gridpoint the average over all levels is computed.
vertvar	Vertical variance For every gridpoint the variance over all levels is computed.
vertstd	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
```

2.8.6. SELSTAT - 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 actual operator the average, minimum, maximum, sum, average, variance or standard deviation of the selected time steps is written to **ofile**. The date information for a time step in **ofile** is the date of the last contributing time step in **ifile**.

Operators

selmin	Time range minimum For every adjacent sequence t_1, \dots, t_n of timesteps of the same selected time range, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
selmax	Time range maximum For every adjacent sequence t_1, \dots, t_n of timesteps of the same selected time range, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
selsum	Time range sum For every adjacent sequence t_1, \dots, t_n of timesteps of the same selected time range, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
selmean	Time range mean For every adjacent sequence t_1, \dots, t_n of timesteps of the same selected time range, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
selavg	Time range average For every adjacent sequence t_1, \dots, t_n of timesteps of the same selected time range, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
selvar	Time range variance For every adjacent sequence t_1, \dots, t_n of timesteps of the same selected time range, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
selstd	Time range standard deviation For every adjacent sequence t_1, \dots, t_n of timesteps of the same selected time range, it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

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 must be skipped:

```
cdo selmean,3,2 ifile ofile
```


2.8.7. SELPCTL - Time range percentile values

Synopsis

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

Description

This module 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 100. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files **ifile2** and **ifile3** must be the result of corresponding selmin and selmax operations, respectively. The date information for 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 timesteps of the same selected time range, it is:

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

Parameter

<i>p</i>	INTEGER	Percentile number in 1, ..., 99
<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

CDO_PCTL_NBINS	Sets the number of histogram bins. The default number is 100.
----------------	---

2.8.8. 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 actual 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 medium contributing time step in **ifile**.

Operators

runmin	Running minimum $o(t + (nts - 1)/2, x) = \min\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
runmax	Running maximum $o(t + (nts - 1)/2, x) = \max\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
runsum	Running sum $o(t + (nts - 1)/2, x) = \text{sum}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
runmean	Running mean $o(t + (nts - 1)/2, x) = \text{mean}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
runavg	Running average $o(t + (nts - 1)/2, x) = \text{avg}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
runvar	Running variance $o(t + (nts - 1)/2, x) = \text{std}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
runstd	Running standard deviation $o(t + (nts - 1)/2, x) = \text{std}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$

Parameter

nts INTEGER Number of time steps

Example

To compute the running mean over 9 time steps, use:

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

2.8.9. 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>	INTEGER	Percentile number in 1, ..., 99
<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 -runmin,9 ifile -runmax,9 ifile ofile
```

2.8.10. 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 actual operator the minimum, maximum, sum, average, variance or standard deviation of all time steps read from `ifile` is written to `ofile`. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Operators

timmin	Time minimum $o(1, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
timmax	Time maximum $o(1, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
timsum	Time sum $o(1, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
timmean	Time mean $o(1, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
timavg	Time average $o(1, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
timvar	Time variance $o(1, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
timstd	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
```

2.8.11. TIMPCTL - Percentile values over all time steps

Synopsis

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

Description

This module 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 100. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files **ifile2** and **ifile3** must be the result of corresponding timmin and timmax operations, respectively. The date information for 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

p INTEGER Percentile number in 1, ..., 99

Environment

CDO_PCTL_NBINS Sets the number of histogram bins. The default number is 100.

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
```

2.8.12. HOURSTAT - Hourly statistical values

Synopsis

```
<operator> ifile ofile
```

Description

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

Operators

hourmin	Hourly minimum For every adjacent sequence t_1, \dots, t_n of time steps of the same hour, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
hourmax	Hourly maximum For every adjacent sequence t_1, \dots, t_n of time steps of the same hour, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
hoursum	Hourly sum For every adjacent sequence t_1, \dots, t_n of time steps of the same hour, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
hourmean	Hourly mean For every adjacent sequence t_1, \dots, t_n of time steps of the same hour, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
houravg	Hourly average For every adjacent sequence t_1, \dots, t_n of time steps of the same hour, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
hourvar	Hourly variance For every adjacent sequence t_1, \dots, t_n of time steps of the same hour, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
hourstd	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
```

2.8.13. HOURPCTL - Hourly percentile values

Synopsis

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

Description

This module computes percentiles over all time steps in *ifile1* of the same hour. The algorithm uses histograms with minimum and maximum bounds given in *ifile2* and *ifile3*, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files *ifile2* and *ifile3* must be the result of corresponding hourmin and hourmax operations, respectively. The date information for 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

p INTEGER Percentile number in 1, ..., 99

Environment

CDO_PCTL_NBINS Sets the number of histogram bins. The default number is 100.

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
```

2.8.14. DAYSTAT - Daily statistical values

Synopsis

```
<operator> ifile ofile
```

Description

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

Operators

daymin	Daily minimum For every adjacent sequence t_1, \dots, t_n of time steps of the same day, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
daymax	Daily maximum For every adjacent sequence t_1, \dots, t_n of time steps of the same day, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
daysum	Daily sum For every adjacent sequence t_1, \dots, t_n of time steps of the same day, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
daymean	Daily mean For every adjacent sequence t_1, \dots, t_n of time steps of the same day, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
dayavg	Daily average For every adjacent sequence t_1, \dots, t_n of time steps of the same day, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
dayvar	Daily variance For every adjacent sequence t_1, \dots, t_n of time steps of the same day, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
daystd	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
```


2.8.15. DAYPCTL - Daily percentile values

Synopsis

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

Description

This module computes percentiles over all time steps in **ifile1** of the same day. The algorithm uses histograms with minimum and maximum bounds given in **ifile2** and **ifile3**, respectively. The default number of histogram bins is 100. The default can be overridden by defining the environment variable **CDO_PCTL_NBINS**. The files **ifile2** and **ifile3** must be the result of corresponding **daymin** and **daymax** operations, respectively. The date information for 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

p INTEGER Percentile number in 1, ..., 99

Environment

CDO_PCTL_NBINS Sets the number of histogram bins. The default number is 100.

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
```

2.8.16. MONSTAT - Monthly statistical values

Synopsis

```
<operator> ifile ofile
```

Description

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

Operators

monmin	Monthly minimum For every adjacent sequence t_1, \dots, t_n of time steps of the same month, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
monmax	Monthly maximum For every adjacent sequence t_1, \dots, t_n of time steps of the same month, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
monsum	Monthly sum For every adjacent sequence t_1, \dots, t_n of time steps of the same month, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
monmean	Monthly mean For every adjacent sequence t_1, \dots, t_n of time steps of the same month, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
monavg	Monthly average For every adjacent sequence t_1, \dots, t_n of time steps of the same month, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
monvar	Monthly variance For every adjacent sequence t_1, \dots, t_n of time steps of the same month, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
monstd	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
```

2.8.17. MONPCTL - Monthly percentile values

Synopsis

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

Description

This module computes percentiles over all time steps in *ifile1* of the same month. The algorithm uses histograms with minimum and maximum bounds given in *ifile2* and *ifile3*, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files *ifile2* and *ifile3* must be the result of corresponding monmin and monmax operations, respectively. The date information for 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

p INTEGER Percentile number in 1, ..., 99

Environment

CDO_PCTL_NBINS Sets the number of histogram bins. The default number is 100.

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
```

2.8.18. YEARSTAT - Yearly statistical values

Synopsis

```
<operator> ifile ofile
```

Description

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

Operators

yearmin	Yearly minimum For every adjacent sequence t_1, \dots, t_n of time steps of the same year, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
yearmax	Yearly maximum For every adjacent sequence t_1, \dots, t_n of time steps of the same year, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
yearsum	Yearly sum For every adjacent sequence t_1, \dots, t_n of time steps of the same year, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
yearmean	Yearly mean For every adjacent sequence t_1, \dots, t_n of time steps of the same year, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
yearavg	Yearly average For every adjacent sequence t_1, \dots, t_n of time steps of the same year, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
yearvar	Yearly variance For every adjacent sequence t_1, \dots, t_n of time steps of the same year, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
yearstd	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** computes only an arithmetical mean!

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 mulc,12 -divdpy -yearmean -muldpm ifile ofile
```

2.8.19. YEARPCTL - Yearly percentile values

Synopsis

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

Description

This module computes percentiles over all time steps in `ifile1` of the same year. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must be the result of corresponding `yearmin` and `yearmax` operations, respectively. The date information for 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

`p` INTEGER Percentile number in 1, ..., 99

Environment

`CDO_PCTL_NBINS` Sets the number of histogram bins. The default number is 100.

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
```

2.8.20. SEASSTAT - Seasonal statistical values

Synopsis

```
<operator> ifile ofile
```

Description

This module computes statistical values over time steps of the same season. Depending on the actual operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same season is written to `ofile`. The date information for 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

seasmin	Seasonal minimum For every adjacent sequence t_1, \dots, t_n of time steps of the same season, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
seasmax	Seasonal maximum For every adjacent sequence t_1, \dots, t_n of time steps of the same season, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
seassum	Seasonal sum For every adjacent sequence t_1, \dots, t_n of time steps of the same season, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
seasmean	Seasonal mean For every adjacent sequence t_1, \dots, t_n of time steps of the same season, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
seasavg	Seasonal average For every adjacent sequence t_1, \dots, t_n of time steps of the same season, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
seasvar	Seasonal variance For every adjacent sequence t_1, \dots, t_n of time steps of the same season, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
seasstd	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
```

2.8.21. SEASPCTL - Seasonal percentile values

Synopsis

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

Description

This module 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 100. The default can be overridden by setting the environment variable **CDO_PCTL_NBINS** to a different value. The files **ifile2** and **ifile3** must be the result of corresponding **seasmin** and **seasmax** operations, respectively. The date information for 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

p INTEGER Percentile number in 1, ..., 99

Environment

CDO_PCTL_NBINS Sets the number of histogram bins. The default number is 100.

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
```

2.8.22. YDAYSTAT - Multi-year daily statistical values

Synopsis

`<operator> ifile ofile`

Description

This module writes to `ofile`, according to the actual operator, the minimum, maximum, sum, average, variance or standard deviation of each day of year in `ifile`. The date information in an output field is the date of the last contributing input field.

Operators

ydaymin	Multi-year daily minimum $o(001, x) = \min\{i(t, x), \text{day}(i(t)) = 001\}$ \vdots $o(366, x) = \min\{i(t, x), \text{day}(i(t)) = 366\}$
ydaymax	Multi-year daily maximum $o(001, x) = \max\{i(t, x), \text{day}(i(t)) = 001\}$ \vdots $o(366, x) = \max\{i(t, x), \text{day}(i(t)) = 366\}$
ydaysum	Multi-year daily sum $o(001, x) = \text{sum}\{i(t, x), \text{day}(i(t)) = 001\}$ \vdots $o(366, x) = \text{sum}\{i(t, x), \text{day}(i(t)) = 366\}$
ydaymean	Multi-year daily mean $o(001, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 001\}$ \vdots $o(366, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 366\}$
ydayavg	Multi-year daily average $o(001, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 001\}$ \vdots $o(366, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 366\}$
ydayvar	Multi-year daily variance $o(001, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 001\}$ \vdots $o(366, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 366\}$
ydaystd	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
```


2.8.23. YDAYPCTL - Multi-year daily percentile values

Synopsis

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

Description

This module writes to `ofile` a certain percentile of each day of 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 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must 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` INTEGER Percentile number in 1, ..., 99

Environment

`CDO_PCTL_NBINS` Sets the number of histogram bins. The default number is 100.

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
```

2.8.24. YMONSTAT - Multi-year monthly statistical values

Synopsis

`<operator> ifile ofile`

Description

This module writes to `ofile`, according to the actual operator, the minimum, maximum, sum, average, variance or standard deviation of each month of year in `ifile`. The date information in an output field is the date of the last contributing input field.

Operators

ymonmin	Multi-year monthly minimum $o(01, x) = \mathbf{min}\{i(t, x), \text{month}(i(t)) = 01\}$ \vdots $o(12, x) = \mathbf{min}\{i(t, x), \text{month}(i(t)) = 12\}$
ymonmax	Multi-year monthly maximum $o(01, x) = \mathbf{max}\{i(t, x), \text{month}(i(t)) = 01\}$ \vdots $o(12, x) = \mathbf{max}\{i(t, x), \text{month}(i(t)) = 12\}$
ymonsum	Multi-year monthly sum $o(01, x) = \mathbf{sum}\{i(t, x), \text{month}(i(t)) = 01\}$ \vdots $o(12, x) = \mathbf{sum}\{i(t, x), \text{month}(i(t)) = 12\}$
ymonmean	Multi-year monthly mean $o(01, x) = \mathbf{mean}\{i(t, x), \text{month}(i(t)) = 01\}$ \vdots $o(12, x) = \mathbf{mean}\{i(t, x), \text{month}(i(t)) = 12\}$
ymonavg	Multi-year monthly average $o(01, x) = \mathbf{avg}\{i(t, x), \text{month}(i(t)) = 01\}$ \vdots $o(12, x) = \mathbf{avg}\{i(t, x), \text{month}(i(t)) = 12\}$
ymonvar	Multi-year monthly variance $o(01, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 01\}$ \vdots $o(12, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 12\}$
ymonstd	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
```

2.8.25. YMONPCTL - Multi-year monthly percentile values

Synopsis

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

Description

This module writes to `ofile` a certain percentile of each month of 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 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must 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.

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

$$\vdots$$

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

Parameter

`p` INTEGER Percentile number in 1, ..., 99

Environment

`CDO_PCTL_NBINS` Sets the number of histogram bins. The default number is 100.

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
```

2.8.26. YSEASSTAT - Multi-year seasonal statistical values

Synopsis

`<operator> ifile ofile`

Description

This module writes to `ofile`, according to the actual operator, the minimum, maximum, sum, average, variance or standard deviation of each season in `ifile`. The date information in an output field is the date of the last contributing input field.

Operators

yseasmin	Multi-year seasonal minimum $o(1, x) = \min\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \min\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \min\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \min\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
yseasmax	Multi-year seasonal maximum $o(1, x) = \max\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \max\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \max\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \max\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
yseassum	Multi-year seasonal sum $o(1, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
yseasmean	Multi-year seasonal mean $o(1, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
yseasavg	Multi-year seasonal average $o(1, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
yseasvar	Multi-year seasonal variance $o(1, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
yseasstd	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
```

2.8.27. YSEASPCTL - Multi-year seasonal percentile values

Synopsis

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

Description

This module writes to `ofile` a certain percentile of each season in `ifile1`. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must be the result of corresponding `yseasmin` and `yseasmax` operations, respectively. The date information in an output field is the date of the last contributing input field.

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

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

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

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

Parameter

`p` INTEGER Percentile number in 1, ..., 99

Environment

`CDO_PCTL_NBINS` Sets the number of histogram bins. The default number is 100.

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
```

2.8.28. YDRUNSTAT - Multi-year daily running statistical values

Synopsis

`<operator>,nts ifile ofile`

Description

This module writes to `ofile` running statistical values for each day of year in `ifile`. Depending on the actual 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 must 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 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

ydrunmin	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 $o(366, x) = \min\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
ydrunmax	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 $o(366, x) = \max\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
ydrunsum	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 $o(366, x) = \text{sum}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
ydrunmean	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 $o(366, x) = \text{mean}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
ydrunavg	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 $o(366, x) = \text{avg}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
ydrunvar	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 $o(366, x) = \text{var}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
ydrunstd	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
```

2.8.29. YDRUNPCTL - Multi-year daily running percentile values

Synopsis

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

Description

This module writes to `ofile` running percentile values for each day of year in `ifile1`. 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 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must 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 must 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 yields physically meaningful results only if the input time series does include the $(nts-1)/2$ days before and after each period of interest.

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

Parameter

<i>p</i>	INTEGER	Percentile number in 1, ..., 99
<i>nts</i>	INTEGER	Number of time steps

Environment

<code>CDO_PCTL_NBINS</code>	Sets the number of histogram bins. The default number is 100.
-----------------------------	---

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. Regression

This sections contains modules for linear regression of time series.

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

detrend	Detrend
trend	Trend
subtrend	Subtract trend

2.9.1. 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

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

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
```

2.9.2. 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 σ^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](#).

2.9.3. 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 detrend the data in `ifile` and to store the detrended data in `ofile` is:

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

The result is identical to operator [detrend](#):

```
cdo detrend ifile ofile
```

2.10. 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 level to height or pressure level. Interpolation in time is possible between time steps and between years.

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

remapbil	Bilinear interpolation
remapbic	Bicubic interpolation
remapcon	Conservative remapping
remapdis	Distance-weighted averaging
genbil	Generate bilinear interpolation weights
genbic	Generate bicubic interpolation weights
gencon	Generate conservative interpolation weights
gendis	Generate distance-weighted averaging weights
remap	SCRIP grid remapping
interpolate	PINGO grid interpolation
intgridbil	Bilinear grid interpolation
ml2pl	Model to pressure level interpolation
ml2hl	Model to height level interpolation
inttime	Time interpolation
intntime	Time interpolation
intyear	Year interpolation

2.10.1. REMAPGRID - SCRIP grid interpolation

Synopsis

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

Description

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

Operators

remapbil	Bilinear interpolation Performs a bilinear interpolation on all input fields. This interpolation method works only on rectangular grids.
remapbic	Bicubic interpolation Performs a bicubic interpolation on all input fields. This interpolation method works only on rectangular grids.
remapcon	Conservative remapping Performs a first order conservative remapping on all input fields.
remapdis	Distance-weighted averaging Performs a distance-weighted average of the four nearest neighbor values on all input fields.

Parameter

<i>grid</i>	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 be 'fracarea' and will include the destination area fraction in the output weights; other options are 'none' and 'destarea' (for more information see [SCRIP]).
---------------	---

Note

For this program the author has converted the original Fortran 90 SCRIP software to ANSI C. In case of any problems send a bug report to CDO and not to SCRIP!

Example

Say *ifile* contains fields on a rectangular grid. Remap all fields bilinear to a T42 gaussian grid:

```
cdo remapbil,t42grid ifile ofile
```

2.10.2. GENWEIGHTS - Generate SCRIP grid interpolation weights

Synopsis

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

Description

Grid interpolation can be a very time consuming process. Especially if the data is on an unstructured or on a large grid. In this case the SCRIP interpolation process can be split into two parts. First generation of the interpolation weights, this is the most time consuming part. These interpolation weights can be reused for every remapping process. This method works 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

genbil	Generate bilinear interpolation weights Generates bilinear interpolation weights and write the result to a file. This interpolation method works only on rectangular grids.
genbic	Generate bicubic interpolation weights Generates bicubic interpolation weights and write the result to a file. This interpolation method works only on rectangular grids.
gencon	Generate conservative interpolation weights Generates first order conservative interpolation weights and write the result to a file.
gendis	Generate distance-weighted averaging weights Generates distance-weighted average weights of the four nearest neighbor values and write the result to a file.

Parameter

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

Environment

NORMALIZE_OPT	This variable is used to choose the normalization of the conservative interpolation. By default, NORMALIZE_OPT is set to be 'fracarea' and will include the destination area fraction in the output weights; other options are 'none' and 'destarea' (for more information see [SCRIP]).
---------------	---

Note

For this program the author has converted the original Fortran 90 SCRIP software to ANSI C. In case of any problems send a bug report to CDO and not to SCRIP!

Example

Say *ifile* contains fields on a rectangular grid. Remap all fields bilinear to a T42 gaussian grid:

```
cdo genbil,t42grid ifile remapweights.nc
cdo remap,t42grid,remapweights ifile ofile
```

2.10.3. REMAP - SCRIP grid remapping

Synopsis

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

Description

This operator remaps all input fields to a new grid. The remap type and the interpolation weights of one grid are read from a netCDF file. The netCDF file with the weights must follow the SCRIP convention. Normaly these weights come from a previous call to module [GENWEIGHTS](#) or was 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)

Note

For this program the author has converted the original Fortran 90 SCRIP software to ANSI C. In case of any problems send a bug report to CDO and not to SCRIP!

Example

Say *ifile* contains fields on a regular grid. Remap all fields bilinear to a T42 gaussian grid:

```
cdo genbil,t42grid ifile remapweights.nc
cdo remap,t42grid,remapweights ifile ofile
```

2.10.4. INTGRID - Grid interpolation

Synopsis

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

Description

This module contains operators to interpolate all input fields to a new grid. All interpolation methods in this module work only on rectangular grids.

Operators

interpolate PINGO grid interpolation
This is the grid interpolation from PINGO. The basis of the interpolation is an underlying continuous field which is constructed in the following way. For two neighboured longitudes x_1 and x_2 and two neighboured latitudes y_1 and y_2 of the input grid every point at longitude x and latitude y with $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$ is assigned the value

$$a = a_{11} + (a_{21} - a_{11}) \frac{x - x_1}{x_2 - x_1} + (a_{12} - a_{11}) \frac{y - y_1}{y_2 - y_1} + (a_{22} - a_{21} - a_{12} + a_{11}) \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)}$$

where a_{ij} is the value at longitude x_i and latitude y_j . If one of the four values a_{11} , a_{12} , a_{21} , a_{22} is the missing value, then a is also the missing value. Afterwards the underlying continuous field is expanded by a half mesh width. For a detailed description of this interpolation method see [\[PINGO\]](#).

intgridbil Bilinear grid interpolation
Performs a bilinear interpolation on all input fields. This implementation is faster than [remapbil](#). The complete source grid must be inside the target grid. Missing values are not supported yet!

Parameter

grid STRING Target grid description file or name

Example

Say *ifile* contains fields on a rectangular grid. To interpolate all fields bilinear to a T42 gaussian grid, use:

```
cdo intgridbil,t42grid ifile ofile
```


2.10.5. INTVERT - Vertical interpolation

Synopsis

```
ml2pl,plevels ifile ofile
```

```
ml2hl,hlevels ifile ofile
```

Description

Interpolate 3D variables on hybrid model level to pressure or height level. The input file must contain the log. surface pressure (LSP/code152) or the surface pressure (APS/code134). To interpolate the temperature, the orography (GEOSP/code129) is also needed.

Operators

ml2pl Model to pressure level interpolation
Interpolates 3D variables on hybrid model level to pressure level.

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

Parameter

<i>plevels</i>	FLOAT	Pressure levels in pascal
<i>hlevels</i>	FLOAT	Height levels in meter (max level: 65535 m)

Environment

EXTRAPOLATE	If set to 1 extrapolate missing values.
-------------	---

Example

To interpolate hybrid model level data to pressure levels of 925, 850, 500 and 200 hPa, use:

```
cdo ml2pl,92500,85000,50000,20000 ifile ofile
```

2.10.6. INTTIME - Time interpolation

Synopsis

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

Description

This module performs linear interpolation between time steps.

Operators

inttime	Time interpolation 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.
intntime	Time interpolation 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)
<i>inc</i>	STRING	Optional increment (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. To interpolate this time series to a one hourly dataset, use:

```
cdo inttime,1987-01-01,12:00,1hour ifile ofile
```

2.10.7. INTYEAR - Year interpolation

Synopsis

```
intyear,years ifile1 ifile2 oprefix
```

Description

This operator performs linear interpolation between two years time step by time step. Appends four digits with the year to **oprefix** to form the output file names.

Parameter

years INTEGER Comma separated list of years

Example

Assumed you have two monthly mean datasets over a year. The first dataset has 12 time steps for year 1985 and the second for 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
```

2.11. Transformation

This section contains modules to perform spectral transformations.

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

sp2gp	Spectral to gridpoint
sp2gpl	Spectral to gridpoint (linear)
gp2sp	Gridpoint to spectral
gp2spl	Gridpoint to spectral (linear)
sp2sp	Spectral to spectral
dv2uv	Divergence and vorticity to U and V wind
dv2uvl	Divergence and vorticity to U and V wind (linear)
uv2dv	U and V wind to divergence and vorticity
uv2dvl	U and V wind to divergence and vorticity (linear)

2.11.1. 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

- sp2gp** Spectral to gridpoint
Convert all fields with spectral coefficients to regular gaussian grid. The number of latitudes of the resulting gaussian grid is calculated from the triangular truncation by:
$$nlat = NINT((trunc * \boxed{3} + 1.) / 2.)$$
- sp2gpl** Spectral to gridpoint (linear)
Convert all fields with spectral coefficients to regular gaussian grid. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:
$$nlat = NINT((trunc * \boxed{2} + 1.) / 2.)$$

Use this operator to convert ERA40 data e.g. from TL159 to N80.
- gp2sp** Gridpoint to spectral
Convert all gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:
$$trunc = (nlat * 2 - 1) / \boxed{3}$$
- gp2spl** Gridpoint to spectral (linear)
Convert all gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:
$$trunc = (nlat * 2 - 1) / \boxed{2}$$

Use this operator to convert ERA40 data e.g. from N80 to TL159 instead of T106.
- sp2sp** Spectral to spectral
Change the triangular truncation of all spectral fields. The operator performs downward conversion by cutting the resolution. Upward conversions are achieved by filling in zeros.

Parameter

trunc INTEGER New spectral resolution

Example

To transform spectral coefficients from T106 to N80 gaussian grid use:

```
cdo sp2gp ifile ofile
```

To transform spectral coefficients from TL159 to N80 gaussian grid use:

```
cdo sp2gpl ifile ofile
```

2.11.2. WIND - Wind transformation

Synopsis

```
<operator> ifile ofile
```

Description

This module converts divergence and vorticity to U and V wind and vice versa.

Operators

- | | |
|---------------|---|
| dv2uv | <p>Divergence and vorticity to U and V wind</p> <p>Calculate U and V wind on a gaussian grid from spherical harmonic coefficients of divergence and vorticity. The U and V wind must have the names u and v or the code numbers 131 and 132. The number of latitudes of the resulting gaussian grid is calculated from the triangular truncation by:</p> $nlat = NINT((trunc * \sqrt{3} + 1.) / 2.)$ |
| dv2uyl | <p>Divergence and vorticity to U and V wind (linear)</p> <p>Calculate U and V wind on a gaussian grid from spherical harmonic coefficients of divergence and vorticity. The U and V wind must have the names u and v or the code numbers 131 and 132. The number of latitudes of the resulting gaussian grid is calculated from the triangular truncation by:</p> $nlat = NINT((trunc * \sqrt{2} + 1.) / 2.)$ |
| uv2dv | <p>U and V wind to divergence and vorticity</p> <p>Calculate spherical harmonic coefficients of divergence and vorticity from U and V wind. The divergence and vorticity must have the names sd and svo or code numbers 155 and 138. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:</p> $trunc = (nlat * 2 - 1) / \sqrt{3}$ |
| uv2dyl | <p>U and V wind to divergence and vorticity (linear)</p> <p>Calculate spherical harmonic coefficients of divergence and vorticity from U and V wind. The divergence and vorticity must have the names sd and svo or code numbers 155 and 138. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:</p> $trunc = (nlat * 2 - 1) / \sqrt{2}$ |

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
```

2.12. Formatted I/O

This section contains modules to read and write ASCII data.

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

input	ASCII input
inputsrv	SERVICE input
inputtext	EXTRA input
output	ASCII output
outputf	Formatted output
outputint	Integer output
outputsrv	SERVICE output
outputtext	EXTRA output

2.12.1. INPUT - Formatted input

Synopsis

`input,grid ofile`

`inputsrv ofile`

`inputtext ofile`

Description

This modules reads time series of one 2D variable from standard input. All input fields must have the same horizontal grid. The format of the input depends on the actual operator.

Operators

input	ASCII input Read fields with ASCII numbers from standard input and stores them in <code>ofile</code> . The numbers that are read are exactly that ones which are written out by output .
inputsrv	SERVICE input Read fields with ASCII numbers from standard input and stores them in <code>ofile</code> . Each field must have a header of 8 integers (SERVICE likely). The numbers that are read are exactly that ones which are written out by outputsrv .
inputtext	EXTRA input Read fields with ASCII numbers from standard input and stores them in <code>ofile</code> . Each field with a header of 4 integers (EXTRA likely). The numbers that are read are exactly that ones which are written out by outputtext .

Parameter

`grid` STRING Grid description file or name

Example

Assume an ASCII dataset contains a field on a global regular grid with 32 longitude and 16 latitudes (512 elements). To create a GRIB dataset from the ASCII dataset use:

```
cdo -f grb input,r32x16 ofile.grb < my_ascii_data
```


2.12.2. OUTPUT - Formatted output

Synopsis

```

output ifiles
outputf,format,nelem ifiles
outputint ifiles
outputsrv ifiles
outputtext ifiles

```

Description

This modules prints all values of all input datasets to standard output. All input fields must have the same horizontal grid. The format of the output depends on the actual operator.

Operators

output	ASCII output Prints all values to standard output. Each row has 6 elements with the C-style format "%13.6g".
outputf	Formatted output Prints all values to standard output. The format and number of elements for each row can be specified by the parameters.
outputint	Integer output Prints all values rounded to the nearest interger to standard output.
outputsrv	SERVICE output Prints all values to standard output. Each field with a header of 8 integers (SERVICE likely).
outputtext	EXTRA 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

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

2.13. 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:

gradsdes1	GrADS data descriptor file (version 1 GRIB map)
gradsdes2	GrADS data descriptor file (version 2 GRIB map)
timsort	Sort over the time
const	Create a constant field
random	Create a field with random values
vardup	Duplicate variables
varmul	Multiply variables
rotuvb	Backward rotation
mastrfu	Mass stream function
hi	Humidity index (C)
wct	Windchill temperature (C)

2.13.1. GRADSDES - GrADS data descriptor file

Synopsis

`<operator> ifile`

Description

Creates a GrADS data descriptor file. Supported file formats are GRIB, SERVICE, EXTRA and IEG. For GRIB files the GrADS map file is also generated. For SERVICE and EXTRA files the grid must 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

- gradsdes1** GrADS data descriptor file (version 1 GRIB map)
Creates a GrADS data descriptor file. Generated a machine specific version 1 GrADS map file for GRIB datasets.
- gradsdes2** GrADS data descriptor file (version 2 GRIB map)
Creates a GrADS data descriptor file. Generated a machine independent version 2 GrADS map file for GRIB 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 GRIB 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 GRIB dataset has 3 variables over 12 time steps on a T21 grid. The contents of the resulting GrADS data description file is approximately:

```
DSET ^ifile.grb
DTYPE GRIB
INDEX ^ifile.gmp
XDEF 64 LINEAR 0.000000 5.625000
YDEF 32 LEVELS -85.761 -80.269 -74.745 -69.213 -63.679 -58.143
              -52.607 -47.070 -41.532 -35.995 -30.458 -24.920
              -19.382 -13.844 -8.307 -2.769 2.769 8.307
              13.844 19.382 24.920 30.458 35.995 41.532
              47.070 52.607 58.143 63.679 69.213 74.745
              80.269 85.761
ZDEF 4 LEVELS 925 850 500 200
TDEF 12 LINEAR 12:00 Z1jan1987 1mo
TITLE ifile.grb T21 grid
OPTIONS yrev
UNDEF -9e+33
VARS 3
geosp 0 129,1,0 surface geopotential (orography) [m^2/s^2]
t      4 130,99,0 temperature [K]
tslm1 0 139,1,0 surface temperature of land [K]
ENDVARS
```

2.13.2. TIMSORT - Timsort

Synopsis

```
timsort ifile ofile
```

Description

Sorts for every field position the elements in ascending order over all time steps. After sorting it is:

$$o(t_1, x) < o(t_2, x) \quad \forall (t_1 < t_2), x$$

Example

To sort all field elements of a dataset over all time steps use:

```
cdo timsort ifile ofile
```

2.13.3. VARGEN - Generate a field

Synopsis

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

Description

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

Operators

const	Create a constant field Creates a constant field. All field elements of the grid have the same value.
random	Create a field with random values Creates a field with rectangularly distributed random numbers in the interval $[0,1]$.

Parameter

<i>const</i>	FLOAT	Constant
<i>grid</i>	STRING	Target grid description file or name

2.13.4. VARDUP - Variable duplication

Synopsis

```
vardup ifile ofile
varmul,nmul ifile ofile
```

Description

Duplicates all variables of a dataset.

Operators

vardup	Duplicate variables Duplicates all variables.
varmul	Multiply variables Multiplies all variables.

Parameter

<i>nmul</i>	INTEGER	Number of multiplications
-------------	---------	---------------------------

2.13.5. ROTUV - Rotation

Synopsis

```
rotuvb,u,v,... ifile ofile
```

Description

This is a special operator for datasets with wind components on an rotated grid, e.g. data from the regional model REMO. It performs a backward transformation of velocity components U and V from an 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 an rotated spherical system to a geographical system use:

```
cdo rotuvb,u,v ifile ofile
```

2.13.6. 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 must 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
```

2.13.7. HI - Humidity index (C)

Synopsis

```
hi ifile1 ifile2 ifile3 ofile
```

Description

Let `ifile1`, `ifile2` and `ifile3` be time series of temperature, vapour pressure and relative humidity records. Then a corresponding time series of resulting humidity index values is written to `ofile`.

2.13.8. WCT - Windchill temperature (C)

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`. Note that the temperature and wind speed records must be given in units of C and m/s, respectively.

2.14. ECA indices

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

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

eca_cdd	Consecutive dry days index per time period
eca_cfd	Consecutive frost days index per time period
eca_csu	Consecutive summer days index per time period
eca_cwd	Consecutive wet days index per time period
eca_cwdi	Cold wave duration index wrt mean of reference period
eca_cwfi	Cold-spell days index wrt 10th percentile of reference period
eca_etr	Intra-period extreme temperature range
eca_fd	Frost days index per time period
eca_fdns	Frost days where no snow index per time period
eca_gsl	Growing season length index
eca_hd	Heating degree days per time period
eca_hwdi	Heat wave duration index wrt mean of reference period
eca_hwfi	Warm spell days index wrt 90th percentile of reference period
eca_id	Ice days index per time period
eca_r10mm	Heavy precipitation days index per time period
eca_r20mm	Very heavy precipitation days index per time period
eca_r75p	Moderate wet days wrt 75th percentile of reference period
eca_r75ptot	Precipitation percent due to R75p days
eca_r90p	Wet days wrt 90th percentile of reference period
eca_r90ptot	Precipitation percent due to R90p days
eca_r95p	Very wet days wrt 95th percentile of reference period
eca_r95ptot	Precipitation percent due to R95p days
eca_r99p	Extremely wet days wrt 99th percentile of reference period
eca_r99ptot	Precipitation percent due to R99p days
eca_rr1	Wet days index per time period
eca_rx1day	Highest one day precipitation amount per time period
eca_rx5day	Highest five-day precipitation amount per time period

eca_sdii	Simple daily intensity index per time period
eca_strwin	Strong wind days index per time period
eca_strbre	Strong breeze days index per time period
eca_strgal	Strong gale days index per time period
eca_hurr	Hurricane days index per time period
eca_su	Summer days index per time period
eca_tg10p	Cold days percent wrt 10th percentile of reference period
eca_tg90p	Warm days percent wrt 90th percentile of reference period
eca_tn10p	Cold nights percent wrt 10th percentile of reference period
eca_tn90p	Warm nights percent wrt 90th percentile of reference period
eca_tr	Tropical nights index per time period
eca_tx10p	Very cold days percent wrt 10th percentile of reference period
eca_tx90p	Very warm days percent wrt 90th percentile of reference period

2.14.1. 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 for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

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

```
cdo eca_cdd rrfil file
```

2.14.2. 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 $TN < 0$ Celsius. Note that `TN` must be given in units of kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

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

```
cdo eca_cfd tnfil file
```

2.14.3. 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 Celsius`. Note that `TN` must be given in units of kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

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 for a time series of daily minimum temperatures, use:

```
cdo eca_csu txfile ofile
```

2.14.4. 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 for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

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

```
cdo eca_cwd rrfile ofile
```

2.14.5. 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*_{norm} 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 Celsius. A further output variable is the number of cold waves longer than or equal to *nday* days. Note that both *TN* and *TN*_{norm} must be given in the same units. The date information for a time step in *ofile* is the date of the last contributing time step in *ifile1*.

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 for a time series of daily minimum temperatures, use:

```
cdo eca_cwdi tnfile tnnormfile ofile
```

2.14.6. 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*_{n10} 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. Note that both *TG* and *TG*_{n10} must be given in the same units. The date information for a time step in *ofile* is the date of the last contributing time step in *ifile1*.

Parameter

<i>nday</i>	INTEGER	Number of consecutive days (default: <i>nday</i> = 6)
-------------	---------	---

Example

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

```
cdo eca_cwfi tgfile tgn10file ofile
```

2.14.7. 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 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time steps in `ifile1` and `ifile2`.

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
```

2.14.8. 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 $TN < 0$ Celsius. Note that TN must be given in units of kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

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

```
cdo eca_fd tnfile ofile
```

2.14.9. ECAFDNS - Frost days where no snow index per time period

Synopsis

```
eca_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 $TN < 0$ Celsius and the surface snow amount is less than 1 cm. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

2.14.10. ECAGSL - Growing season length index

Synopsis

```
eca_gsl[,nday[,T]] ifile ofile
```

Description

Let `ifile` be a time series of daily mean temperatures `TG`, then counted are the number of days between the first occurrence of at least `nday` consecutive days with $TG > T$ and the first occurrence after 1 July of at least `nday` consecutive days with $TG < T$. The numbers `nday` and `T` are optional parameter with default `nday = 6` and `T = 5` Celsius. Further output variables are the start and end dates of the growing season. Note that `TG` must be given in units of kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Parameter

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

Example

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

```
cdo eca_gsl tgfile ofile
```

2.14.11. 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 Celsius is used for both parameters. If only $T1$ is given, $T2$ is set to $T1$. Note that TG must be given in units of kelvin. The date information for a time step in *ofile* is the date of the last contributing time step in *ifile*.

Parameter

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

Example

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

```
cdo eca_hd tgfile ofile
```

2.14.12. 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 Celsius. A further output variable is the number of heat waves longer than or equal to *nday* days. Note that both TX and TXnorm must be given in the same units. The date information for a time step in *ofile* is the date of the last contributing time step in *ifile1*.

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 heat wave duration index for a time series of daily maximum temperatures, use:

```
cdo eca_hwdi txfile txnormfile ofile
```

2.14.13. 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. Note that both `TG` and `TGn90` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

Parameter

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

Example

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

```
cdo eca_hwfi tgfile tgn90file ofile
```

2.14.14. 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 $TX < 0$ Celsius. Note that `TX` must be given in units of kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

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

```
cdo eca_id txfile ofile
```


2.14.15. ECAR10MM - Heavy precipitation days index per time period

Synopsis

```
eca_r10mm 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 10 mm. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

To get the number of days with precipitation greater than 10 mm for a time series of daily precipitation amounts, use:

```
cdo eca_r10mm rfile ofile
```

2.14.16. ECAR20MM - Very heavy precipitation days index per time period

Synopsis

```
eca_r20mm 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 20 mm. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

To get the number of days with precipitation greater than 20 mm for a time series of daily precipitation amounts, use:

```
cdo eca_r20mm rfile ofile
```

2.14.17. 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 $RR > RRn75$. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.18. 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. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

2.14.19. 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 $RR > RRn90$. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.20. 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. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

2.14.21. 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 $RR > RRn95$. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.22. 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. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

2.14.23. 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 $RR > RRn99$. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.24. 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. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

2.14.25. 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 for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

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

```
cdo eca_rr1 rrfile ofile
```

2.14.26. 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 for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Parameter

<i>mode</i>	STRING	Operation mode (optional). If <code>mode = 'm'</code> 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
```

2.14.27. 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 for a time step in `ofile` is the date of the last contributing time step in `ifile`.

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
```

2.14.28. 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 for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Example

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

```
cdo eca_sdii rrfile ofile
```

2.14.29. ECASTRWIN - Strong wind days index per time period

Synopsis

```
eca_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. Note that both VX and v must 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 for 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)
----------------	-------	---

2.14.30. ECASTRBRE - Strong breeze days index per time period

Synopsis

```
eca_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. Note that VX is defined as the square root of the sum of squares of the zonal and meridional wind speeds and must be given in units of m/s. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

2.14.31. ECASTRGAL - Strong gale days index per time period

Synopsis

```
eca_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. Note that VX is defined as the square root of the sum of squares of the zonal and meridional wind speeds and must be given in units of m/s. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

2.14.32. ECAHURR - Hurricane days index per time period

Synopsis

```
eca_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. Note that `VX` is defined as the square root of the sum of squares of the zonal and meridional wind speeds and must be given in units of m/s. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

2.14.33. 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` Celsius. Note that `TX` must be given in units of kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Parameter

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

Example

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

```
cdo eca_su txfile ofile
```

2.14.34. 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 $TG < TGn10$. Note that both TG and TGn10 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.35. 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 $TG > TGn90$. Note that both TG and TGn90 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.36. 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 $TN < TNn10$. Note that both `TN` and `TNn10` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.37. 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 $TN > TNn90$. Note that both `TN` and `TNn90` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.38. 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 $T = 20$ Celsius. Note that `TN` must be given in units of kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

Parameter

<code>T</code>	FLOAT	Temperature threshold (Celsius, default: $T = 20$ Celsius)
----------------	-------	---

Example

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

```
cdo eca_tr tnfile ofile
```

2.14.39. 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 `TNx10` of daily maximum temperatures for any period used as reference. Then calculated is the percentage of time where $TX < TX_{n10}$. Note that both `TX` and `TNx10` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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
```

2.14.40. 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 TXn90 of daily maximum temperatures for any period used as reference. Then calculated is the percentage of time where $TX > TXn90$. Note that both TX and TXn90 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

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](#)

[ECHAM]

The atmospheric general circulation model ECHAM5, from the [Max Planck Institute for Meteorologie](#)

[GRIB]

[GRIB version 1](#), from the World Meteorological Organisation ([WMO](#))

[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

[SCRIP]

[SCRIP Software Package](#), from the Los Alamos National Laboratory

A. Hints for PINGO user

Some **CDO** operators have the same name as in PINGO but the meaning is different. The following table gives an overview of those operators.

Operator name	CDO	PINGO
min	Minimum of two fields	Time minimum
max	Maximum of two fields	Time maximum
daymean	Daily mean	Multi-year daily mean
daymin	Daily minimum	Multi-year daily minimum
daymax	Daily maximum	Multi-year daily maximum
monmean	Monthly mean	Multi-year monthly mean
monmin	Monthly minimum	Multi-year monthly minimum
monmax	Monthly maximum	Multi-year monthly maximum
seasmean	Seasonally mean	Multi-year seasonally mean

There are also some **CDO** operators with the same functionality as in PINGO but the name is different. The following table gives an overview of those operators.

	CDO	PINGO
Maximum of two fields	max	max2
Minimum of two fields	min	min2
Field mean, min, max	fldmean, fldmin, fldmax	meanr minr, maxr
Time mean, min, max	timmean, timmin, timmax	mean, min, max
Daily mean, min, max	daymean, daymin, daymax	daymeans, daymins, daymaxs
Monthly mean, min, max	monmean, monmin, monmax	monmeans, monmins, monmaxs
Yearly mean, min, max	yearmean, yearmin, yearmax	yearmeans, yearmins, yearmaxs
Running mean	runmean	runmeans
Seasonally mean	seasmean	seasmeans
Multi-year daily mean	ydaymean	daymean
Multi-year monthly mean	ymonmean	monmean
Multi-year seasonally mean	yseasmean	seasmean

B. Grid description examples

B.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.

<code>gridtype</code>	:	<code>curvilinear</code>																
<code>gridsize</code>	:	<code>30</code>																
<code>xsize</code>	:	<code>6</code>																
<code>ysize</code>	:	<code>5</code>																
<code>xvals</code>	:	-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							
<code>xbounds</code>	:	-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			
<code>yvals</code>	:	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							
<code>ybounds</code>	:	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			

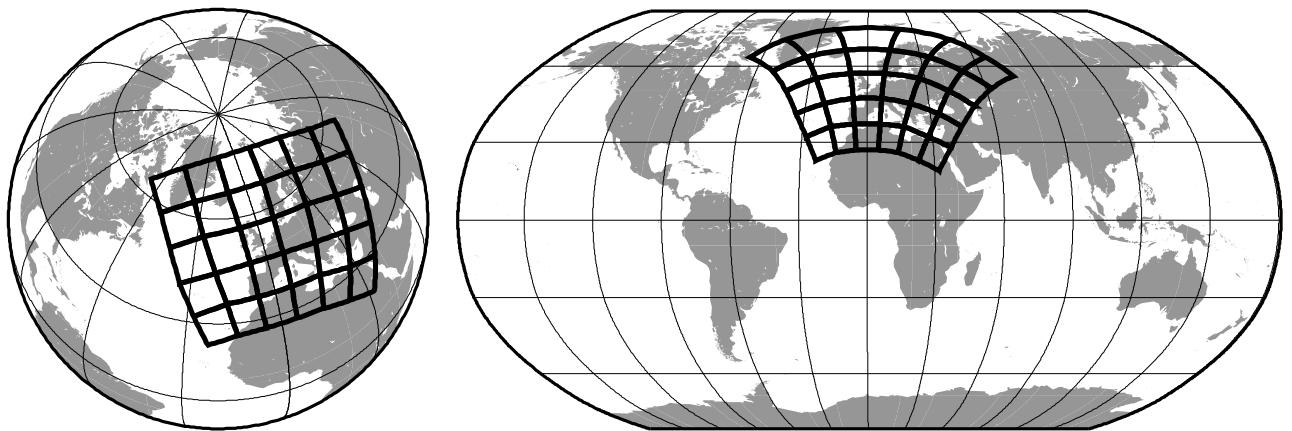


Figure B.1.: Orthographic and Robinson projection of the curvilinear grid

B.2. Example description for unstructured grid cells

Here is an example of the **CDO** description for unstructured grid cells. 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.

```

gridtype : cell
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 -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    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

```

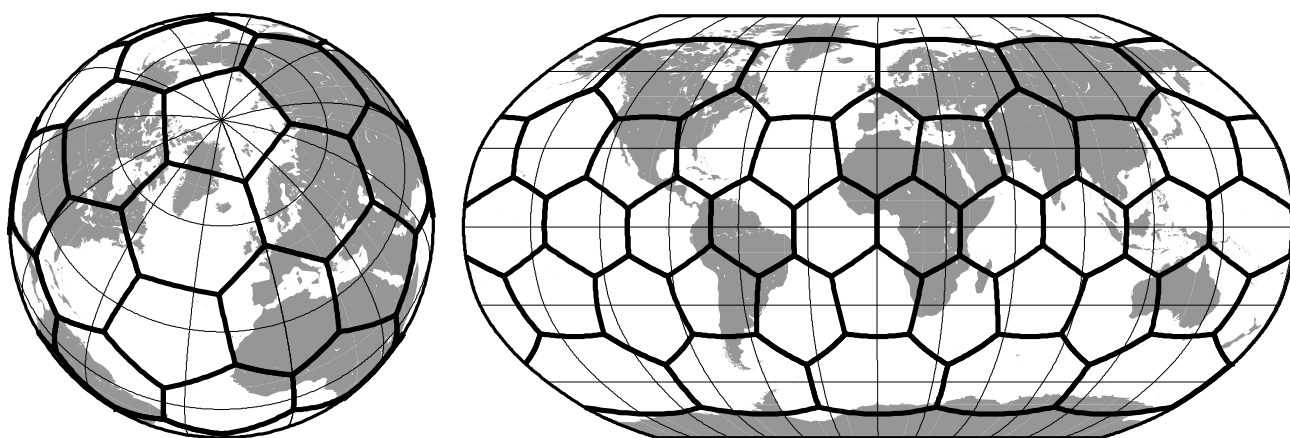


Figure B.2.: Orthographic and Robinson projection of the unstructured grid cells

Operator index

A

abs	54
acos	54
add	56
addc	55
asin	54
atan	54
atan2	56

C

cat	24
chcode	44
chlevel	44
chlevelc	44
chlevelv	44
chvar	44
const	117
copy	24
cos	54

D

dayavg	80
daymax	80
daymean	80
daymin	80
daypctl	81
daystd	80
daysum	80
dayvar	80
delcode	29
delvar	29
detrend	98
diff	19
diffv	19
div	56
divc	55
divdpm	58
divdpy	58
dv2uv	110
dv2uvl	110

E

eca_cdd	122
eca_cfd	122
eca_csu	123
eca_cwd	123
eca_cwdi	124
eca_cwfi	124

eca_etr	125
eca_fd	125
eca_fdns	126
eca_gsl	126
eca_hd	127
eca_hurr	137
eca_hwdi	127
eca_hwfi	128
eca_id	128
eca_r10mm	129
eca_r20mm	129
eca_r75p	130
eca_r75ptot	130
eca_r90p	131
eca_r90ptot	131
eca_r95p	132
eca_r95ptot	132
eca_r99p	133
eca_r99ptot	133
eca_rr1	134
eca_rx1day	134
eca_rx5day	135
eca_sdii	135
eca_strbre	136
eca_strgal	136
eca_strwin	136
eca_su	137
eca_tg10p	138
eca_tg90p	138
eca_tn10p	139
eca_tn90p	139
eca_tr	140
eca_tx10p	140
eca_tx90p	141
enlarge	50
ensavg	63
ensmax	63
ensmean	63
ensmin	63
enspctl	63
ensstd	63
enssum	63
ensvar	63
eq	38
eqc	39
exp	54
expr	53
exprf	53

F

fldavg	65
fldmax	65
fldmean	65
fldmin	65
fldpctl	65
fldstd	65
fldsum	65
fldvar	65

G

ge	38
gec	39
genbic	102
genbil	102
gencon	102
gendis	102
gp2sp	109
gp2spl	109
gradsdes1	115
gradsdes2	115
griddes	22
gt	38
gtc	39

H

hi	119
houravg	78
hourmax	78
hourmean	78
hourmin	78
hourpctl	79
hourstd	78
hoursum	78
hourvar	78

I

ifnotthen	35
ifnotthenc	36
ifthen	35
ifthenc	36
ifthenelse	35
info	17
infov	17
input	112
inputext	112
inputsrv	112
int	54
interpolate	104
intgridbil	104
intntime	106
inttime	106
intyear	107
invertlat	47
invertlatdata	47
invertlatdes	47
invertlon	47
invertlondata	47

invertlondes	47
--------------	----

L

le	38
lec	39
ln	54
log10	54
lt	38
ltc	39

M

map	17
maskindexbox	48
masklonlatbox	48
mastrfu	118
max	56
meravg	69
merge	25
mergetime	25
mermax	69
mermean	69
mermin	69
merpctl	69
merstd	69
mersum	69
mervar	69
min	56
ml2hl	105
ml2pl	105
monavg	82
monmax	82
monmean	82
monmin	82
monpctl	83
monstd	82
monsum	82
monvar	82
mul	56
mulc	55
muldpm	58
muldpy	58

N

ncode	20
ndate	20
ne	38
nec	39
nint	54
nlevel	20
nmon	20
ntime	20
nvar	20
nyear	20

O

output	113
outputext	113
outputf	113

outputint	113
outputsrv	113

R

random	117
remap	103
remapbic	101
remapbil	101
remapcon	101
remapdis	101
replace	24
rotuvb	118
runavg	74
runmax	74
runmean	74
runmin	74
runpctl	75
runstd	74
runsum	74
runvar	74

S

seasavg	86
seasmax	86
seasmean	86
seasmin	86
seaspctl	87
seasstd	86
seassum	86
seasvar	86
selavg	72
selcode	29
seldate	31
selday	31
selgrid	29
selgridname	29
selhour	31
selindexbox	33
sellevel	29
sellonlatbox	33
selmax	72
selmean	72
selmin	72
selmon	31
selpctl	73
selrec	29
selseas	31
selsmon	31
selstd	72
selstdname	29
selsum	72
seltabnum	29
selttime	31
seltimestep	31
selvar	29, 72
selyear	31
selzaxis	29
selzaxisname	29

setcalendar	42
setcindexbox	49
setclonlatbox	49
setcode	41
setctomiss	51
setdate	42
setday	42
setgatt	46
setgatts	46
setgrid	45
setgridtype	45
setlevel	41
setmisstoc	51
setmisval	51
setmon	42
setpartab	41
setreftime	42
setrtomiss	51
settaxis	42
settime	42
setunits	42
setvar	41
setyear	42
setzaxis	45
shifttime	42
showcode	21
showdate	21
showformat	21
showlevel	21
showmon	21
showstdname	21
showtime	21
showvar	21
showyear	21
sin	54
sinfo	18
sinfop	18
sinfov	18
sp2gp	109
sp2gpl	109
sp2sp	109
splitcode	26
splitday	27
splitgrid	26
splithour	27
splitlevel	26
splitmon	27
splitrec	26
splitseas	27
splitvar	26
splityear	27
splitzaxis	26
sqr	54
sqrt	54
sub	56
subc	55
subtrend	99

T

tan	54
timavg	76
timmax	76
timmean	76
timmin	76
timpctl	77
timsort	116
timstd	76
tisum	76
timvar	76
trend	99

U

uv2dv	110
uv2dvl	110

V

varides	22
vardup	117
varmul	117
vct	22
vertavg	71
vertmax	71
vertmean	71
vertmin	71
vertstd	71
vertsum	71
vertvar	71

W

wct	119
-----------	-----

Y

ydayavg	88
ydaymax	88
ydaymean	88
ydaymin	88
ydaypctl	89
ydaystd	88
ydaysum	88
ydayvar	88
ydrunavg	94
ydrunmax	94
ydrunmean	94
ydrunmin	94
ydrunpctl	96
ydrunstd	94
ydrunsum	94
ydrunvar	94
yearavg	84
yearmax	84
yearmean	84
yearmin	84
yearpctl	85
yearstd	84
yearsum	84
yearvar	84

ymonadd	57
ymonavg	90
ymonddiv	57
ymonmax	90
ymonmean	90
ymonmin	90
ymonmul	57
ymonpctl	91
ymonstd	90
ymonsub	57
ymonsum	90
ymonvar	90
yseasavg	92
yseasmax	92
yseasmean	92
yseasmin	92
yseaspctl	93
yseasstd	92
yseassum	92
yseasvar	92

Z

zonavg	67
zonmax	67
zonmean	67
zonmin	67
zonpctl	67
zonstd	67
zonsum	67
zonvar	67