CDO User's Guide

Climate Data Operators
Version 1.7.1
February 2016

Uwe Schulzweida – *MPI for Meteorology*

Contents

1. Introduction

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

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

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

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

The main **CDO** features are:

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

1.1. Building from sources

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

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

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

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

- PROJ.4 library (http://trac.osgeo.org/proj) version 4.6 or higher.
  This library is needed to convert Sinusoidal and Lambert Azimuthal Equal Area coordinates to geographic coordinates, for e.g. remapping.
- Magics library (https://software.ecmwf.int/wiki/display/MAGP/Magics) version 2.18 or higher.
  This library is needed to create contour, vector and graph plots with **CDO**.

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

1.1.1. Compilation

Compilation is done by performing the following steps:

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

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

2. Run the configure script:

   ```
   ./configure
   ```

   - Optionaly with NetCDF [NetCDF] support:

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

   - and with GRIB_API:

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

   For an overview of other configuration options use

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

3. Compile the program by running make:

   ```
   make
   ```

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

1.1.2. Installation

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

```
make install
```

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

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

1.2. Usage

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

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

1.2.1. Options

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

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

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

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

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

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

GRIB2 is only available if **CDO** was compiled with GRIB_API support and all NetCDF file types are only available if **CDO** was compiled with NetCDF support!

-g <grid>	Define the default grid description by name or from file (see chapter 1.3 on page 11). Available grid names are: `r<NX>x<NY>`, `lon=<LON>/lat=<LAT>`, `n<N>`, `gme<NI>`
-h, --help	Help information for the operators.
--no_history	Do not append to NetCDF *history* global attribute.
--netcdf_hdr_pad, --hdr_pad, --header_pad <nbr>	Pad NetCDF output header with *nbr* bytes.
-k <chunktype>	NetCDF4 chunk type: auto, grid or lines.
-L	Lock I/O (sequential access).
-M	Switch to indicate that the I/O streams have missing values.
-m <missval>	Set the default missing value (default: `-9e+33`).
-O	Overwrite existing output file, if checked. Existing output file is checked only for: `ens<STAT>`, `merge`, `mergetime`
--operators	List of all operators.
-P <nthreads>	Set number of OpenMP threads (Only available if OpenMP support was compiled in).
--percentile <method>	Percentile method: `nrank nist numpy numpy_lower numpy_higher numpy_nearest`
-Q	Alphanumeric sorting of NetCDF parameter names.
--reduce_dim	Reduce NetCDF dimensions (module: TIMSTAT, FLDSTAT).
-R, --regular	Convert GRIB1 data from reduced to regular grid (only with cgribex lib).
-r	Generate a relative time axis.
-S	Create an extra output stream for the module TIMSTAT. This stream contains the number of non missing values for each output period.
-s, --silent	Silent mode.
-t <partab>	Set the GRIB1 (cgribex) default parameter table name or file (see chapter 1.6 on page 15). Predefined tables are: `echam4 echam5 echam6 mpiom1 ecmwf remo`
-V, --version	Print the version number.
-v, --verbose	Print extra details for some operators.
-W	Print extra warning messages.
-z szip	SZIP compression of GRIB1 records.

jpeg	JPEG compression of GRIB2 records.
zip[_1-9]	Deflate compression of NetCDF4 variables.

1.2.2. Environment variables

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

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

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

1.2.3. Operators

There are more than 700 operators available. A detailed description of all operators can be found in the **Reference Manual** section.

1.2.4. Operator chaining

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

- reducing unnecessary disk I/O
- parallel processing

Use

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

instead of

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

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

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

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

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

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

1.2.5. Parallelized operators

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

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

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

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

1.2.6. Operator parameter

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

- STRING

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

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

- FLOAT

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

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

- INTEGER

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

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

  The result is the same as:

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

1.3. Horizontal grids

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

1.3.1. Grid area weights

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

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

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

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

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

1.3.2. Grid description

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

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

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

1.3.2.1. Predefined grids

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

Global regular grid: global_<DXY>

global_<DXY> defines a global regular lon/lat grid. The grid increment <DXY> can be selected at will. The longitudes start at <DXY>/2 - 180° and the latitudes start at <DXY>/2 - 90°.

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

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

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

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

Global Gaussian grid: n<N>

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

Global icosahedral-hexagonal GME grid: gme<NI>

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

1.3.2.2. Grids from data files

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

1.3.2.3. SCRIP grids

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

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

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

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

1.3.2.4. CDO grids

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

Keyword	Datatype	Description
gridtype	STRING	Type of the grid (gaussian, lonlat, curvilinear, unstructured).
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 cell center.
yvals	FLOAT ARRAY	Y values of the grid cell center.
xnpole	FLOAT	X value of the north pole (rotated grid).
ynpole	FLOAT	Y value of the north pole (rotated grid).
angle	FLOAT	Angle of the rotated north pole (default: 0).
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, xfirst is the x value of the first grid cell center.
yfirst, yinc	FLOAT, FLOAT	Macros to define yvals with a constant increment, yfirst is the y value of the first grid cell center.

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

gridtype	lonlat	gaussian	curvilinear	unstructured
gridsize	xsize*ysize	xsize*ysize	xsize*ysize	**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			
angle	0			
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 area weights are not needed. The grid cell corners xbounds and ybounds have to rotate counterclockwise.

CDO grid description example of a T21 gaussian grid:

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

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

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

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

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

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

Example **CDO** descriptions of a curvilinear and an unstructured grid can be found in Appendix C.

1.4. Z-axis description

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

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

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

Available z-axis types:

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

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

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

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

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

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

1.5. Time axis

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

1.5.1. Absolute time

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

1.5.2. Relative time

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

1.5.3. Conversion of the time

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

1.6. Parameter table

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

Example of a **CDO** parameter table:

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

1.7. Missing values

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

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

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		$miss$

sum	b		miss
a	$a+b$		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, treading them as not belonging to the sample, with the side-effect of a reduced sample size.

1.7.1. Mean and average

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

1.8. Percentile

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

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

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

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

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

1.8.1. Percentile over timesteps

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

2. Reference manual

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

$i(t)$ Timestep t of `ifile`

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

$o(t)$ Timestep t of `ofile`

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

2.1. Information

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

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

info	Dataset information listed by parameter identifier
infon	Dataset information listed by parameter name
map	Dataset information and simple map
sinfo	Short information listed by parameter identifier
sinfon	Short information listed by parameter name
diff	Compare two datasets listed by parameter id
diffn	Compare two datasets listed by parameter name
npar	Number of parameters
nlevel	Number of levels
nyear	Number of years
nmon	Number of months
ndate	Number of dates
ntime	Number of timesteps
showformat	Show file format
showcode	Show code numbers
showname	Show variable names
showstdname	Show standard names
showlevel	Show levels
showltype	Show GRIB level types
showyear	Show years
showmon	Show months
showdate	Show date information
showtime	Show time information
showtimestamp	Show timestamp
pardes	Parameter description
griddes	Grid description
zaxisdes	Z-axis 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 files to standard output. All input files need to have the same structure with the same variables on different timesteps. The information displayed depends on the chosen operator.

Operators

info Dataset information listed by parameter identifier
Prints information and simple statistics for each field of all input datasets. For each field the operator prints one line with the following elements:

- Date and Time
- Level, Gridsize and number of Missing values
- Minimum, Mean and Maximum
  The mean value is computed without the use of area weights!
- Parameter identifier

infon Dataset information listed by parameter name
The same as operator info but using the name instead of the identifier to label the parameter.

map Dataset information and simple map
Prints information, simple statistics and a map for each field of all input datasets. The map will be printed only for fields on a regular lon/lat grid.

Example

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

```
cdo infon ifile
```

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

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

2.1.2. SINFO - Short information

Synopsis

<*operator*> `ifiles`

Description

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

Operators

sinfo Short information listed by parameter identifier
Prints short information of a dataset. The information is divided into 4 sections. Section 1 prints one line per parameter with the following information:

- institute and source
- timestep type
- number of levels and z-axis number
- horizontal grid size and number
- data type
- parameter identifier

Section 2 and 3 gives a short overview of all grid and vertical coordinates. And the last section contains short information of the time coordinate.

sinfon Short information listed by parameter name
The same as operator sinfo but using the name instead of the identifier to label the parameter.

Example

To print short information of a dataset use:

```
cdo sinfon ifile
```

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

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

2.1.3. DIFF - Compare two datasets field by field

Synopsis

<operator> `ifile1 ifile2`

Description

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

Operators

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

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

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

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

diffn Compare two datasets listed by parameter name
The same as operator diff. Using the name instead of the identifier to label the parameter.

Example

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

```
cdo diffn ifile1 ifile2
```

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

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

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

Synopsis

<*operator*> `ifile`

Description

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

Operators

npar Number of parameters
Prints the number of parameters (variables).

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 timesteps
Prints the number of timesteps.

Example

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

```
cdo npar ifile
```

To print the number of months in a dataset use:

```
cdo nmon ifile
```

2.1.5. SHOWINFO - Show variables, levels or times

Synopsis

<*operator*> `ifile`

Description

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

Operators

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

Example

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

```
cdo showcode ifile
```

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

```
129 130 139
```

To print all months in a dataset use:

```
cdo showmon ifile
```

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

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

2.1.6. FILEDES - Dataset description

Synopsis

<*operator*> `ifile`

Description

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

Operators

pardes Parameter description
Prints a table with a description of all variables. For each variable the operator prints one line listing the code, name, description and units.

griddes Grid description
Prints the description of all grids.

zaxisdes Z-axis description
Prints the description of all z-axes.

vct Vertical coordinate table
Prints the vertical coordinate table.

Example

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

```
cdo griddes ifile
```

Result:

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

2.2. File operations

This section contains modules to perform operations on files.

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

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

2.2.1. COPY - Copy datasets

Synopsis

<*operator*> `ifiles ofile`

Description

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

Operators

copy Copy datasets
Copies all input datasets to `ofile`.

cat Concatenate datasets
Concatenates all input datasets and appends 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 timesteps of the same variables use:

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

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

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

2.2.2. REPLACE - Replace variables

Synopsis

replace `ifile1 ifile2 ofile`

Description

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

Example

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

```
cdo replace ifile1 ifile2 ofile
```

2.2.3. DUPLICATE - Duplicates a dataset

Synopsis

duplicate[*,ndup*] `ifile ofile`

Description

This operator duplicates the contents of `ifile` and writes the result to `ofile`. The optional parameter sets the number of duplicates, the default is 2.

Parameter

ndup	INTEGER	Number of duplicates, default is 2.

2.2.4. MERGEGRID - Merge grid

Synopsis

mergegrid `ifile1 ifile2 ofile`

Description

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

2.2.5. MERGE - Merge datasets

Synopsis

<*operator*> `ifiles ofile`

Description

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

Operators

merge Merge datasets with different fields
Merges time series of different fields from several input datasets. The number of fields per timestep written to `ofile` is the sum of the field numbers per timestep in all input datasets. The time series on all input datasets are required to have different fields and the same number of timesteps. The fields in each different input file either have to be different variables or different levels of the same variable. A mixture of different variables on different levels in different input files is not allowed.

mergetime Merge datasets sorted by date and time
Merges all timesteps of all input files sorted by date and time. All input files need to have the same structure with the same variables on different timesteps. After this operation every input timestep is in `ofile` and all timesteps are sorted by date and time.

Environment

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

Example

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

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

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

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

2.2.6. SPLIT - Split a dataset

Synopsis

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

Description

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

Operators

splitcode Split code numbers
Splits a dataset into pieces, one for each different code number. `xxx` will have three digits with the code number.

splitparam Split parameter identifiers
Splits a dataset into pieces, one for each different parameter identifier. `xxx` will be a string with the parameter identifier.

splitname Split variable names
Splits a dataset into pieces, one for each variable name. `xxx` will be a string with the variable name.

splitlevel Split levels
Splits a dataset into pieces, one for each different level. `xxx` will have six digits with the level.

splitgrid Split grids
Splits a dataset into pieces, one for each different grid. `xxx` will have two digits with the grid number.

splitzaxis Split z-axes
Splits a dataset into pieces, one for each different z-axis. `xxx` will have two digits with the z-axis number.

splittabnum Split parameter table numbers
Splits a dataset into pieces, one for each GRIB1 parameter table number. `xxx` will have three digits with the GRIB1 parameter table number.

Parameter

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

Environment

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

Example

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

```
cdo splitcode ifile code
```

Result of 'dir code*':

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

2.2.7. SPLITTIME - Split timesteps of a dataset

Synopsis

<*operator*> `ifile obase`

splitmon[*,format*] `ifile obase`

Description

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

Operators

splithour Split hours
Splits a file into pieces, one for each different hour. `xxx` will have two digits with the hour.

splitday Split days
Splits a file into pieces, one for each different day. `xxx` will have two digits with the day.

splitseas Split seasons
Splits a file into pieces, one for each different season. `xxx` will have three characters with the season.

splityear Split years
Splits a file into pieces, one for each different year. `xxx` will have four digits with the year (YYYY).

splityearmon Split in years and months
Splits a file into pieces, one for each different year and month. `xxx` will have six digits with the year and month (YYYYMM).

splitmon Split months
Splits a file into pieces, one for each different month. `xxx` will have two digits with the month.

Parameter

format STRING C-style format for strftime() (e.g. %B for the full month name)

Environment

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

Example

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

```
cdo splitmon ifile mon
```

Result of '`dir mon*`':

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

2.2.8. SPLITSEL - Split selected timesteps

Synopsis

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

Description

This operator splits `ifile` into pieces, one for each adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range. The output files will be named `<obase><nnnnnn><suffix>` where `nnnnnn` is the sequence number and `suffix` is the filename extension derived from the file format.

Parameter

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

Environment

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

2.2.9. DISTGRID - Distribute horizontal grid

Synopsis

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

Description

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

Parameter

nx	INTEGER	Number of regions in x direction
ny	INTEGER	Number of regions in y direction [default: 1]

Note

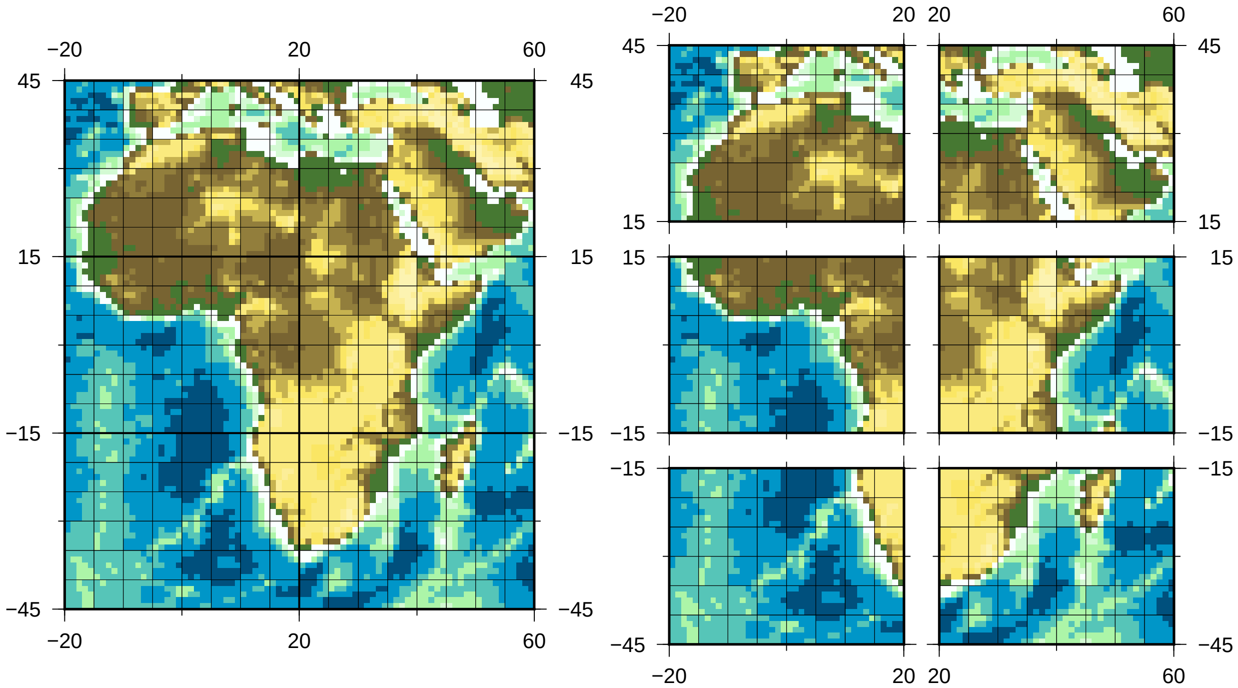
This operator needs to open all output files simultaneously. The maximum number of open files depends on the operating system!

Example

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

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

Below is a schematic illustration of this example:



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

2.2.10. COLLGRID - Collect horizontal grid

Synopsis

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

Description

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

Parameter

nx	INTEGER	Number of regions in x direction [default: number of input files]
names	STRING	Comma separated list of variable names [default: all variables]

Note

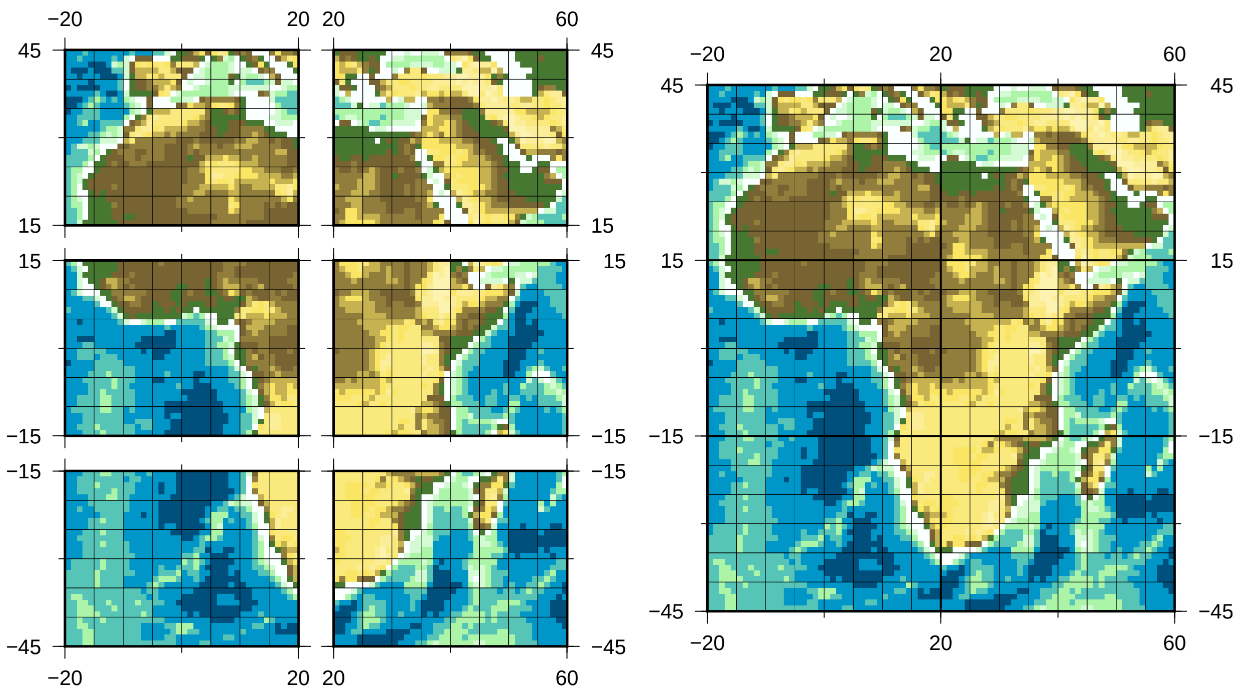
This operator needs to open all input files simultaneously. The maximum number of open files depends on the operating system!

Example

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

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

Below is a schematic illustration of this example:



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

2.3. Selection

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

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

select	Select fields
delete	Delete fields
selparam	Select parameters by identifier
delparam	Delete parameters by identifier
selcode	Select parameters by code number
delcode	Delete parameters by code number
selname	Select parameters by name
delname	Delete parameters by name
selstdname	Select parameters by standard name
sellevel	Select levels
sellevidx	Select levels by index
selgrid	Select grids
selzaxis	Select z-axes
selzaxisname	Select z-axes by name
selltype	Select GRIB level types
seltabnum	Select parameter table numbers
seltimestep	Select timesteps
seltime	Select times
selhour	Select hours
selday	Select days
selmonth	Select months
selyear	Select years
selseason	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

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

Description

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

Operators

select Select fields
Selects all fields with parameters in a user given list.

delete Delete fields
Deletes all fields with parameters in a user given list.

Parameter

name	STRING	Comma separated list of variable names.
param	STRING	Comma separated list of parameter identifiers.
code	INTEGER	Comma separated list of code numbers.
level	FLOAT	Comma separated list of vertical levels.
levidx	INTEGER	Comma separated list of index of levels.
zaxisname	STRING	Comma separated list of zaxis names.
zaxisnum	INTEGER	Comma separated list of zaxis numbers.
ltype	INTEGER	Comma separated list of GRIB level types.
gridname	STRING	Comma separated list of grid names.
gridnum	INTEGER	Comma separated list of grid numbers.
steptype	STRING	Comma separated list of timestep types.
date	STRING	Comma separated list of dates (format YYYY-MM-DDThh:mm:ss).
startdate	STRING	Start date (format YYYY-MM-DDThh:mm:ss).
enddate	STRING	End date (format YYYY-MM-DDThh:mm:ss).
minute	INTEGER	Comma separated list of minutes.
hour	INTEGER	Comma separated list of hours.
day	INTEGER	Comma separated list of days.
month	INTEGER	Comma separated list of months.
season	STRING	Comma separated list of seasons (substring of DJFMAMJJASOND or ANN).
year	INTEGER	Comma separated list of years.
timestep	INTEGER	Comma separated list of timesteps. Negative values selects timesteps from the end (NetCDF only).
timestep_of_year	INTEGER	Comma separated list of timesteps of year.

Example

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

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

2.3.2. SELVAR - Select fields

Synopsis

<*operator*>,*params* `ifile ofile`

selcode,*codes* `ifile ofile`

delcode,*codes* `ifile ofile`

selname,*names* `ifile ofile`

delname,*names* `ifile ofile`

selstdname,*stdnames* `ifile ofile`

sellevel,*levels* `ifile ofile`

sellevidx,*levidx* `ifile ofile`

selgrid,*grids* `ifile ofile`

selzaxis,*zaxes* `ifile ofile`

selzaxisname,*zaxisnames* `ifile ofile`

selltype,*ltypes* `ifile ofile`

seltabnum,*tabnums* `ifile ofile`

Description

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

Operators

selparam Select parameters by identifier
Selects all fields with parameter identifiers in a user given list.

delparam Delete parameters by identifier
Deletes all fields with parameter identifiers in a user given list.

selcode Select parameters by code number
Selects all fields with code numbers in a user given list.

delcode Delete parameters by code number
Deletes all fields with code numbers in a user given list.

selname Select parameters by name
Selects all fields with parameter names in a user given list.

delname Delete parameters by name
Deletes all fields with parameter names in a user given list.

selstdname Select parameters by standard name
Selects all fields with standard names in a user given list.

sellevel Select levels
Selects all fields with levels in a user given list.

sellevidx Select levels by index
Selects all fields with index of levels in a user given list.

selgrid Select grids
Selects all fields with grids in a user given list.

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

Parameter

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

Example

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

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

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

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

2.3.3. SELTIME - Select timesteps

Synopsis

seltimestep,*timesteps* `ifile ofile`

seltime,*times* `ifile ofile`

selhour,*hours* `ifile ofile`

selday,*days* `ifile ofile`

selmonth,*months* `ifile ofile`

selyear,*years* `ifile ofile`

selseason,*seasons* `ifile ofile`

seldate,*date1[,date2]* `ifile ofile`

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

Description

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

Operators

seltimestep	Select timesteps Selects all timesteps with a timestep in a user given list.
seltime	Select times Selects all timesteps with a time in a user given list.
selhour	Select hours Selects all timesteps with a hour in a user given list.
selday	Select days Selects all timesteps with a day in a user given list.
selmonth	Select months Selects all timesteps with a month in a user given list.
selyear	Select years Selects all timesteps with a year in a user given list.
selseason	Select seasons Selects all timesteps with a month of a season in a user given list.
seldate	Select dates Selects all timesteps with a date in a user given range.
selsmon	Select single month Selects a month and optional an arbitrary number of timesteps before and after this month.

Parameter

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

days	INTEGER	Comma separated list of days.
months	INTEGER	Comma separated list of months.
years	INTEGER	Comma separated list of years.
seasons	STRING	Comma separated list of seasons (substring of DJFMAMJJASOND or ANN).
date1	STRING	Start date (format YYYY-MM-DDThh:mm:ss).
date2	STRING	End date (format YYYY-MM-DDThh:mm:ss) [default: date1].
nts1	INTEGER	Number of timesteps before the selected month [default: 0].
nts2	INTEGER	Number of timesteps after the selected month [default: nts1].

2.3.4. SELBOX - Select a box of a field

Synopsis

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

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

Description

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

Operators

sellonlatbox Select a longitude/latitude box
Selects a regular longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box. Considered are only those grid cells with the grid center inside the lon/lat box. For rotated lon/lat grids the parameter needs to be rotated coordinates.

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

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

Example

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

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

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

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

2.4. Conditional selection

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

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

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

Operators

ifthen If then

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

ifnotthen If not then

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

Example

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

```
cdo ifthen ifile1 ifile2 ofile
```

2.4.2. COND2 - Conditional select two fields

Synopsis

ifthenelse `ifile1 ifile2 ifile3 ofile`

Description

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

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

Example

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

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

2.4.3. CONDC - Conditional select a constant

Synopsis

<operator>,*c* `ifile ofile`

Description

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

Operators

ifthenc If then constant

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

ifnotthenc If not then constant

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

Parameter

c	FLOAT	Constant

Example

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

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

2.5. Comparison

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

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

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

2.5.1. COMP - Comparison of two fields

Synopsis

<*operator*> `ifile1 ifile2 ofile`

Description

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

Operators

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 containing 1 if the elements of two fields are the same and 0 if the elements are different use:

```
cdo eq ifile1 ifile2 ofile
```

2.5.2. COMPC - Comparison of a field with a constant

Synopsis

<operator>,c `ifile ofile`

Description

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

Operators

eqc Equal constant

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

nec Not equal constant

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

lec Less equal constant

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

ltc Less than constant

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

gec Greater equal constant

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

gtc Greater than constant

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

Parameter

c	FLOAT	Constant

Example

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

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

2.6. Modification

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

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

setpartabp	Set parameter table
setpartabn	Set parameter table
setpartab	Set parameter table
setcode	Set code number
setparam	Set parameter identifier
setname	Set variable name
setunit	Set variable unit
setlevel	Set level
setltype	Set GRIB level type
setdate	Set date
settime	Set time of the day
setday	Set day
setmon	Set month
setyear	Set year
settunits	Set time units
settaxis	Set time axis
setreftime	Set reference time
setcalendar	Set calendar
shifttime	Shift timesteps
chcode	Change code number
chparam	Change parameter identifier
chname	Change variable name
chunit	Change variable unit
chlevel	Change level
chlevelc	Change level of one code
chlevelv	Change level of one variable
setgrid	Set grid
setgridtype	Set grid type
setgridarea	Set grid cell area
setzaxis	Set z-axis
genlevelbounds	Generate level bounds
setgatt	Set global attribute
setgatts	Set global attributes
invertlat	Invert latitudes
invertlev	Invert levels
maskregion	Mask regions
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
setvrange	Set valid range
setmisstonn	Set missing value to nearest neighbor
setmisstodis	Set missing value to distance-weighted average

2.6.1. SETPARTAB - Set parameter table

Synopsis

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

Description

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

The following parameter table entries are supported:

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

The search key for the variable depends on the operator. Use setpartabn to search variables by the name. This is typically used for NetCDF datasets. The operator setpartabp searches variables by the parameter ID.

Operators

setpartabp Set parameter table
Search variables by the parameter identifier.

setpartabn Set parameter table
Search variables by name.

Parameter

table STRING Parameter table file or name

convert STRING Converts the units if necessary

Example

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

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

To apply this parameter table to a dataset use:

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

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

2.6.2. SET - Set field info

Synopsis

setpartab,*table* `ifile ofile`

setcode,*code* `ifile ofile`

setparam,*param* `ifile ofile`

setname,*name* `ifile ofile`

setunit,*unit* `ifile ofile`

setlevel,*level* `ifile ofile`

setltype,*ltype* `ifile ofile`

Description

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

Operators

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.

setparam Set parameter identifier
Sets the parameter identifier of the first variable.

setname Set variable name
Sets the name of the first variable.

setunit Set variable unit
Sets the unit of the first variable.

setlevel Set level
Sets the first level of all variables.

setltype Set GRIB level type
Sets the GRIB level type of all variables.

Parameter

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

2.6.3. SETTIME - Set time

Synopsis

setdate,*date* `ifile ofile`

settime,*time* `ifile ofile`

setday,*day* `ifile ofile`

setmon,*month* `ifile ofile`

setyear,*year* `ifile ofile`

settunits,*units* `ifile ofile`

settaxis,*date,time[,inc]* `ifile ofile`

setreftime,*date,time[,units]* `ifile ofile`

setcalendar,*calendar* `ifile ofile`

shifttime,*sval* `ifile ofile`

Description

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

Operators

setdate	Set date Sets the date in every timestep to the same given value.
settime	Set time of the day Sets the time in every timestep to the same given value.
setday	Set day Sets the day in every timestep to the same given value.
setmon	Set month Sets the month in every timestep to the same given value.
setyear	Set year Sets the year in every timestep 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 a relative time axis.
setcalendar	Set calendar Sets the calendar of a relative time axis.
shifttime	Shift timesteps Shifts all timesteps by the parameter sval.

Parameter

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

Example

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

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

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

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

To shift this time axis by -15 days use:

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

Result of 'cdo showdate ofile':

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

2.6.4. CHANGE - Change field header

Synopsis

chcode,*oldcode,newcode[,...]* `ifile ofile`

chparam,*oldparam,newparam,...* `ifile ofile`

chname,*oldname,newname,...* `ifile ofile`

chunit,*oldunit,newunit,...* `ifile ofile`

chlevel,*oldlev,newlev,...* `ifile ofile`

chlevelc,*code,oldlev,newlev* `ifile ofile`

chlevelv,*name,oldlev,newlev* `ifile ofile`

Description

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

Operators

chcode	Change code number Changes some user given code numbers to new user given values.
chparam	Change parameter identifier Changes some user given parameter identifiers to new user given values.
chname	Change variable name Changes some user given variable names to new user given names.
chunit	Change variable unit Changes some user given variable units to new user given units.
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 name.

Parameter

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

Example

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

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

2.6.5. SETGRID - Set grid information

Synopsis

setgrid,*grid* `ifile ofile`

setgridtype,*gridtype* `ifile ofile`

setgridarea,*gridarea* `ifile ofile`

Description

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

Operators

setgrid Set grid
Sets a new grid description. The input fields need to have the same grid size as the size of the target grid description.

setgridtype Set grid type
Sets the grid type of all input fields. The following grid types are available:

curvilinear	Converts a regular grid to a curvilinear grid
unstructured	Converts a regular or curvilinear grid to an unstructured grid
dereference	Dereference a reference to a grid
regular	Converts a reduced Gaussian grid to a regular Gaussian grid
lonlat	Converts a regular lonlat grid stored as a curvilinear grid back to a lonlat grid

setgridarea Set grid cell area
Sets the grid cell area. The parameter *gridarea* is the path to a data file, the first field is used as grid cell area. The input fields need to have the same grid size as the grid cell area. The grid cell area is used to compute the weights of each grid cell if needed by an operator, e.g. for fldmean.

Parameter

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

Example

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

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

2.6.6. SETZAXIS - Set z-axis information

Synopsis

setzaxis,*zaxis* `ifile ofile`

genlevelbounds*[,zbot[,ztop]]* `ifile ofile`

Description

This module modifies the metadata of the vertical grid.

Operators

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

Parameter

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

2.6.7. SETGATT - Set global attribute

Synopsis

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

setgatts,*attfile* `ifile ofile`

Description

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

Operators

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

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

Note

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

Example

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

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

Result of 'ncdump -h ofile':

```
netcdf ofile {
dimensions: ...

variables: ...

// global attributes:
                :myatt = "myattcontents" ;
}
```

2.6.8. INVERT - Invert latitudes

Synopsis

invertlat `ifile ofile`

Description

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

Example

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

```
cdo invertlat ifile ofile
```

2.6.9. INVERTLEV - Invert levels

Synopsis

invertlev `ifile ofile`

Description

This operator inverts the levels of all 3D variables.

2.6.10. MASKREGION - Mask regions

Synopsis

maskregion,*regions* `ifile ofile`

Description

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

Parameter

regions	STRING	Comma separated list of ASCII formatted files with different regions

Example

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

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

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

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

2.6.11. MASKBOX - Mask a box

Synopsis

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

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

Description

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

Operators

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

Parameter

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

Example

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

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

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

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

2.6.12. SETBOX - Set a box to constant

Synopsis

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

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

Description

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

Operators

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 can be greater than the one of the right edge.

Parameter

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

Example

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

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

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

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

2.6.13. ENLARGE - Enlarge fields

Synopsis

enlarge,*grid* `ifile ofile`

Description

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

Parameter

grid	STRING	Target grid description file or name

Example

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

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

Or shorter using operator piping:

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

2.6.14. SETMISS - Set missing value

Synopsis

setmissval,*newmiss* `ifile ofile`

setctomiss,*c* `ifile ofile`

setmisstoc,*c* `ifile ofile`

setrtomiss,*rmin*,*rmax* `ifile ofile`

setvrange,*rmin*,*rmax* `ifile ofile`

setmisstonn `ifile ofile`

setmisstodis[,*neighbors*] `ifile ofile`

Description

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

Operators

setmissval Set a new missing value

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

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

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

setvrange Set valid range

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

setmisstonn Set missing value to nearest neighbor
Set all missing values to the nearest non missing value.

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

setmisstodis Set missing value to distance-weighted average
Set all missing values to the distance-weighted average of the nearest non missing values. The default number of nearest neighbors is 4.

Parameter

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

Example

setrtomiss

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

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

Result of 'cdo info ifile':

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

Result of 'cdo info ofile':

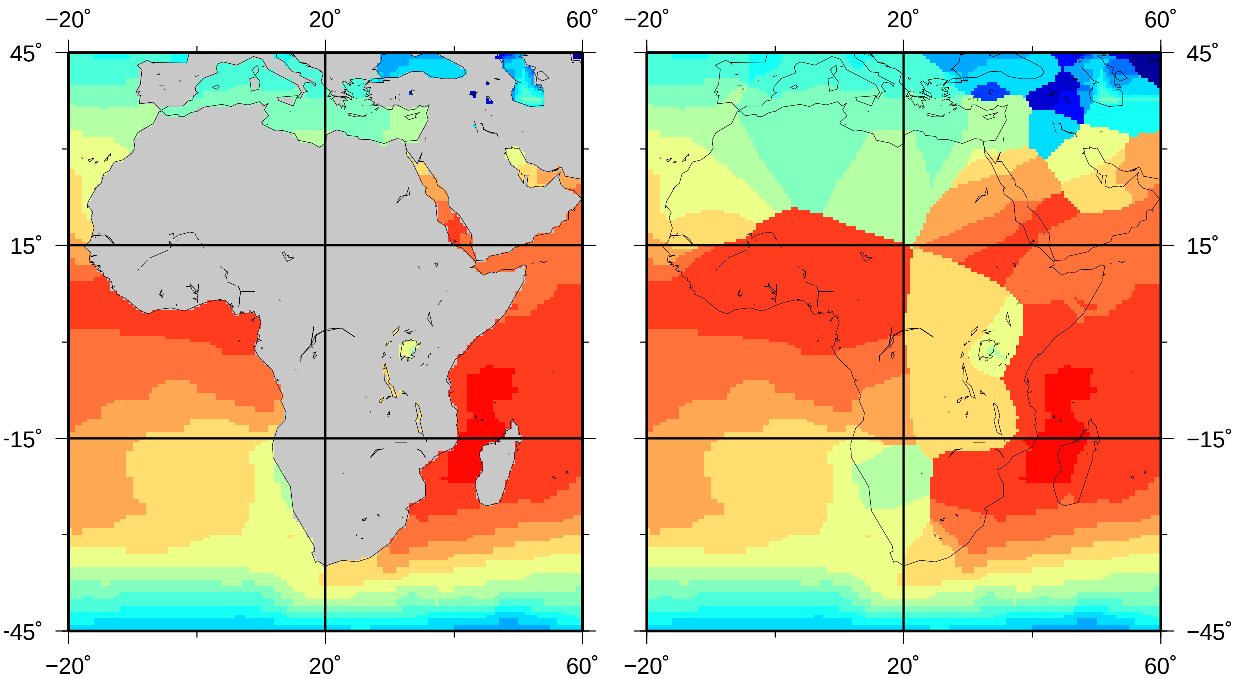
```
 -1 :       Date     Time   Code  Level   Size    Miss :   Minimum      Mean   Maximum
  1 : 1987-12-31 12:00:00  139       0   2048     871 :    273.16    287.08    303.71
```

setmisstonn

Set all missing values to the nearest non missing value:

```
cdo setmisstonn ifile ofile
```

Below is a schematic illustration of this example:



On the left side is input data with missing values in grey and on the right side the result with the filled missing values.

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 script
aexpr	Evaluate expressions and append results
aexprf	Evaluate expression script and append results
abs	Absolute value
int	Integer value
nint	Nearest integer value
pow	Power
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
reci	Reciprocal value
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
monadd	Add monthly time series
monsub	Subtract monthly time series
monmul	Multiply monthly time series
mondiv	Divide monthly time series
yhouradd	Add multi-year hourly time series
yhoursub	Subtract multi-year hourly time series
yhourmul	Multiply multi-year hourly time series
yhourdiv	Divide multi-year hourly time series
ydayadd	Add multi-year daily time series
ydaysub	Subtract multi-year daily time series
ydaymul	Multiply multi-year daily time series
ydaydiv	Divide multi-year daily time series

ymonadd	Add multi-year monthly time series
ymonsub	Subtract multi-year monthly time series
ymonmul	Multiply multi-year monthly time series
ymondiv	Divide multi-year monthly time series
yseasadd	Add multi-year seasonal time series
yseassub	Subtract multi-year seasonal time series
yseasmul	Multiply multi-year seasonal time series
yseasdiv	Divide multi-year seasonal time series
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`

aexpr,*instr* `ifile ofile`

aexprf,*filename* `ifile ofile`

Description

This module arithmetically processes every timestep of the input dataset. Each individual assignment statement have to end with a semi-colon. Unlike regular variables, temporary variables are never written to the output stream. To define a temporary variable simply prefix the variable name with an underscore (e.g. _varname) when the variable is declared.

The following operators are supported:

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

The following functions are supported:

Math intrinsics:

abs(x)	Absolute value of x
floor(x)	Round to largest integral value not greater than x
ceil(x)	Round to smallest integral value not less than x
int(x)	Integer value of x
nint(x)	Nearest integer value of x
sqr(x)	Square of x
sqrt(x)	Square Root of x
exp(x)	Exponential of x
log(x)	Natural logarithm of x
log10(x)	Base 10 logarithm of x

sin(x)	Sine of x, where x is specified in radians
cos(x)	Cosine of x, where x is specified in radians
tan(x)	Tangent of x, where x is specified in radians
asin(x)	Arc-sine of x, where x is specified in radians
acos(x)	Arc-cosine of x, where x is specified in radians
atan(x)	Arc-tangent of x, where x is specified in radians
rad(x)	Convert x from degrees to radians
deg(x)	Convert x from radians to degrees

Coordinates:

clon(x)	Longitude coordinate of x (available only if x has geographical coordinates)
clat(x)	Latitude coordinate of x (available only if x has geographical coordinates)
clev(x)	Level coordinate of x (0, if x is a 2D surface variable)

Constants:

ngp(x)	Number of horizontal grid points
nlev(x)	Number of vertical levels
size(x)	Total number of elements (ngp(x)*nlev(x))
missval(x)	Returns the missing value of variable x

Statistical values over a field:

fldmin(x), fldmax(x), fldsum(x), fldmean(x), fldavg(x), fldstd(x), fldstd1(x), fldvar(x), fldvar1(x)

Vertical statistical values:

vertmin(x), vertmax(x), vertsum(x), vertmean(x), vertavg(x), vertstd(x), vertstd1(x), vertvar(x), vertvar1(x)

Miscellaneous:

sellevel(x,k)	Select level k of variable x
sellevidx(x,k)	Select level index k of variable x
remove(x)	Remove variable x from output stream

Operators

expr Evaluate expressions
The processing instructions are read from the parameter.

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

aexpr Evaluate expressions and append results
Same as expr, but keep input variables and append results

aexprf Evaluate expression script and append results
Same as exprf, but keep input variables and append results

Parameter

instr	STRING	Processing instructions (need to be 'quoted' in most cases)
filename	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
$o(t,x) = \mathrm{abs}(i(t,x))$

int Integer value
$o(t,x) = \mathrm{int}(i(t,x))$

nint Nearest integer value
$o(t,x) = \mathrm{nint}(i(t,x))$

pow Power
$o(t,x) = i(t,x)^y$

sqr Square
$o(t,x) = i(t,x)^2$

sqrt Square root
$o(t,x) = \sqrt{i(t,x)}$

exp Exponential
$o(t,x) = \mathrm{e}^{i(t,x)}$

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
$o(t,x) = \arctan(i(t,x))$

reci Reciprocal value
$o(t,x) = 1/i(t,x)$

Example

To calculate the square root for all field elements use:

```
cdo sqrt ifile ofile
```

2.7.3. ARITHC - Arithmetic with a constant

Synopsis

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

Description

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

Operators

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

Operators

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 $atan2$ operator calculates the arc tangent of two fields. The result is in radians, which is between -PI and PI (inclusive).

$o(t,x) = \mathrm{atan2}(i_1(t,x), i_2(t,x))$

Example

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

```
cdo add ifile1 ifile2 ofile
```

2.7.5. MONARITH - Monthly arithmetic

Synopsis

<*operator*> `ifile1 ifile2 ofile`

Description

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

Operators

monadd Add monthly time series
Adds a time series and a monthly time series.

monsub Subtract monthly time series
Subtracts a time series and a monthly time series.

monmul Multiply monthly time series
Multiplies a time series and a monthly time series.

mondiv Divide monthly time series
Divides a time series and a monthly time series.

Example

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

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

2.7.6. YHOURARITH - Multi-year hourly arithmetic

Synopsis

<*operator*> `ifile1 ifile2 ofile`

Description

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

Operators

yhouradd Add multi-year hourly time series
Adds a time series and a multi-year hourly time series.

yhoursub Subtract multi-year hourly time series
Subtracts a time series and a multi-year hourly time series.

yhourmul Multiply multi-year hourly time series
Multiplies a time series and a multi-year hourly time series.

yhourdiv Divide multi-year hourly time series
Divides a time series and a multi-year hourly time series.

Example

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

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

2.7.7. YDAYARITH - Multi-year daily arithmetic

Synopsis

<*operator*> `ifile1 ifile2 ofile`

Description

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

Operators

ydayadd Add multi-year daily time series
Adds a time series and a multi-year daily time series.

ydaysub Subtract multi-year daily time series
Subtracts a time series and a multi-year daily time series.

ydaymul Multiply multi-year daily time series
Multiplies a time series and a multi-year daily time series.

ydaydiv Divide multi-year daily time series
Divides a time series and a multi-year daily time series.

Example

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

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

2.7.8. YMONARITH - Multi-year monthly arithmetic

Synopsis

<operator> `ifile1 ifile2 ofile`

Description

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

Operators

ymonadd Add multi-year monthly time series
Adds a time series and a multi-year monthly time series.

ymonsub Subtract multi-year monthly time series
Subtracts a time series and a multi-year monthly time series.

ymonmul Multiply multi-year monthly time series
Multiplies a time series and a multi-year monthly time series.

ymondiv Divide multi-year monthly time series
Divides a time series and a multi-year monthly time series.

Example

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

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

2.7.9. YSEASARITH - Multi-year seasonal arithmetic

Synopsis

<operator> `ifile1 ifile2 ofile`

Description

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

Operators

yseasadd Add multi-year seasonal time series
Adds a time series and a multi-year seasonal time series.

yseassub Subtract multi-year seasonal time series
Subtracts a time series and a multi-year seasonal time series.

yseasmul Multiply multi-year seasonal time series
Multiplies a time series and a multi-year seasonal time series.

yseasdiv Divide multi-year seasonal time series
Divides a time series and a multi-year seasonal time series.

Example

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

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

2.7.10. ARITHDAYS - Arithmetic with days

Synopsis

<operator> `ifile ofile`

Description

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

Operators

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
$o(t,x) = i(t,x)/days\_per\_year$

2.8. Statistical values

This section contains modules to compute statistical values of datasets. In this program there is the different notion of "mean" and "average" to distinguish two different kinds of treatment of missing values. While computing the mean, only the not missing values are considered to belong to the sample with the side effect of a probably reduced sample size. Computing the average is just adding the sample members and divide the result by the sample size. For example, the mean of 1, 2, miss and 3 is (1+2+3)/3 = 2, whereas the average is (1+2+miss+3)/4 = miss/4 = miss. If there are no missing values in the sample, the average and the mean are identical.
This program is using the verification time to identify the time range for time-statistics. The time bounds are never used!

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

sum	$\sum_{i=1}^{n} x_i$
mean resp. **avg**	$n^{-1} \sum_{i=1}^{n} x_i$
mean resp. **avg** weighted by $\{w_i, i = 1, ..., n\}$	$\left(\sum_{j=1}^{n} w_j\right)^{-1} \sum_{i=1}^{n} w_i\, x_i$
Variance **var**	$n^{-1} \sum_{i=1}^{n} (x_i - \overline{x})^2$
var1	$(n-1)^{-1} \sum_{i=1}^{n} (x_i - \overline{x})^2$
var weighted by $\{w_i, i = 1, ..., n\}$	$\left(\sum_{j=1}^{n} w_j\right)^{-1} \sum_{i=1}^{n} w_i \left(x_i - \left(\sum_{j=1}^{n} w_j\right)^{-1} \sum_{j=1}^{n} w_j\, x_j\right)^2$
Standard deviation **std**	$\sqrt{n^{-1} \sum_{i=1}^{n} (x_i - \overline{x})^2}$
std1	$\sqrt{(n-1)^{-1} \sum_{i=1}^{n} (x_i - \overline{x})^2}$
std weighted by $\{w_i, i = 1, ..., n\}$	$\sqrt{\left(\sum_{j=1}^{n} w_j\right)^{-1} \sum_{i=1}^{n} w_i \left(x_i - \left(\sum_{j=1}^{n} w_j\right)^{-1} \sum_{j=1}^{n} w_j\, x_j\right)^2}$
Cumulative Ranked Probability Score **crps**	$\int_{-\infty}^{\infty} \left[H(x_1) - \mathit{cdf}(\{x_2 \ldots x_n\})\|_r\right]^2 dr$

with $\mathit{cdf}(X)|_r$ being the cumulative distribution function of $\{x_i, i = 2 \ldots n\}$ at r

and $H(x)$ the Heavyside function jumping at x.

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

consecsum	Consecutive Sum
consects	Consecutive Timesteps
ensmin	Ensemble minimum
ensmax	Ensemble maximum
enssum	Ensemble sum
ensmean	Ensemble mean
ensavg	Ensemble average
ensstd	Ensemble standard deviation
ensstd1	Ensemble standard deviation (n-1)
ensvar	Ensemble variance
ensvar1	Ensemble variance (n-1)
enspctl	Ensemble percentiles
ensrkhistspace	Ranked Histogram averaged over time
ensrkhisttime	Ranked Histogram averaged over space
ensroc	Ensemble Receiver Operating characteristics
enscrps	Ensemble CRPS and decomposition
ensbrs	Ensemble Brier score
fldmin	Field minimum
fldmax	Field maximum
fldsum	Field sum
fldmean	Field mean
fldavg	Field average
fldstd	Field standard deviation
fldstd1	Field standard deviation (n-1)
fldvar	Field variance
fldvar1	Field variance (n-1)
fldpctl	Field percentiles
zonmin	Zonal minimum
zonmax	Zonal maximum
zonsum	Zonal sum
zonmean	Zonal mean
zonavg	Zonal average
zonstd	Zonal standard deviation
zonstd1	Zonal standard deviation (n-1)
zonvar	Zonal variance
zonvar1	Zonal variance (n-1)
zonpctl	Zonal percentiles
mermin	Meridional minimum
mermax	Meridional maximum
mersum	Meridional sum
mermean	Meridional mean
meravg	Meridional average
merstd	Meridional standard deviation
merstd1	Meridional standard deviation (n-1)
mervar	Meridional variance
mervar1	Meridional variance (n-1)
merpctl	Meridional percentiles

gridboxmin	Gridbox minimum
gridboxmax	Gridbox maximum
gridboxsum	Gridbox sum
gridboxmean	Gridbox mean
gridboxavg	Gridbox average
gridboxstd	Gridbox standard deviation
gridboxstd1	Gridbox standard deviation (n-1)
gridboxvar	Gridbox variance
gridboxvar1	Gridbox variance (n-1)
vertmin	Vertical minimum
vertmax	Vertical maximum
vertsum	Vertical sum
vertmean	Vertical mean
vertavg	Vertical average
vertstd	Vertical standard deviation
vertstd1	Vertical standard deviation (n-1)
vertvar	Vertical variance
vertvar1	Vertical variance (n-1)
timselmin	Time range minimum
timselmax	Time range maximum
timselsum	Time range sum
timselmean	Time range mean
timselavg	Time range average
timselstd	Time range standard deviation
timselstd1	Time range standard deviation (n-1)
timselvar	Time range variance
timselvar1	Time range variance (n-1)
timselpctl	Time range percentiles
runmin	Running minimum
runmax	Running maximum
runsum	Running sum
runmean	Running mean
runavg	Running average
runstd	Running standard deviation
runstd1	Running standard deviation (n-1)
runvar	Running variance
runvar1	Running variance (n-1)
runpctl	Running percentiles
timmin	Time minimum
timmax	Time maximum
timsum	Time sum
timmean	Time mean
timavg	Time average
timstd	Time standard deviation
timstd1	Time standard deviation (n-1)
timvar	Time variance
timvar1	Time variance (n-1)
timpctl	Time percentiles

hourmin	Hourly minimum
hourmax	Hourly maximum
hoursum	Hourly sum
hourmean	Hourly mean
houravg	Hourly average
hourstd	Hourly standard deviation
hourstd1	Hourly standard deviation (n-1)
hourvar	Hourly variance
hourvar1	Hourly variance (n-1)
hourpctl	Hourly percentiles
daymin	Daily minimum
daymax	Daily maximum
daysum	Daily sum
daymean	Daily mean
dayavg	Daily average
daystd	Daily standard deviation
daystd1	Daily standard deviation (n-1)
dayvar	Daily variance
dayvar1	Daily variance (n-1)
daypctl	Daily percentiles
monmin	Monthly minimum
monmax	Monthly maximum
monsum	Monthly sum
monmean	Monthly mean
monavg	Monthly average
monstd	Monthly standard deviation
monstd1	Monthly standard deviation (n-1)
monvar	Monthly variance
monvar1	Monthly variance (n-1)
monpctl	Monthly percentiles
yearmonmean	Yearly mean from monthly data
yearmin	Yearly minimum
yearmax	Yearly maximum
yearsum	Yearly sum
yearmean	Yearly mean
yearavg	Yearly average
yearstd	Yearly standard deviation
yearstd1	Yearly standard deviation (n-1)
yearvar	Yearly variance
yearvar1	Yearly variance (n-1)
yearpctl	Yearly percentiles

seasmin	Seasonal minimum
seasmax	Seasonal maximum
seassum	Seasonal sum
seasmean	Seasonal mean
seasavg	Seasonal average
seasstd	Seasonal standard deviation
seasstd1	Seasonal standard deviation (n-1)
seasvar	Seasonal variance
seasvar1	Seasonal variance (n-1)
seaspctl	Seasonal percentiles
yhourmin	Multi-year hourly minimum
yhourmax	Multi-year hourly maximum
yhoursum	Multi-year hourly sum
yhourmean	Multi-year hourly mean
yhouravg	Multi-year hourly average
yhourstd	Multi-year hourly standard deviation
yhourstd1	Multi-year hourly standard deviation (n-1)
yhourvar	Multi-year hourly variance
yhourvar1	Multi-year hourly variance (n-1)
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
ydaystd	Multi-year daily standard deviation
ydaystd1	Multi-year daily standard deviation (n-1)
ydayvar	Multi-year daily variance
ydayvar1	Multi-year daily variance (n-1)
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
ymonstd	Multi-year monthly standard deviation
ymonstd1	Multi-year monthly standard deviation (n-1)
ymonvar	Multi-year monthly variance
ymonvar1	Multi-year monthly variance (n-1)
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
yseasstd	Multi-year seasonal standard deviation
yseasstd1	Multi-year seasonal standard deviation (n-1)
yseasvar	Multi-year seasonal variance
yseasvar1	Multi-year seasonal variance (n-1)

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
ydrunstd	Multi-year daily running standard deviation
ydrunstd1	Multi-year daily running standard deviation (n-1)
ydrunvar	Multi-year daily running variance
ydrunvar1	Multi-year daily running variance (n-1)
ydrunpctl	Multi-year daily running percentiles

2.8.1. CONSECSTAT - Consecute timestep periods

Synopsis

<operator> `ifile ofile`

Description

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

Operators

consecsum Consecutive Sum
This operator computes periods of consecutive timesteps similar to a runsum, but periods are finished, when the mask value is 0. That way multiple periods can be found. Timesteps from the input are preserved. Missing values are handled like 0, i.e. finish periods of consecutive timesteps.

consects Consecutive Timesteps
In contrast to the operator above consects only computes the lenght of each period together with its last timestep. To be able to perform statistical analysis like min, max or mean, everything else is set to missing value.

Example

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

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

2.8.2. ENSSTAT - Statistical values over an ensemble

Synopsis

<*operator*> `ifiles ofile`

enspctl,*p* `ifiles ofile`

Description

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

Operators

ensmin Ensemble minimum
$o(t,x) = \mathbf{min}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

ensmax Ensemble maximum
$o(t,x) = \mathbf{max}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

enssum Ensemble sum
$o(t,x) = \mathbf{sum}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

ensmean Ensemble mean
$o(t,x) = \mathbf{mean}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

ensavg Ensemble average
$o(t,x) = \mathbf{avg}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

ensstd Ensemble standard deviation
Normalize by n.

$o(t,x) = \mathbf{std}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

ensstd1 Ensemble standard deviation (n-1)
Normalize by (n-1).

$o(t,x) = \mathbf{std1}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

ensvar Ensemble variance
Normalize by n.

$o(t,x) = \mathbf{var}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

ensvar1 Ensemble variance (n-1)
Normalize by (n-1).

$o(t,x) = \mathbf{var1}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

enspctl Ensemble percentiles
$o(t,x) = \mathbf{pth\ percentile}\{i_1(t,x), i_2(t,x), \cdots, i_n(t,x)\}$

Parameter

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

Example

To compute the ensemble mean over 6 input files use:

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

Or shorter with filename substitution:

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

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

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

2.8.3. ENSSTAT2 - Statistical values over an ensemble

Synopsis

<*operator*> `obsfile ensfiles ofile`

Description

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

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

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

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

Operators

ensrkhistspace	Ranked Histogram averaged over time
ensrkhisttime	Ranked Histogram averaged over space
ensroc	Ensemble Receiver Operating characteristics

Example

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

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

Or shorter with filename substitution:

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

2.8.4. ENSVAL - Ensemble validation tools

Synopsis

enscrps `rfile ifiles ofilebase`

ensbrs,*x* `rfile ifiles ofilebase`

Description

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

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

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

holds.

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

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

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

Operators

enscrps	Ensemble CRPS and decomposition
ensbrs	Ensemble Brier score Ensemble Brier Score and Decomposition

Example

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

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

or shorter using file name substitution:

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

2.8.5. FLDSTAT - Statistical values over a field

Synopsis

<*operator*> `ifile ofile`

fldpctl,p `ifile ofile`

Description

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

Operators

fldmin Field minimum
For every gridpoint $x\_1, ..., x\_n$ 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 $x\_1, ..., x\_n$ 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 $x\_1, ..., x\_n$ 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 $x\_1, ..., x\_n$ 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 $x\_1, ..., x\_n$ 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.

fldstd Field standard deviation
Normalize by n. For every gridpoint $x\_1, ..., x\_n$ 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.

fldstd1 Field standard deviation (n-1)
Normalize by (n-1). For every gridpoint $x\_1, ..., x\_n$ of the same field it is:
$o(t, 1) = \mathbf{std1}\{i(t, x'), x_1 < x' \leq x_n\}$

weighted by area weights obtained by the input field.

fldvar Field variance
Normalize by n. For every gridpoint $x\_1, ..., x\_n$ 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.

fldvar1 Field variance (n-1)
Normalize by (n-1). For every gridpoint $x\_1, ..., x\_n$ of the same field it is:
$o(t, 1) = \mathbf{var1}\{i(t, x'), x_1 < x' \leq x_n\}$

weighted by area weights obtained by the input field.

fldpctl Field percentiles
For every gridpoint $x\_1, ..., x\_n$ of the same field it is:
$o(t, 1) = \mathbf{pth\ percentile}\{i(t, x'), x_1 < x' \leq x_n\}$

Parameter

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

Example

To compute the field mean of all input fields use:

```
cdo fldmean ifile ofile
```

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

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

2.8.6. ZONSTAT - Zonal statistical values

Synopsis

<operator> `ifile ofile`

zonpctl,*p* `ifile ofile`

Description

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

Operators

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.

zonstd Zonal standard deviation
For every latitude the standard deviation over all longitudes is computed. Normalize by n.

zonstd1 Zonal standard deviation (n-1)
For every latitude the standard deviation over all longitudes is computed. Normalize by (n-1).

zonvar Zonal variance
For every latitude the variance over all longitudes is computed. Normalize by n.

zonvar1 Zonal variance (n-1)
For every latitude the variance over all longitudes is computed. Normalize by (n-1).

zonpctl Zonal percentiles
For every latitude the pth percentile over all longitudes is computed.

Parameter

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

Example

To compute the zonal mean of all input fields use:

```
cdo zonmean ifile ofile
```

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

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

2.8.7. MERSTAT - Meridional statistical values

Synopsis

<*operator*> `ifile ofile`

merpctl,*p* `ifile ofile`

Description

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

Operators

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.
merstd	Meridional standard deviation For every longitude the standard deviation over all latitudes is computed. Normalize by n.
merstd1	Meridional standard deviation (n-1) For every longitude the standard deviation over all latitudes is computed. Normalize by (n-1).
mervar	Meridional variance For every longitude the variance over all latitudes is computed. Normalize by n.
mervar1	Meridional variance (n-1) For every longitude the variance over all latitudes is computed. Normalize by (n-1).
merpctl	Meridional percentiles For every longitude the pth percentile over all latitudes is computed.

Parameter

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

Example

To compute the meridional mean of all input fields use:

```
cdo mermean ifile ofile
```

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

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

2.8.8. GRIDBOXSTAT - Statistical values over grid boxes

Synopsis

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

Description

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

Operators

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

Parameter

nx	INTEGER	Number of grid boxes in x direction
ny	INTEGER	Number of grid boxes in y direction

Example

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

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

2.8.9. VERTSTAT - Vertical statistical values

Synopsis

<operator> `ifile ofile`

Description

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

Operators

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 layer weighted mean over all levels is computed.

vertavg Vertical average
For every gridpoint the layer weighted average over all levels is computed.

vertstd Vertical standard deviation
For every gridpoint the standard deviation over all levels is computed. Normalize by n.

vertstd1 Vertical standard deviation (n-1)
For every gridpoint the standard deviation over all levels is computed. Normalize by (n-1).

vertvar Vertical variance
For every gridpoint the variance over all levels is computed. Normalize by n.

vertvar1 Vertical variance (n-1)
For every gridpoint the variance over all levels is computed. Normalize by (n-1).

Example

To compute the vertical sum of all input variables use:

```
cdo vertsum ifile ofile
```

2.8.10. TIMSELSTAT - Time range statistical values

Synopsis

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

Description

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

Operators

timselmin Time range minimum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$

timselmax Time range maximum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$

timselsum Time range sum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$

timselmean Time range mean
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$

timselavg Time range average
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$

timselstd Time range standard deviation
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

timselstd1 Time range standard deviation (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$

timselvar Time range variance
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$

timselvar1 Time range variance (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:
$o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

Parameter

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

Example

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

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

2.8.11. TIMSELPCTL - Time range percentile values

Synopsis

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

Description

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

For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same selected time range it is:

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

Parameter

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

Environment

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

2.8.12. RUNSTAT - Running statistical values

Synopsis

<*operator*>,*nts* `ifile ofile`

Description

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

Operators

runmin Running minimum
$o(t + (nts - 1)/2, x) = \mathbf{min}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

runmax Running maximum
$o(t + (nts - 1)/2, x) = \mathbf{max}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

runsum Running sum
$o(t + (nts - 1)/2, x) = \mathbf{sum}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

runmean Running mean
$o(t + (nts - 1)/2, x) = \mathbf{mean}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

runavg Running average
$o(t + (nts - 1)/2, x) = \mathbf{avg}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

runstd Running standard deviation
Normalize by n.

$o(t + (nts - 1)/2, x) = \mathbf{std}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

runstd1 Running standard deviation (n-1)
Normalize by (n-1).

$o(t + (nts - 1)/2, x) = \mathbf{std1}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

runvar Running variance
Normalize by n.

$o(t + (nts - 1)/2, x) = \mathbf{var}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

runvar1 Running variance (n-1)
Normalize by (n-1).

$o(t + (nts - 1)/2, x) = \mathbf{var1}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\}$

Parameter

nts INTEGER Number of timesteps

Environment

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

Example

To compute the running mean over 9 timesteps use:

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

2.8.13. RUNPCTL - Running percentile values

Synopsis

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

Description

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

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

Parameter

p	FLOAT	Percentile number in 0, ..., 100
nts	INTEGER	Number of timesteps

Example

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

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

2.8.14. TIMSTAT - Statistical values over all timesteps

Synopsis

<*operator*> `ifile ofile`

Description

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

Operators

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
$o(1, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$

timstd Time standard deviation
Normalize by n.

$o(1, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

timstd1 Time standard deviation (n-1)
Normalize by (n-1).

$o(1, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$

timvar Time variance
Normalize by n.

$o(1, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$

timvar1 Time variance (n-1)
Normalize by (n-1).

$o(1, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

Example

To compute the mean over all input timesteps use:

```
cdo timmean ifile ofile
```

2.8.15. TIMPCTL - Percentile values over all timesteps

Synopsis

timpctl,*p* `ifile1 ifile2 ifile3 ofile`

Description

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

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

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.16. HOURSTAT - Hourly statistical values

Synopsis

<*operator*> `ifile ofile`

Description

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

Operators

hourmin Hourly minimum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps of the same hour it is:
$o(t,x) = \mathbf{avg}\{i(t',x), t_1 < t' \leq t_n\}$

hourstd Hourly standard deviation
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same hour it is:
$o(t,x) = \mathbf{std}\{i(t',x), t_1 < t' \leq t_n\}$

hourstd1 Hourly standard deviation (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same hour it is:
$o(t,x) = \mathbf{std1}\{i(t',x), t_1 < t' \leq t_n\}$

hourvar Hourly variance
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same hour it is:
$o(t,x) = \mathbf{var}\{i(t',x), t_1 < t' \leq t_n\}$

hourvar1 Hourly variance (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same hour it is:
$o(t,x) = \mathbf{var1}\{i(t',x), t_1 < t' \leq t_n\}$

Example

To compute the hourly mean of a time series use:

```
cdo hourmean ifile ofile
```

2.8.17. HOURPCTL - Hourly percentile values

Synopsis

hourpctl,*p* `ifile1 ifile2 ifile3 ofile`

Description

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

For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same hour it is:

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

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.18. DAYSTAT - Daily statistical values

Synopsis

<*operator*> `ifile ofile`

Description

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

Operators

daymin Daily minimum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps of the same day it is:
$o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$

daystd Daily standard deviation
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same day it is:
$o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

daystd1 Daily standard deviation (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same day it is:
$o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$

dayvar Daily variance
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same day it is:
$o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$

dayvar1 Daily variance (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same day it is:
$o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

Example

To compute the daily mean of a time series use:

```
cdo daymean ifile ofile
```

2.8.19. DAYPCTL - Daily percentile values

Synopsis

daypctl,p `ifile1 ifile2 ifile3 ofile`

Description

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

For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same day it is:

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

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.20. MONSTAT - Monthly statistical values

Synopsis

<*operator*> `ifile ofile`

Description

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

Operators

monmin Monthly minimum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps of the same month it is:
$o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$

monstd Monthly standard deviation
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same month it is:
$o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

monstd1 Monthly standard deviation (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same month it is:
$o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$

monvar Monthly variance
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same month it is:
$o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$

monvar1 Monthly variance (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same month it is:
$o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

Example

To compute the monthly mean of a time series use:

```
cdo monmean ifile ofile
```

2.8.21. MONPCTL - Monthly percentile values

Synopsis

monpctl,*p* `ifile1 ifile2 ifile3 ofile`

Description

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

For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same month it is:

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

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.22. YEARMONSTAT - Yearly mean from monthly data

Synopsis

yearmonmean `ifile ofile`

Description

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

For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same year it is:
$o(t,x) = \textbf{mean}\{i(t',x), t_1 < t' \leq t_n\}$

Environment

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

Example

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

```
cdo yearmonmean ifile ofile
```

2.8.23. YEARSTAT - Yearly statistical values

Synopsis

<*operator*> `ifile ofile`

Description

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

Operators

yearmin Yearly minimum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps 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 $t\_1, ..., t\_n$ of timesteps of the same year it is:
$o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$

yearstd Yearly standard deviation
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same year it is:
$o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

yearstd1 Yearly standard deviation (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same year it is:
$o(t, x) = \mathbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$

yearvar Yearly variance
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same year it is:
$o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$

yearvar1 Yearly variance (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same year it is:
$o(t, x) = \mathbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

Note

The operators yearmean and yearavg compute only arithmetical means!

Example

To compute the yearly mean of a time series use:

```
cdo yearmean ifile ofile
```

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

```
cdo yearmonmean ifile ofile
```

2.8.24. YEARPCTL - Yearly percentile values

Synopsis

yearpctl,*p* `ifile1 ifile2 ifile3 ofile`

Description

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

For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same year it is:

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

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.25. SEASSTAT - Seasonal statistical values

Synopsis

<operator> `ifile ofile`

Description

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

Operators

seasmin Seasonal minimum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{min}\{i(t', x), t_1 < t' \leq t_n\}$

seasmax Seasonal maximum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{max}\{i(t', x), t_1 < t' \leq t_n\}$

seassum Seasonal sum
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$

seasmean Seasonal mean
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$

seasavg Seasonal average
For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$

seasstd Seasonal standard deviation
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

seasstd1 Seasonal standard deviation (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{std1}\{i(t', x), t_1 < t' \leq t_n\}$

seasvar Seasonal variance
Normalize by n. For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{var}\{i(t', x), t_1 < t' \leq t_n\}$

seasvar1 Seasonal variance (n-1)
Normalize by (n-1). For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:
$o(t, x) = \textbf{var1}\{i(t', x), t_1 < t' \leq t_n\}$

Example

To compute the seasonal mean of a time series use:

```
cdo seasmean ifile ofile
```

2.8.26. SEASPCTL - Seasonal percentile values

Synopsis

seaspctl,*p* `ifile1 ifile2 ifile3 ofile`

Description

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

For every adjacent sequence $t\_1, ..., t\_n$ of timesteps of the same season it is:

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

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.27. YHOURSTAT - Multi-year hourly statistical values

Synopsis

<*operator*> `ifile ofile`

Description

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

Operators

yhourmin Multi-year hourly minimum

$$o(0001,x) = \mathbf{min}\{i(t,x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784,x) = \mathbf{min}\{i(t,x), \mathrm{day}(i(t)) = 8784\}$$

yhourmax Multi-year hourly maximum

$$o(0001,x) = \mathbf{max}\{i(t,x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784,x) = \mathbf{max}\{i(t,x), \mathrm{day}(i(t)) = 8784\}$$

yhoursum Multi-year hourly sum

$$o(0001,x) = \mathbf{sum}\{i(t,x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784,x) = \mathbf{sum}\{i(t,x), \mathrm{day}(i(t)) = 8784\}$$

yhourmean Multi-year hourly mean

$$o(0001,x) = \mathbf{mean}\{i(t,x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784,x) = \mathbf{mean}\{i(t,x), \mathrm{day}(i(t)) = 8784\}$$

yhouravg Multi-year hourly average

$$o(0001,x) = \mathbf{avg}\{i(t,x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784,x) = \mathbf{avg}\{i(t,x), \mathrm{day}(i(t)) = 8784\}$$

yhourstd Multi-year hourly standard deviation
Normalize by n.

$$o(0001,x) = \mathbf{std}\{i(t,x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784,x) = \mathbf{std}\{i(t,x), \mathrm{day}(i(t)) = 8784\}$$

yhourstd1 Multi-year hourly standard deviation (n-1)
Normalize by (n-1).

$$o(0001,x) = \mathbf{std1}\{i(t,x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784,x) = \mathbf{std1}\{i(t,x), \mathrm{day}(i(t)) = 8784\}$$

yhourvar Multi-year hourly variance
Normalize by n.

$$o(0001, x) = \mathbf{var}\{i(t, x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784, x) = \mathbf{var}\{i(t, x), \mathrm{day}(i(t)) = 8784\}$$

yhourvar1 Multi-year hourly variance (n-1)
Normalize by (n-1).

$$o(0001, x) = \mathbf{var1}\{i(t, x), \mathrm{day}(i(t)) = 0001\}$$
$$\vdots$$
$$o(8784, x) = \mathbf{var1}\{i(t, x), \mathrm{day}(i(t)) = 8784\}$$

2.8.28. YDAYSTAT - Multi-year daily statistical values

Synopsis

<operator> `ifile ofile`

Description

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

Operators

ydaymin Multi-year daily minimum

$$o(001,x) = \mathbf{min}\{i(t,x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{min}\{i(t,x), \mathrm{day}(i(t)) = 366\}$$

ydaymax Multi-year daily maximum

$$o(001,x) = \mathbf{max}\{i(t,x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{max}\{i(t,x), \mathrm{day}(i(t)) = 366\}$$

ydaysum Multi-year daily sum

$$o(001,x) = \mathbf{sum}\{i(t,x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{sum}\{i(t,x), \mathrm{day}(i(t)) = 366\}$$

ydaymean Multi-year daily mean

$$o(001,x) = \mathbf{mean}\{i(t,x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{mean}\{i(t,x), \mathrm{day}(i(t)) = 366\}$$

ydayavg Multi-year daily average

$$o(001,x) = \mathbf{avg}\{i(t,x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{avg}\{i(t,x), \mathrm{day}(i(t)) = 366\}$$

ydaystd Multi-year daily standard deviation
Normalize by n.

$$o(001,x) = \mathbf{std}\{i(t,x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{std}\{i(t,x), \mathrm{day}(i(t)) = 366\}$$

ydaystd1 Multi-year daily standard deviation (n-1)
Normalize by (n-1).

$$o(001,x) = \mathbf{std1}\{i(t,x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{std1}\{i(t,x), \mathrm{day}(i(t)) = 366\}$$

ydayvar Multi-year daily variance
Normalize by n.

$$o(001, x) = \textbf{var}\{i(t, x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366, x) = \textbf{var}\{i(t, x), \mathrm{day}(i(t)) = 366\}$$

ydayvar1 Multi-year daily variance (n-1)
Normalize by (n-1).

$$o(001, x) = \textbf{var1}\{i(t, x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366, x) = \textbf{var1}\{i(t, x), \mathrm{day}(i(t)) = 366\}$$

Example

To compute the daily mean over all input years use:

```
cdo ydaymean ifile ofile
```

2.8.29. YDAYPCTL - Multi-year daily percentile values

Synopsis

ydaypctl,*p* `ifile1 ifile2 ifile3 ofile`

Description

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

$$o(001, x) = \textbf{pth percentile}\{i(t, x), \mathrm{day}(i(t)) = 001\}$$
$$\vdots$$
$$o(366, x) = \textbf{pth percentile}\{i(t, x), \mathrm{day}(i(t)) = 366\}$$

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.30. YMONSTAT - Multi-year monthly statistical values

Synopsis

<*operator*> `ifile ofile`

Description

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

Operators

ymonmin Multi-year monthly minimum

$$o(01, x) = \mathbf{min}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{min}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

ymonmax Multi-year monthly maximum

$$o(01, x) = \mathbf{max}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{max}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

ymonsum Multi-year monthly sum

$$o(01, x) = \mathbf{sum}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{sum}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

ymonmean Multi-year monthly mean

$$o(01, x) = \mathbf{mean}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{mean}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

ymonavg Multi-year monthly average

$$o(01, x) = \mathbf{avg}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{avg}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

ymonstd Multi-year monthly standard deviation
Normalize by n.

$$o(01, x) = \mathbf{std}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{std}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

ymonstd1 Multi-year monthly standard deviation (n-1)
Normalize by (n-1).

$$o(01, x) = \mathbf{std1}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{std1}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

ymonvar Multi-year monthly variance
Normalize by n.

$$o(01, x) = \mathbf{var}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{var}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

ymonvar1 Multi-year monthly variance (n-1)
Normalize by (n-1).

$$o(01, x) = \mathbf{var1}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \mathbf{var1}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

Example

To compute the monthly mean over all input years use:

```
cdo ymonmean ifile ofile
```

2.8.31. YMONPCTL - Multi-year monthly percentile values

Synopsis

ymonpctl,*p* `ifile1 ifile2 ifile3 ofile`

Description

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

$$o(01, x) = \textbf{pth percentile}\{i(t, x), \mathrm{month}(i(t)) = 01\}$$
$$\vdots$$
$$o(12, x) = \textbf{pth percentile}\{i(t, x), \mathrm{month}(i(t)) = 12\}$$

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.32. YSEASSTAT - Multi-year seasonal statistical values

Synopsis

$<operator>$ `ifile ofile`

Description

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

Operators

yseasmin Multi-year seasonal minimum

$o(1,x) = \mathbf{min}\{i(t,x), \mathrm{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2,x) = \mathbf{min}\{i(t,x), \mathrm{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3,x) = \mathbf{min}\{i(t,x), \mathrm{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4,x) = \mathbf{min}\{i(t,x), \mathrm{month}(i(t)) = 09,\ 10,\ 11\}$

yseasmax Multi-year seasonal maximum

$o(1,x) = \mathbf{max}\{i(t,x), \mathrm{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2,x) = \mathbf{max}\{i(t,x), \mathrm{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3,x) = \mathbf{max}\{i(t,x), \mathrm{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4,x) = \mathbf{max}\{i(t,x), \mathrm{month}(i(t)) = 09,\ 10,\ 11\}$

yseassum Multi-year seasonal sum

$o(1,x) = \mathbf{sum}\{i(t,x), \mathrm{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2,x) = \mathbf{sum}\{i(t,x), \mathrm{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3,x) = \mathbf{sum}\{i(t,x), \mathrm{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4,x) = \mathbf{sum}\{i(t,x), \mathrm{month}(i(t)) = 09,\ 10,\ 11\}$

yseasmean Multi-year seasonal mean

$o(1,x) = \mathbf{mean}\{i(t,x), \mathrm{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2,x) = \mathbf{mean}\{i(t,x), \mathrm{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3,x) = \mathbf{mean}\{i(t,x), \mathrm{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4,x) = \mathbf{mean}\{i(t,x), \mathrm{month}(i(t)) = 09,\ 10,\ 11\}$

yseasavg Multi-year seasonal average

$o(1,x) = \mathbf{avg}\{i(t,x), \mathrm{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2,x) = \mathbf{avg}\{i(t,x), \mathrm{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3,x) = \mathbf{avg}\{i(t,x), \mathrm{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4,x) = \mathbf{avg}\{i(t,x), \mathrm{month}(i(t)) = 09,\ 10,\ 11\}$

yseasstd Multi-year seasonal standard deviation

$o(1,x) = \mathbf{std}\{i(t,x), \mathrm{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2,x) = \mathbf{std}\{i(t,x), \mathrm{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3,x) = \mathbf{std}\{i(t,x), \mathrm{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4,x) = \mathbf{std}\{i(t,x), \mathrm{month}(i(t)) = 09,\ 10,\ 11\}$

yseasstd1 Multi-year seasonal standard deviation (n-1)

$o(1,x) = \mathbf{std1}\{i(t,x), \mathrm{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2,x) = \mathbf{std1}\{i(t,x), \mathrm{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3,x) = \mathbf{std1}\{i(t,x), \mathrm{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4,x) = \mathbf{std1}\{i(t,x), \mathrm{month}(i(t)) = 09,\ 10,\ 11\}$

yseasvar Multi-year seasonal variance

$o(1, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4, x) = \mathbf{var}\{i(t, x), \text{month}(i(t)) = 09,\ 10,\ 11\}$

yseasvar1 Multi-year seasonal variance (n-1)

$o(1, x) = \mathbf{var1}\{i(t, x), \text{month}(i(t)) = 12,\ 01,\ 02\}$
$o(2, x) = \mathbf{var1}\{i(t, x), \text{month}(i(t)) = 03,\ 04,\ 05\}$
$o(3, x) = \mathbf{var1}\{i(t, x), \text{month}(i(t)) = 06,\ 07,\ 08\}$
$o(4, x) = \mathbf{var1}\{i(t, x), \text{month}(i(t)) = 09,\ 10,\ 11\}$

Example

To compute the seasonal mean over all input years use:

```
cdo yseasmean ifile ofile
```

2.8.33. YSEASPCTL - Multi-year seasonal percentile values

Synopsis

yseaspctl,*p* `ifile1 ifile2 ifile3 ofile`

Description

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

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

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.8.34. YDRUNSTAT - Multi-year daily running statistical values

Synopsis

<*operator*>*,nts* `ifile ofile`

Description

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

Operators

ydrunmin Multi-year daily running minimum

$$o(001,x) = \mathbf{min}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{min}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 366\}$$

ydrunmax Multi-year daily running maximum

$$o(001,x) = \mathbf{max}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{max}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 366\}$$

ydrunsum Multi-year daily running sum

$$o(001,x) = \mathbf{sum}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{sum}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 366\}$$

ydrunmean Multi-year daily running mean

$$o(001,x) = \mathbf{mean}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{mean}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 366\}$$

ydrunavg Multi-year daily running average

$$o(001,x) = \mathbf{avg}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{avg}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 366\}$$

ydrunstd Multi-year daily running standard deviation
Normalize by n.

$$o(001,x) = \mathbf{std}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 001\}$$
$$\vdots$$
$$o(366,x) = \mathbf{std}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); \mathrm{day}[(i(t+(nts-1)/2)] = 366\}$$

ydrunstd1 Multi-year daily running standard deviation (n-1)
Normalize by (n-1).

$$o(001, x) = \mathbf{std1}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x); \mathrm{day}[(i(t + (nts - 1)/2)] = 001\}$$
$$\vdots$$
$$o(366, x) = \mathbf{std1}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x); \mathrm{day}[(i(t + (nts - 1)/2)] = 366\}$$

ydrunvar Multi-year daily running variance
Normalize by n.

$$o(001, x) = \mathbf{var}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x); \mathrm{day}[(i(t + (nts - 1)/2)] = 001\}$$
$$\vdots$$
$$o(366, x) = \mathbf{var}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x); \mathrm{day}[(i(t + (nts - 1)/2)] = 366\}$$

ydrunvar1 Multi-year daily running variance (n-1)
Normalize by (n-1).

$$o(001, x) = \mathbf{var1}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x); \mathrm{day}[(i(t + (nts - 1)/2)] = 001\}$$
$$\vdots$$
$$o(366, x) = \mathbf{var1}\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x); \mathrm{day}[(i(t + (nts - 1)/2)] = 366\}$$

Parameter

nts INTEGER Number of timesteps

Example

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

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

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

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

2.8.35. YDRUNPCTL - Multi-year daily running percentile values

Synopsis

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

Description

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

$$o(001, x) = \textbf{pth percentile}\{i(t, x), i(t+1, x), ..., i(t+nts-1, x); \mathrm{day}[(i(t+(nts-1)/2)] = 001\}$$
$$\vdots$$
$$o(366, x) = \textbf{pth percentile}\{i(t, x), i(t+1, x), ..., i(t+nts-1, x); \mathrm{day}[(i(t+(nts-1)/2)] = 366\}$$

Parameter

p	FLOAT	Percentile number in 0, ..., 100
nts	INTEGER	Number of timesteps

Environment

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

Example

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

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

Or shorter using operator piping:

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

2.9. Correlation and co.

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

Covariance **covar**	$n^{-1} \sum_{i=1}^{n} (x_i - \overline{x})^2 (y_i - \overline{y})^2$
covar weighted by $\{w_i, i = 1, ..., n\}$	$\left(\sum_{j=1}^{n} w_j \right)^{-1} \sum_{i=1}^{n} w_i \left(x_i - \left(\sum_{j=1}^{n} w_j \right)^{-1} \sum_{j=1}^{n} w_j \, x_j \right) \left(y_i - \left(\sum_{j=1}^{n} w_j \right)^{-1} \sum_{j=1}^{n} w_j \, y_j \right)$

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

fldcor	Correlation in grid space
timcor	Correlation over time
fldcovar	Covariance in grid space
timcovar	Covariance over time

2.9.1. FLDCOR - Correlation in grid space

Synopsis

fldcor `ifile1 ifile2 ofile`

Description

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

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

it is

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

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

2.9.2. TIMCOR - Correlation over time

Synopsis

timcor `ifile1 ifile2 ofile`

Description

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

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

it is

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

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

2.9.3. FLDCOVAR - Covariance in grid space

Synopsis

fldcovar `ifile1 ifile2 ofile`

Description

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

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

it is

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

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

2.9.4. TIMCOVAR - Covariance over time

Synopsis

timcovar `ifile1 ifile2 ofile`

Description

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

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

it is

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

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

2.10. Regression

This sections contains modules for linear regression of time series.

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

regres	Regression
detrend	Detrend
trend	Trend
subtrend	Subtract trend

2.10.1. REGRES - Regression

Synopsis

regres `ifile ofile`

Description

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

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

2.10.2. DETREND - Detrend time series

Synopsis

detrend `ifile ofile`

Description

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

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

and

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

it is

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

Note

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

Example

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

```
cdo detrend ifile ofile
```

2.10.3. TREND - Trend of time series

Synopsis

trend `ifile ofile1 ofile2`

Description

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

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

and

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

Thus the estimation for a is stored in `ofile1` and that for b is stored in `ofile2`. To subtract the trend from the data see operator subtrend.

2.10.4. SUBTREND - Subtract a trend

Synopsis

subtrend `ifile1 ifile2 ifile3 ofile`

Description

This operator is for subtracting a trend computed by the operator trend. It is

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

where t is the timesteps.

Example

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

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

The result is identical to a call of the operator detrend:

```
cdo detrend ifile ofile
```

2.11. EOFs

This section contains modules to compute Empirical Orthogonal Functions and - once they are computed - their principal coefficients.
An introduction to the theory of principal component analysis as applied here can be found in:
Principal Component Analysis [Peisendorfer]
Details about calculation in the time- and spatial spaces are found in:
Statistical Analysis in Climate Research [vonStorch]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.11.1. EOFS - Empirical Orthogonal Functions

Synopsis

<operator>,neof `ifile ofile1 ofile2`

Description

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

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

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

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

Operators

eof	Calculate EOFs in spatial or time space
eoftime	Calculate EOFs in time space
eofspatial	Calculate EOFs in spatial space
eof3d	Calculate 3-Dimensional EOFs in time space

Parameter

neof	INTEGER	Number of eigen functions

Environment

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

Example

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

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

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

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

2.11.2. EOFCOEFF - Principal coefficients of EOFs

Synopsis

eofcoeff `ifile1 ifile2 obase`

Description

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

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

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

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

$$\sum_{t=1}^{n} o_j(t)o_k(t) = \left\{ \begin{array}{l} 0 \; if \; j \neq k \\ \lambda_j \; if \; j = k \end{array} \right. \quad with \; \sum_{t=1}^{n} o_j(t) = 0 \forall j \in \{1, \ldots, p\}.$$

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

Environment

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

Example

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

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

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

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

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

2.12. Interpolation

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

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

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

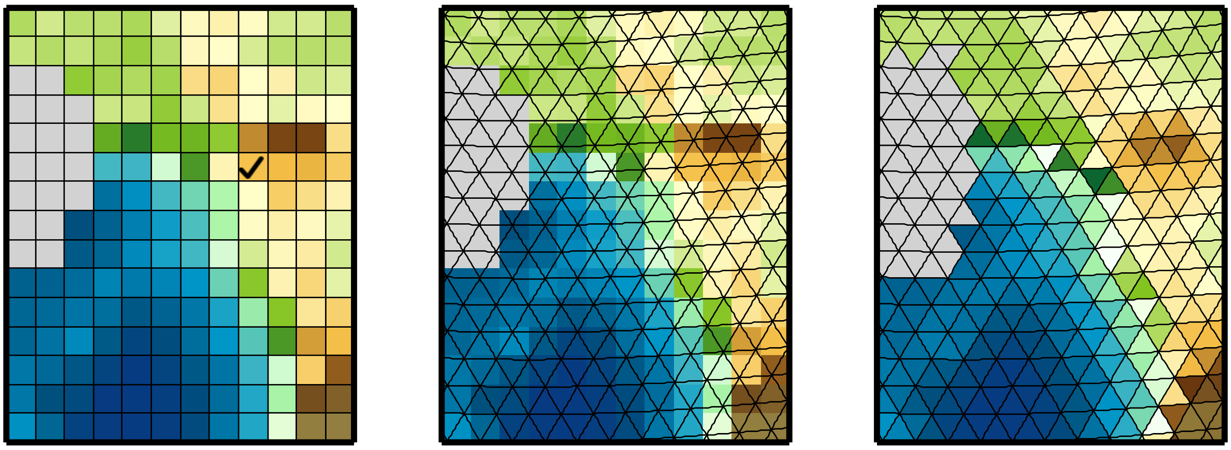
2.12.1. REMAPBIL - Bilinear interpolation

Synopsis

<operator>,grid `ifile ofile`

Description

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



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapbil Bilinear interpolation
Performs a bilinear interpolation on all input fields.

genbil Generate bilinear interpolation weights
Generates bilinear interpolation weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator remap to apply this remapping weights to a data file with the same source grid.

Parameter

grid STRING Target grid description file or name

Environment

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

Example

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

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

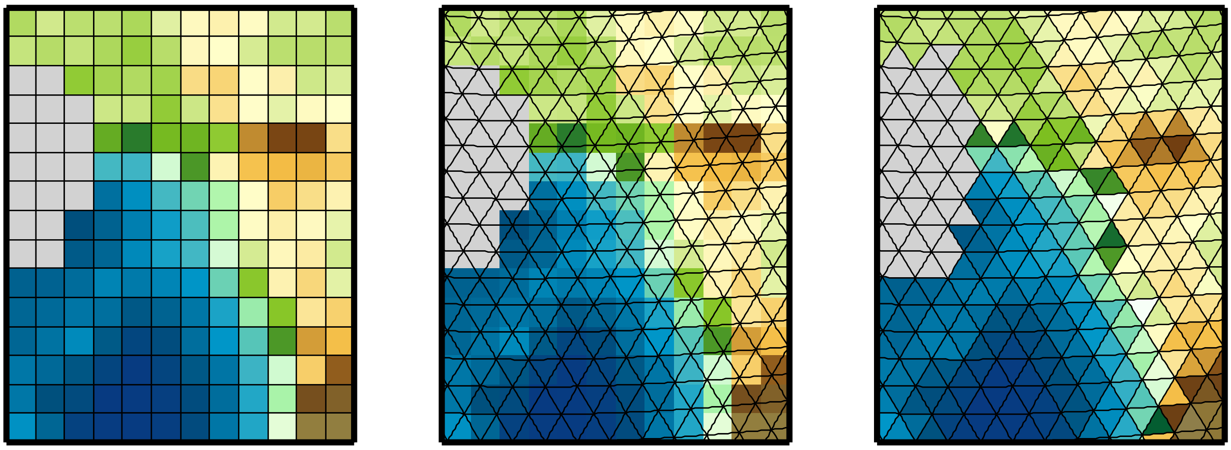
2.12.2. REMAPBIC - Bicubic interpolation

Synopsis

<operator>,grid `ifile ofile`

Description

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



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapbic Bicubic interpolation
Performs a bicubic interpolation on all input fields.

genbic Generate bicubic interpolation weights
Generates bicubic interpolation weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator remap to apply this remapping weights to a data file with the same source grid.

Parameter

grid STRING Target grid description file or name

Environment

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

Example

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

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

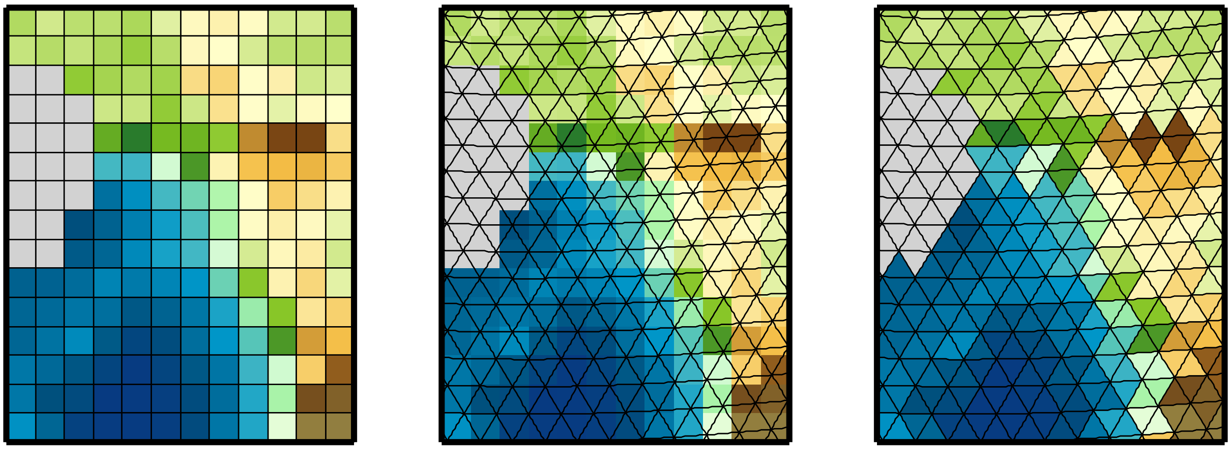
2.12.3. REMAPNN - Nearest neighbor remapping

Synopsis

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

Description

This module contains operators for a nearest neighbor remapping of fields between grids in spherical coordinates. Below is a schematic illustration of the nearest neighbor remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapnn Nearest neighbor remapping
Performs a nearest neighbor remapping on all input fields.

gennn Generate nearest neighbor remap weights
Generates nearest neighbor remapping weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator remap to apply this remapping weights to a data file with the same source grid.

Parameter

grid STRING Target grid description file or name

Environment

`REMAP_EXTRAPOLATE` This variable is used to switch the extrapolation feature 'on' or 'off'. By default the extrapolation is enabled for this remapping method.

`CDO_GRIDSEARCH_RADIUS` Grid search radius in degree, default 180 degree.

2.12.4. REMAPDIS - Distance-weighted average remapping

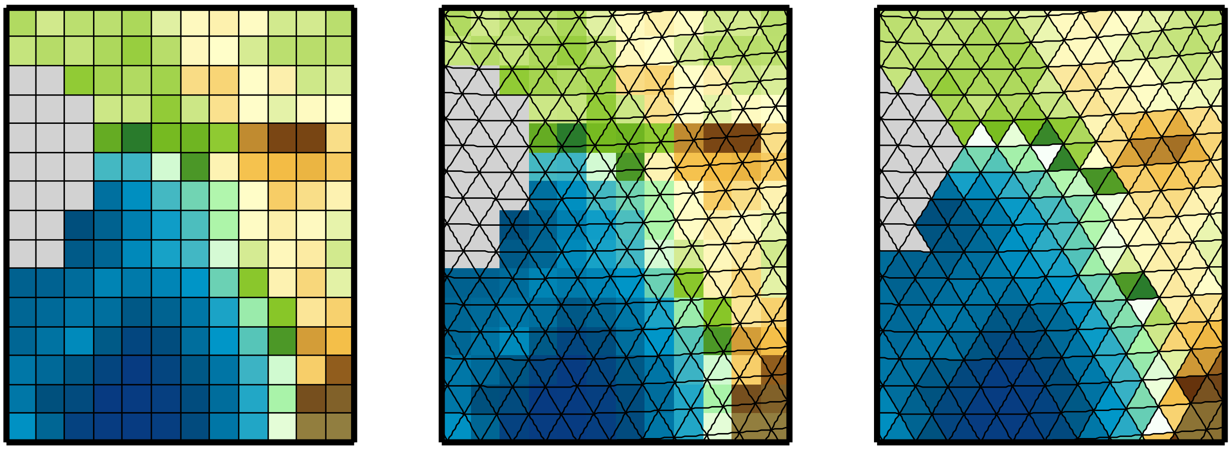
Synopsis

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

gendis,*grid* `ifile ofile`

Description

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



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

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

Parameter

grid	STRING	Target grid description file or name
neighbors	INTEGER	Number of nearest neighbors

Environment

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

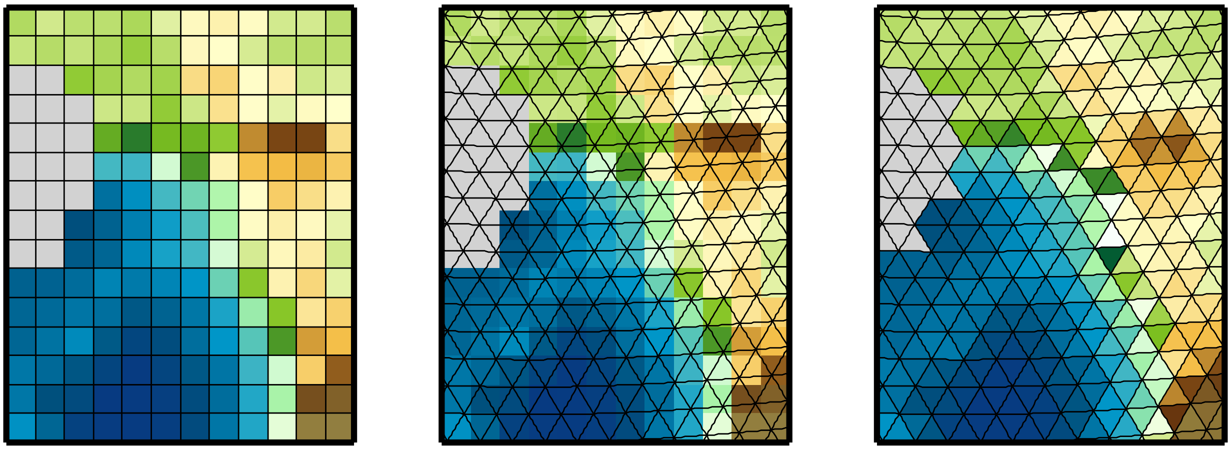
2.12.5. REMAPYCON - First order conservative remapping

Synopsis

<operator>,grid `ifile ofile`

Description

This module contains operators for a first order conservative remapping of fields between grids in spherical coordinates. The operators in this module uses code from the YAC software package to compute the conservative remapping weights. For a detailed description of the interpolation method see [YAC]. The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for the conservative remapping requires that no grid cell occurs more than once. Below is a schematic illustration of the 1st order conservative remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapycon First order conservative remapping
Performs a first order conservative remapping on all input fields.

genycon Generate 1st order conservative remap weights
Generates first order conservative remapping weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator remap to apply this remapping weights to a data file with the same source grid.

Parameter

grid STRING Target grid description file or name

Environment

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

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

Example

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

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

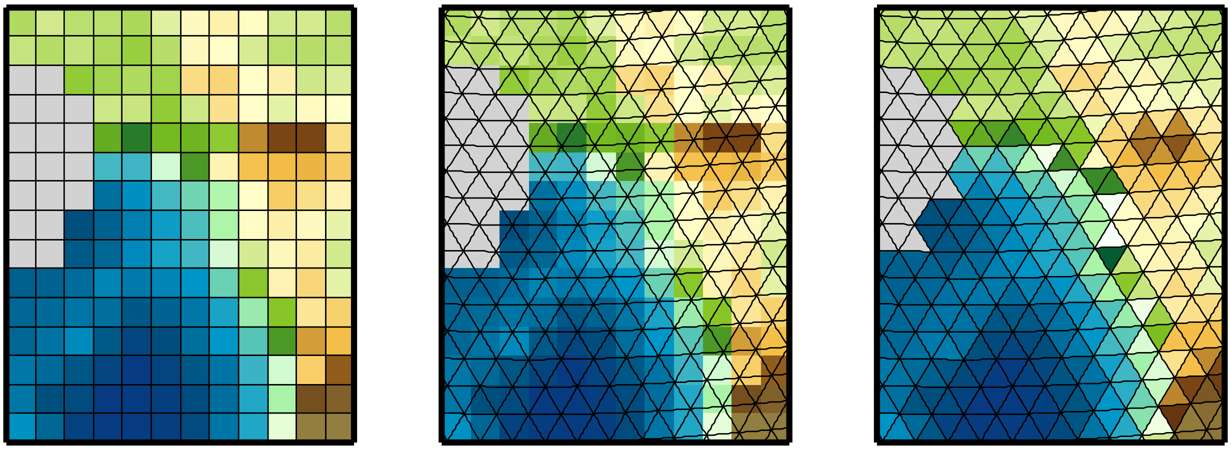
2.12.6. REMAPCON - First order conservative remapping

Synopsis

<operator>,grid `ifile ofile`

Description

This module contains operators for a first order conservative remapping of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see [SCRIP]. The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for the conservative remapping requires that no grid cell occurs more than once. Below is a schematic illustration of the 1st order conservative remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapcon First order conservative remapping
Performs a first order conservative remapping on all input fields.

gencon Generate 1st order conservative remap weights
Generates first order conservative remapping weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator remap to apply this remapping weights to a data file with the same source grid.

Parameter

grid STRING Target grid description file or name

Environment

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

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

Note

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

Example

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

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

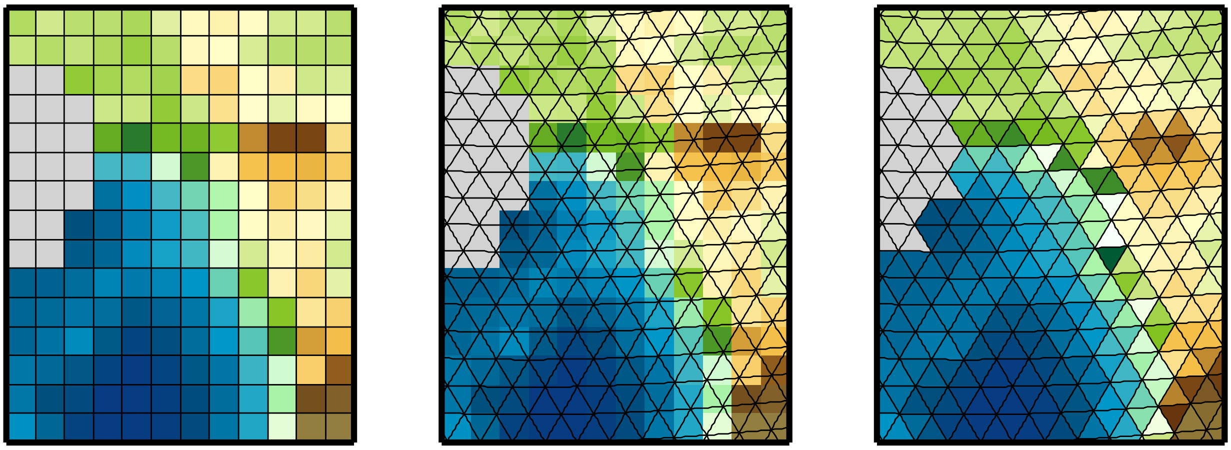
2.12.7. REMAPCON2 - Second order conservative remapping

Synopsis

<operator>,grid `ifile ofile`

Description

This module contains operators for a second order conservative remapping of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see [SCRIP]. The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for the conservative remapping requires that no grid cell occurs more than once. Below is a schematic illustration of the 2nd order conservative remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapcon2 Second order conservative remapping
Performs a second order conservative remapping on all input fields.

gencon2 Generate 2nd order conservative remap weights
Generates second order conservative remapping weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator remap to apply this remapping weights to a data file with the same source grid.

Parameter

grid STRING Target grid description file or name

Environment

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

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

Note

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

Example

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

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

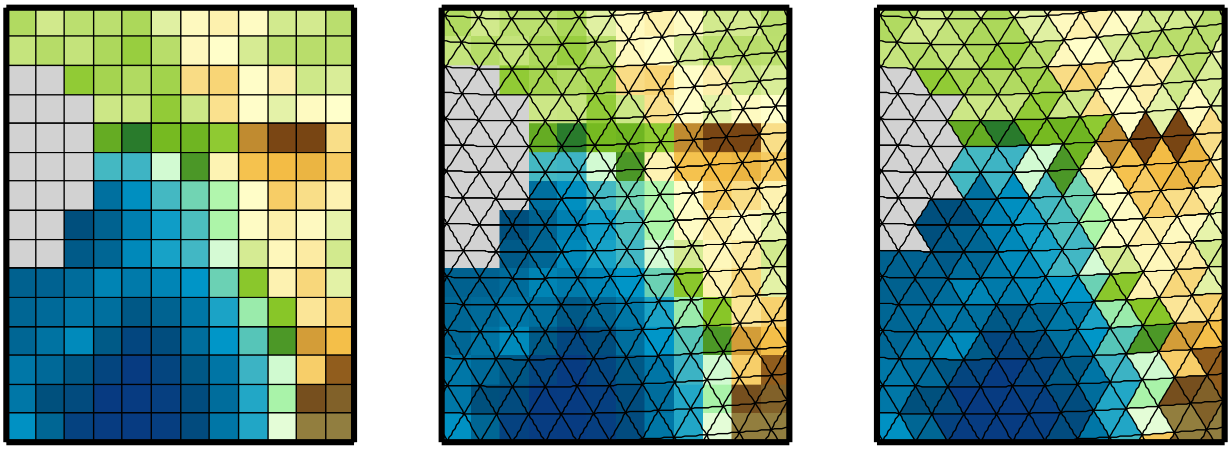
2.12.8. REMAPLAF - Largest area fraction remapping

Synopsis

<operator>,grid `ifile ofile`

Description

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



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remaplaf Largest area fraction remapping
Performs a largest area fraction remapping on all input fields.

genlaf Generate largest area fraction remap weights
Generates largest area fraction remapping weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator remap to apply this remapping weights to a data file with the same source grid.

Parameter

grid	STRING	Target grid description file or name

Environment

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

2.12.9. REMAP - Grid remapping

Synopsis

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

Description

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

Parameter

grid	STRING	Target grid description file or name
weights	STRING	Interpolation weights (SCRIP NetCDF file)

Environment

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

Example

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

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

The result will be the same as:

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

2.12.10. REMAPETA - Remap vertical hybrid level

Synopsis

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

Description

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

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

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

Parameter

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

Environment

`REMAPETA_PTOP`	Sets the minimum pressure level for condensation. Above this level the humidity is set to the constant 1.E-6. The default value is 0 Pa.

Note

The code numbers or the variable names of the required parameter have to follow the [ECHAM] convention. Presently, the vertical coordinate definition of a NetCDF file has also to follow the ECHAM convention. This means:

- the dimension of the full level coordinate and the corresponding variable is called mlev,
- the dimension of the half level coordinate and the corresponding variable is called ilev (ilev must have one element more than mlev)
- the hybrid vertical coefficient a is given in units of Pa and called hyai (hyam for level midpoints)
- the hybrid vertical coefficient b is given in units of 1 and called hybi (hybm for level midpoints)

- the mlev variable has a borders attribute containing the character string 'ilev'

Use the sinfo command to test if your vertical coordinate system is recognized as hybrid system.

In case remapeta complains about not finding any data on hybrid model levels you may wish to use the setzaxis command to generate a zaxis description which conforms to the ECHAM convention. See section "1.4 Z-axis description" for an example how to define a hybrid Z-axis.

Example

To remap between different hybrid model level data use:

```
cdo remapeta,vct ifile ofile
```

Here is an example vct file with 19 hybrid model level:

```
 0       0.00000000000000000    0.00000000000000000
 1    2000.00000000000000000    0.00000000000000000
 2    4000.00000000000000000    0.00000000000000000
 3    6046.10937500000000000    0.00033899326808751
 4    8267.92968750000000000    0.00335718691349030
 5   10609.51171875000000000    0.01307003945112228
 6   12851.10156250000000000    0.03407714888453484
 7   14698.50000000000000000    0.07064980268478394
 8   15861.12890625000000000    0.12591671943664551
 9   16116.23828125000000000    0.20119541883468628
10   15356.92187500000000000    0.29551959037780762
11   13621.46093750000000000    0.40540921688079834
12   11101.55859375000000000    0.52493220567703247
13    8127.14453125000000000    0.64610791206359863
14    5125.14062500000000000    0.75969839096069336
15    2549.96899414062500000    0.85643762350082397
16     783.19506835937500000    0.92874687910079956
17       0.00000000000000000    0.97298520803451538
18       0.00000000000000000    0.99228149652481079
19       0.00000000000000000    1.00000000000000000
```

2.12.11. VERTINTML - Vertical interpolation

Synopsis

ml2pl,*plevels* `ifile ofile`

ml2hl,*hlevels* `ifile ofile`

Description

Interpolate 3D variables on hybrid sigma pressure level to pressure or height levels. The input file should contain the log. surface pressure or the surface pressure. To interpolate the temperature, the surface geopotential is also needed. The pressure, temperature, and surface geopotential are identified by their GRIB1 code number or NetCDF CF standard name. Supported parameter tables are: WMO standard table number 2 and ECMWF local table number 128. Use the alias **ml2plx**/**ml2hlx** or the environment variable `EXTRAPOLATE` to extrapolate missing values. This operator requires all variables on the same horizontal grid.

Operators

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

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

Parameter

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

Environment

`EXTRAPOLATE`	If set to 1 extrapolate missing values.

Example

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

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

2.12.12. VERTINTAP - Vertical interpolation

Synopsis

ap2pl,*plevels* `ifile ofile`

Description

Interpolate 3D variables on hybrid sigma height coordinates to pressure levels. The input file must contain the 3D air pressure. The air pressure is identified by the NetCDF CF standard name air_pressure. This operator requires all variables on the same horizontal grid.

Parameter

plevels FLOAT Pressure levels in pascal

Note

This is a specific implementation for NetCDF files from the ICON model, it may not work with data from other sources.

Example

To interpolate 3D variables on hybrid sigma height level to pressure levels of 925, 850, 500 and 200 hPa use:

```
cdo ap2pl,92500,85000,50000,20000 ifile ofile
```

2.12.13. INTLEVEL - Linear level interpolation

Synopsis

intlevel,*levels* `ifile ofile`

Description

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

Parameter

levels FLOAT Target levels

Example

To interpolate 3D variables on height levels to a new set of height levels use:

```
cdo intlevel,10,50,100,500,1000 ifile ofile
```

2.12.14. INTLEVEL3D - Linear level interpolation from/to 3d vertical coordinates

Synopsis

<operator>,icoordinate `ifile1 ifile2 ofile`

Description

This operator performs a linear vertical interpolation of 3D variables fields with given 3D vertical coordinates.

Operators

intlevel3d	Linear level interpolation onto a 3d vertical coordinate
intlevelx3d	like intlevel3d but with extrapolation

Parameter

icoordinate	STRING	filename for vertical source coordinates variable
ifile2	STRING	target vertical coordinate field (intlevel3d only)

Example

To interpolate 3D variables from one set of 3d height levels into another one where

- `icoordinate` contains a single 3d variable, which represents the input 3d vertical coordinate
- `ifile1` contains the source data, which the vertical coordinate from icoordinate belongs to
- `ifile2` only contains the target 3d height levels

```
cdo intlevel3d,icoordinate ifile1 ifile2 ofile
```

2.12.15. INTTIME - Time interpolation

Synopsis

inttime,*date*,*time*[,*inc*] `ifile ofile`

intntime,*n* `ifile ofile`

Description

This module performs linear interpolation between timesteps.

Operators

inttime	Interpolation between timesteps This operator creates a new dataset by linear interpolation between timesteps. The user has to define the start date/time with an optional increment.
intntime	Interpolation between timesteps This operator performs linear interpolation between timesteps. The user has to define the number of timesteps from one timestep to the next.

Parameter

date	STRING	Start date (format YYYY-MM-DD)
time	STRING	Start time (format hh:mm:ss)
inc	STRING	Optional increment (seconds, minutes, hours, days, months, years) [default: 0hour]
n	INTEGER	Number of timesteps from one timestep to the next

Example

Assumed a 6 hourly dataset starts at 1987-01-01 12:00:00. To interpolate this time series to a one hourly dataset use:

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

2.12.16. INTYEAR - Year interpolation

Synopsis

intyear,*years* `ifile1 ifile2 obase`

Description

This operator performs linear interpolation between two years, timestep by timestep. The input files need to have the same structure with the same variables. The output files will be named `<obase><yyyy><suffix>` where yyyy will be the year and `suffix` is the filename extension derived from the file format.

Parameter

years	INTEGER	Comma separated list of years

Environment

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

Example

Assume there are two monthly mean datasets over a year. The first dataset has 12 timesteps for the year 1985 and the second one for the year 1990. To interpolate the years between 1985 and 1990 month by month use:

```
cdo intyear,1986,1987,1988,1989 ifile1 ifile2 year
```

Example result of 'dir year*' for NetCDF datasets:

```
year1986.nc year1987.nc year1988.nc year1989.nc
```

2.13. Transformation

This section contains modules to perform spectral transformations.

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

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)
dv2ps	D and V to velocity potential and stream function

2.13.1. SPECTRAL - Spectral transformation

Synopsis

<*operator*> `ifile ofile`

sp2sp,*trunc* `ifile ofile`

Description

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

Operators

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

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

Use this operator to convert ERA40 data e.g. from TL159 to N80.

gp2sp Gridpoint to spectral
Convert all Gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by: trunc $= (\text{nlat} * 2 - 1)/\boxed{3}$

gp2spl Gridpoint to spectral (linear)
Convert all Gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by: trunc $= (\text{nlat} * 2 - 1)/\boxed{2}$

Use this operator to convert ERA40 data e.g. from N80 to TL159 instead of T106.

sp2sp Spectral to spectral
Change the triangular truncation of all spectral fields. The operator performs downward conversion by cutting the resolution. Upward conversions are achieved by filling in zeros.

Parameter

trunc INTEGER New spectral resolution

Example

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

```
cdo sp2gp ifile ofile
```

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

```
cdo sp2gpl ifile ofile
```

2.13.2. WIND - Wind transformation

Synopsis

<*operator*> `ifile ofile`

Description

This module converts relative divergence and vorticity to U and V wind and vice versa. Divergence and vorticity are spherical harmonic coefficients in spectral space and U and V are on a regular Gaussian grid. The Gaussian latitudes needs to be ordered from north to south.

Operators

dv2uv Divergence and vorticity to U and V wind
Calculate U and V wind on a Gaussian grid from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need to have the names sd and svo or code numbers 155 and 138. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:
$\mathrm{nlat} = NINT((\mathrm{trunc} * \boxed{3} + 1.)/2.)$

dv2uvl Divergence and vorticity to U and V wind (linear)
Calculate U and V wind on a Gaussian grid from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need tohave the names sd and svo or code numbers 155 and 138. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:
$\mathrm{nlat} = NINT((\mathrm{trunc} * \boxed{2} + 1.)/2.)$

uv2dv U and V wind to divergence and vorticity
Calculate spherical harmonic coefficients of relative divergence and vorticity from U and V wind. The U and V wind need to be on a Gaussian grid and need to have the names u and v or the code numbers 131 and 132. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:
$\mathrm{trunc} = (\mathrm{nlat} * 2 - 1)/\boxed{3}$

uv2dvl U and V wind to divergence and vorticity (linear)
Calculate spherical harmonic coefficients of relative divergence and vorticity from U and V wind. The U and V wind need to be on a Gaussian grid and need to have the names u and v or the code numbers 131 and 132. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:
$\mathrm{trunc} = (\mathrm{nlat} * 2 - 1)/\boxed{2}$

dv2ps D and V to velocity potential and stream function
Calculate spherical harmonic coefficients of velocity potential and stream function from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need to have the names sd and svo or code numbers 155 and 138.

Example

Assume a dataset has at least spherical harmonic coefficients of divergence and vorticity. To transform the spectral divergence and vorticity to U and V wind on a Gaussian grid use:

```
cdo dv2uv ifile ofile
```

2.14. Import/Export

This section contains modules to import and export data files which can not read or write directly with CDO.

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

import_binary	Import binary data sets
import_cmsaf	Import CM-SAF HDF5 files
import_amsr	Import AMSR binary files
input	ASCII input
inputsrv	SERVICE ASCII input
inputext	EXTRA ASCII input
output	ASCII output
outputf	Formatted output
outputint	Integer output
outputsrv	SERVICE ASCII output
outputext	EXTRA ASCII output
outputtab	Table output
gmtxyz	GMT xyz format
gmtcells	GMT multiple segment format

2.14.1. IMPORTBINARY - Import binary data sets

Synopsis

import_binary `ifile ofile`

Description

This operator imports gridded binary data sets via a GrADS data descriptor file. The GrADS data descriptor file contains a complete description of the binary data as well as instructions on where to find the data and how to read it. The descriptor file is an ASCII file that can be created easily with a text editor. The general contents of a gridded data descriptor file are as follows:

- Filename for the binary data
- Missing or undefined data value
- Mapping between grid coordinates and world coordinates
- Description of variables in the binary data set

A detailed description of the components of a GrADS data descriptor file can be found in [GrADS]. Here is a list of the supported components: BYTESWAPPED, CHSUB, DSET, ENDVARS, FILEHEADER, HEADERBYTES, OPTIONS, TDEF, TITLE, TRAILERBYTES, UNDEF, VARS, XDEF, XYHEADER, YDEF, ZDEF

Note

Only 32-bit IEEE floats are supported for standard binary files!

Example

To convert a binary data file to NetCDF use:

```
cdo -f nc import_binary ifile.ctl ofile.nc
```

Here is an example of a GrADS data descriptor file:

```
DSET   ^ifile.bin
OPTIONS sequential
UNDEF  -9e+33
XDEF 360 LINEAR -179.5 1
YDEF 180 LINEAR  -89.5 1
ZDEF   1 LINEAR 1 1
TDEF   1 LINEAR 00:00Z15jun1989 12hr
VARS   1
param  1  99  description of the variable
ENDVARS
```

The binary data file ifile.bin contains one parameter on a global 1 degree lon/lat grid written with FORTRAN record length headers (sequential).

2.14.2. IMPORTCMSAF - Import CM-SAF HDF5 files

Synopsis

import_cmsaf `ifile ofile`

Description

This operator imports gridded CM-SAF (Satellite Application Facility on Climate Monitoring) HDF5 files. CM-SAF exploits data from polar-orbiting and geostationary satellites in order to provide climate monitoring products of the following parameters:

Cloud parameters: cloud fraction (CFC), cloud type (CTY), cloud phase (CPH), cloud top height, pressure and temperature (CTH,CTP,CTT), cloud optical thickness (COT), cloud water path (CWP).

Surface radiation components: Surface albedo (SAL); surface incoming (SIS) and net (SNS) shortwave radiation; surface downward (SDL) and outgoing (SOL) longwave radiation, surface net longwave radiation (SNL) and surface radiation budget (SRB).

Top-of-atmosphere radiation components: Incoming (TIS) and reflected (TRS) solar radiative flux at top-of-atmosphere. Emitted thermal radiative flux at top-of-atmosphere (TET).

Water vapour: Vertically integrated water vapour (HTW), layered vertically integrated water vapour and layer mean temperature and relative humidity for 5 layers (HLW), temperature and mixing ratio at 6 pressure levels.

Daily and monthly mean products can be ordered via the CM-SAF web page (www.cmsaf.eu). Products with higher spatial and temporal resolution, i.e. instantaneous swath-based products, are available on request (contact.cmsaf@dwd.de). All products are distributed free-of-charge. More information on the data is available on the CM-SAF homepage (www.cmsaf.eu).

Daily and monthly mean products are provided in equal-area projections. CDO reads the projection parameters from the metadata in the HDF5-headers in order to allow spatial operations like remapping. For spatial operations with instantaneous products on original satellite projection, additional files with arrays of latitudes and longitudes are needed. These can be obtained from CM-SAF together with the data.

Note

To use this operator, it is necessary to build CDO with HDF5 support (version 1.6 or higher). The PROJ.4 library (version 4.6 or higher) is needed for full support of the remapping functionality.

Example

A typical sequence of commands with this operator could look like this:

```
cdo -f nc remapbil,r360x180 -import_cmsaf cmsaf_product.hdf output.nc
```

(bilinear remapping to a predefined global grid with 1 deg resolution and conversion to NetCDF).

If you work with CM-SAF data on original satellite project, an additional file with information on geolocation is required, to perform such spatial operations:

```
cdo -f nc remapbil,r720x360 -setgrid,cmsaf_latlon.h5 -import_cmsaf cmsaf.hdf out.nc
```

Some CM-SAF data are stored as scaled integer values. For some operations, it could be desirable (or necessary) to increase the accuracy of the converted products:

```
cdo -b f32 -f nc fldmean -sellonlatbox,0,10,0,10 -remapbil,r720x360 \
              -import_cmsaf cmsaf_product.hdf output.nc
```

2.14.3. IMPORTAMSR - Import AMSR binary files

Synopsis

import_amsr `ifile ofile`

Description

This operator imports gridded binary AMSR (Advanced Microwave Scanning Radiometer) data. The binary data files are available from the AMSR ftp site (ftp://ftp.ssmi.com/amsre). Each file consists of twelve (daily) or five (averaged) 0.25 x 0.25 degree grid (1440,720) byte maps. For daily files, six daytime maps in the following order, Time (UTC), Sea Surface Temperature (SST), 10 meter Surface Wind Speed (WSPD), Atmospheric Water Vapor (VAPOR), Cloud Liquid Water (CLOUD), and Rain Rate (RAIN), are followed by six nighttime maps in the same order. Time-Averaged files contain just the geophysical layers in the same order [SST, WSPD, VAPOR, CLOUD, RAIN]. More information to the data is available on the AMSR homepage http://www.remss.com/amsr.

Example

To convert monthly binary AMSR files to NetCDF use:

```
cdo -f nc amsre_yyyymmv5 amsre_yyyymmv5.nc
```

2.14.4. INPUT - Formatted input

Synopsis

input,*grid* `ofile`

inputsrv `ofile`

inputext `ofile`

Description

This module reads time series of one 2D variable from standard input. All input fields need to have the same horizontal grid. The format of the input depends on the chosen operator.

Operators

input ASCII input
Reads fields with ASCII numbers from standard input and stores them in `ofile`. The numbers read are exactly that ones which are written out by the output operator.

inputsrv SERVICE ASCII input
Reads fields with ASCII numbers from standard input and stores them in `ofile`. Each field should have a header of 8 integers (SERVICE likely). The numbers that are read are exactly that ones which are written out by the outputsrv operator.

inputext EXTRA ASCII input
Read fields with ASCII numbers from standard input and stores them in `ofile`. Each field should have header of 4 integers (EXTRA likely). The numbers read are exactly that ones which are written out by the outputext operator.

Parameter

grid	STRING	Grid description file or name

Example

Assume an ASCII dataset contains a field on a global regular grid with 32 longitudes and 16 latitudes (512 elements). To create a GRIB1 dataset from the ASCII dataset use:

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

2.14.5. OUTPUT - Formatted output

Synopsis

output `ifiles`

outputf,*format[,nelem]* `ifiles`

outputint `ifiles`

outputsrv `ifiles`

outputext `ifiles`

Description

This module prints all values of all input datasets to standard output. All input fields need to have the same horizontal grid. All input files need to have the same structure with the same variables. The format of the output depends on the chosen operator.

Operators

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 have to be specified by the parameters *format* and *nelem*. The default for *nelem* is 1.

outputint Integer output
Prints all values rounded to the nearest integer to standard output.

outputsrv SERVICE ASCII output
Prints all values to standard output. Each field with a header of 8 integers (SERVICE likely).

outputext EXTRA ASCII output
Prints all values to standard output. Each field with a header of 4 integers (EXTRA likely).

Parameter

format	STRING	C-style format for one element (e.g. %13.6g)
nelem	INTEGER	Number of elements for each row (default: nelem = 1)

Example

To print all field elements of a dataset formatted with "%8.4g" and 8 values per line use:

```
cdo outputf,%8.4g,8 ifile
```

Example result of a dataset with one field on 64 grid points:

```
 261.7     262   257.8   252.5   248.8   247.7   246.3   246.1
 250.6   252.6   253.9   254.8     252   246.6   249.7   257.9
 273.4   266.2   259.8   261.6   257.2   253.4     251   263.7
 267.5   267.4   272.2   266.7   259.6   255.2   272.9   277.1
 275.3   275.5   276.4   278.4     282   269.6   278.7   279.5
 282.3   284.5   280.3   280.3     280   281.5   284.7   283.6
 292.9   290.5   293.9   292.6   292.7   292.8   294.1   293.6
 293.8   292.6   291.2   292.6   293.2   292.8     291   291.2
```

2.14.6. OUTPUTTAB - Table output

Synopsis

outputtab,*params* `ifiles ofile`

Description

This operator prints a table of all input datasets to standard output. `ifiles` is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. All input fields need to have the same horizontal grid.

The contents of the table depends on the chosen paramters. The format of each table parameter is keyname[:len]. len is the optional length of a table entry. Here is a list of all valid keynames:

Keyname	**Type**	**Description**
value	FLOAT	Value of the variable [len:8]
name	STRING	Name of the variable [len:8]
param	STRING	Parameter ID (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]]) [len:11]
code	INTEGER	Code number [len:4]
lon	FLOAT	Longitude coordinate [len:6]
lat	FLOAT	Latitude coordinate [len:6]
lev	FLOAT	Vertical level [len:6]
xind	INTEGER	Grid x index [len:4]
yind	INTEGER	Grid y index [len:4]
timestep	INTEGER	Timestep number [len:6]
date	STRING	Date (format YYYY-MM-DD) [len:10]
time	STRING	Time (format hh:mm:ss) [len:8]
year	INTEGER	Year [len:5]
month	INTEGER	Month [len:2]
day	INTEGER	Day [len:2]
nohead	INTEGER	Disable output of header line

Parameter

params STRING Comma separated list of keynames, one for each column of the table

Example

To print a table with name, date, lon, lat and value information use:

```
cdo outputtab,name,date,lon,lat,value ifile
```

Here is an example output of a time series with the yearly mean temperatur at lon=10/lat=53.5:

```
#   name        date     lon     lat     value
   tsurf  1991-12-31      10    53.5   8.83903
   tsurf  1992-12-31      10    53.5   8.17439
   tsurf  1993-12-31      10    53.5   7.90489
   tsurf  1994-12-31      10    53.5   10.0216
   tsurf  1995-12-31      10    53.5   9.07798
```

2.14.7. OUTPUTGMT - GMT output

Synopsis

<*operator*> `ifile`

Description

This module prints the first field of the input dataset to standard output. The output can be used to generate 2D Lon/Lat plots with [GMT]. The format of the output depends on the chosen operator.

Operators

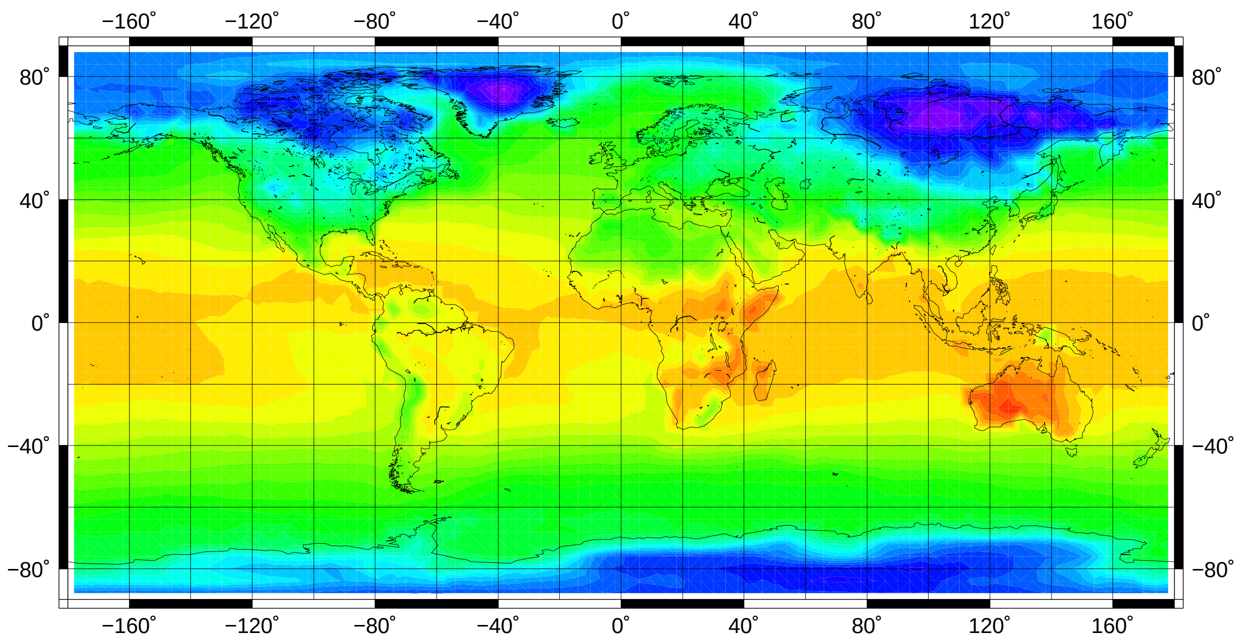
gmtxyz GMT xyz format
The operator exports the first field to the GMT xyz ASCII format. The output can be used to create contour plots with the GMT module pscontour.

gmtcells GMT multiple segment format
The operator exports the first field to the GMT multiple segment ASCII format. The output can be used to create shaded gridfill plots with the GMT module psxy.

Example

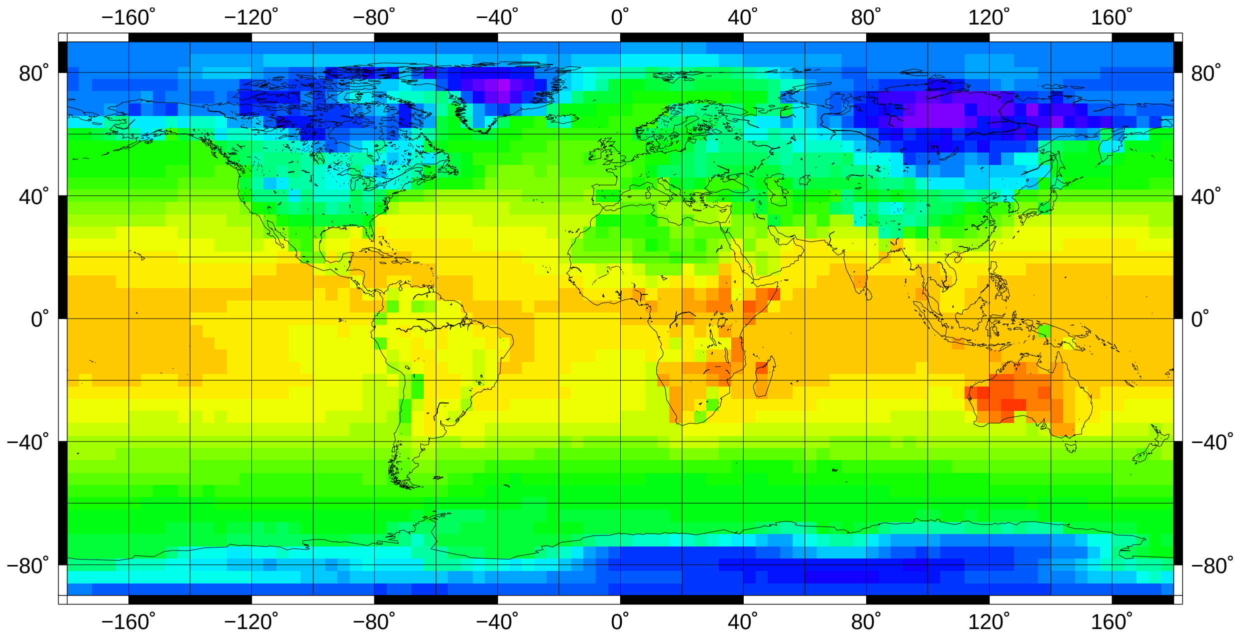
1) GMT shaded contour plot of a global temperature field with a resolution of 4 degree. The contour interval is 3 with a rainbow color table.

```
cdo gmtxyz temp > data.gmt
makecpt -T213/318/3 -Crainbow > gmt.cpt
pscontour -K -JQ0/10i -Rd -I -Cgmt.cpt data.gmt > gmtplot.ps
pscoast -O -J -R -Dc -W -B40g20 >> gmtplot.ps
```



2) GMT shaded gridfill plot of a global temperature field with a resolution of 4 degree. The contour interval is 3 with a rainbow color table.

```
cdo gmtcells temp > data.gmt
makecpt -T213/318/3 -Crainbow > gmt.cpt
psxy -K -JQ0/10i -Rd -L -Cgmt.cpt -m data.gmt > gmtplot.ps
pscoast -O -J -R -Dc -W -B40g20 >> gmtplot.ps
```



2.15. Miscellaneous

This section contains miscellaneous modules which do not fit to the other sections before.

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

gradsdes	GrADS data descriptor file
after	ECHAM standard post processor
bandpass	Bandpass filtering
lowpass	Lowpass filtering
highpass	Highpass filtering
gridarea	Grid cell area
gridweights	Grid cell weights
smooth9	9 point smoothing
setvals	Set list of old values to new values
setrtoc	Set range to constant
setrtoc2	Set range to constant others to constant2
timsort	Sort over the time
const	Create a constant field
random	Create a field with random numbers
topo	Create a field with topography
for	Create a time series
stdatm	Create values for pressure and temperature for hydrostatic atmosphere
rotuvb	Backward rotation
mastrfu	Mass stream function
sealevelpressure	Sea level pressure
adisit	Potential temperature to in-situ temperature
adipot	In-situ temperature to potential temperature
rhopot	Calculates potential density
histcount	Histogram count
histsum	Histogram sum
histmean	Histogram mean
histfreq	Histogram frequency
sethalo	Set the left and right bounds of a field
wct	Windchill temperature
fdns	Frost days where no snow index per time period
strwin	Strong wind days index per time period
strbre	Strong breeze days index per time period
strgal	Strong gale days index per time period
hurr	Hurricane days index per time period

2.15.1. GRADSDES - GrADS data descriptor file

Synopsis

gradsdes*[,mapversion]* `ifile`

Description

Creates a GrADS data descriptor file. Supported file formats are GRIB1, NetCDF, SERVICE, EXTRA and IEG. For GRIB1 files the GrADS map file is also generated. For SERVICE and EXTRA files the grid have to be specified with the CDO option '-g <grid>'. This module takes `ifile` in order to create filenames for the descriptor (`ifile.ctl`) and the map (`ifile.gmp`) file.

Parameter

mapversion INTEGER Format version of the GrADS map file for GRIB1 datasets. Use 1 for a machine specific version 1 GrADS map file, 2 for a machine independent version 2 GrADS map file and 4 to support GRIB files >2GB. A version 2 map file can be used only with GrADS version 1.8 or newer. A version 4 map file can be used only with GrADS version 2.0 or newer. The default is 4 for files >2GB, otherwise 2.

Example

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

```
cdo gradsdes ifile.grb
```

This will create a descriptor file with the name `ifile.ctl` and the map file `ifile.gmp`.

Assumed the input GRIB1 dataset has 3 variables over 12 timesteps on a Gaussian N16 grid. The contents of the resulting GrADS data description file is approximately:

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

2.15.2. AFTERBURNER - ECHAM standard post processor

Synopsis

after[,*vct*] `ifiles ofile`

Description

The "afterburner" is the standard post processor for [ECHAM] data which provides the following operations:

- Extract specified variables and levels
- Compute derived variables
- Transform spectral data to Gaussian grid representation
- Vertical interpolation to pressure levels
- Compute temporal means

This operator reads selection parameters as namelist from stdin. Use the UNIX redirection "<namelistfile" to read the namelist from file.

Namelist

Namelist parameter and there defaults:

```
TYPE=0, CODE=-1, LEVEL=-1, INTERVAL=0, MEAN=0, EXTRAPOLATE=0
```

`TYPE` controls the transformation and vertical interpolation. Transforming spectral data to Gaussian grid representation and vertical interpolation to pressure levels are performed in a chain of steps. The `TYPE` parameter may be used to stop the chain at a certain step. Valid values are:

```
TYPE  =  0 : Hybrid   level spectral coefficients
TYPE  = 10 : Hybrid   level fourier  coefficients
TYPE  = 11 : Hybrid   level zonal mean sections
TYPE  = 20 : Hybrid   level gauss grids
TYPE  = 30 : Pressure level gauss grids
TYPE  = 40 : Pressure level fourier  coefficients
TYPE  = 41 : Pressure level zonal mean sections
TYPE  = 50 : Pressure level spectral coefficients
TYPE  = 60 : Pressure level fourier  coefficients
TYPE  = 61 : Pressure level zonal mean sections
TYPE  = 70 : Pressure level gauss grids
```

Vorticity, divergence, streamfunction and velocity potential need special treatment in the vertical transformation. They are not available as types 30, 40 and 41. If you select one of these combinations, type is automatically switched to the equivalent types 70, 60 and 61. The type of all other variables will be switched too, because the type is a global parameter.

`CODE` selects the variables by the ECHAM GRIB1 code number (1-255). The default value `-1` processes all detected codes. Derived variables computed by the afterburner:

Code	Name	Longname	Level	Needed Codes/Computation
34	low_cld	low cloud	single	223 on modellevel
35	mid_cld	mid cloud	single	223 on modellevel
36	hih_cld	high cloud	single	223 on modellevel
131	u	u-velocity	atm (ml+pl)	138, 155
132	v	v-velocity	atm (ml+pl)	138, 155
135	omega	vertical velocity	atm (ml+pl)	138, 152, 155
148	stream	streamfunction	atm (ml+pl)	131, 132
149	velopot	velocity potential	atm (ml+pl)	131, 132
151	slp	mean sea level pressure	surface	129, 130, 152
156	geopoth	geopotential height	atm (ml+pl)	129, 130, 133, 152
157	rhumidity	relative humidity	atm (ml+pl)	130, 133, 152
189	sclfs	surface solar cloud forcing	surface	176-185
190	tclfs	surface thermal cloud forcing	surface	177-186
191	sclf0	top solar cloud forcing	surface	178-187
192	tclf0	top thermal cloud forcing	surface	179-188
259	windspeed	windspeed	atm (ml+pl)	sqrt(u*u+v*v)
260	precip	total precipitation	surface	142+143

`LEVEL` selects the hybrid or pressure levels. The allowed values depends on the parameter `TYPE`. The default value `-1` processes all detected levels.

`INTERVAL` selects the processing interval. The default value `0` process data on monthly intervals. `INTERVAL=1` sets the interval to daily.

`MEAN=1` compute and write monthly or daily mean fields. The default value `0` writes out all timesteps.

`EXTRAPOLATE=0` switch of the extrapolation of missing values during the interpolation from model to pressure level (only available with `MEAN=0` and `TYPE=30`). The default value `1` extrapolate missing values.

Possible combinations of TYPE, CODE and MEAN:

TYPE	CODE	MEAN
0/10/11	130 temperature	0
0/10/11	131 u-velocity	0
0/10/11	132 v-velocity	0
0/10/11	133 specific humidity	0
0/10/11	138 vorticity	0
0/10/11	148 streamfunction	0
0/10/11	149 velocity potential	0
0/10/11	152 LnPs	0
0/10/11	155 divergence	0
>11	all codes	0/1

Parameter

vct `STRING` File with VCT in ASCII format

Example

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

```
cdo after ifile ofile << EON
  TYPE=30 LEVEL=92500,85000,50000,20000
EON
```

2.15.3. FILTER - Time series filtering

Synopsis

bandpass,*fmin*,*fmax* `ifile ofile`

lowpass,*fmax* `ifile ofile`

highpass,*fmin* `ifile ofile`

Description

This module takes the time series for each gridpoint in `ifile` and (fast fourier) transforms it into the frequency domain. According to the particular operator and its parameters certain frequencies are filtered (set to zero) in the frequency domain and the spectrum is (inverse fast fourier) transformed back into the time domain. To determine the frequency the time-axis of `ifile` is used. (Data should have a constant time increment since this assumption applies for transformation. However, the time increment has to be different from zero.) All frequencies given as parameter are interpreted per year. This is done by the assumption of a 365-day calendar. Consequently if you want to perform multiyear-filtering accurately you have to delete the 29th of February. If your `ifile` has a 360 year calendar the frequency parameters *fmin* respectively *fmax* should be multiplied with a factor of 360/365 in order to obtain accurate results. For the set up of a frequency filter the frequency parameters have to be adjusted to a frequency in the data. Here *fmin* is rounded down and *fmax* is always rounded up. Consequently it is possible to use bandpass with *fmin=fmax* without getting a zero-field for `ofile`. Hints for efficient usage:

- to get reliable results the time-series has to be detrended (cdo detrend)
- the lowest frequency greater zero that can be contained in ifile is 1/(N*dT),
- the greatest frequency is 1/(2dT) (Nyquist frequency),

with N the number of timesteps and dT the time increment of `ifile` in years.

Operators

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

Parameter

fmin	FLOAT	Minimum frequency per year that passes the filter.
fmax	FLOAT	Maximum frequency per year that passes the filter.

Example

Now assume your data are still hourly for a time period of 5 years but with a 365/366-day- calendar and you want to suppress the variability on timescales greater or equal to one year (we suggest here to use a number x bigger than one (e.g. x=1.5) since there will be dominant frequencies around the peak (if there is one) as well due to the issue that the time series is not of infinite length). Therefor you can use the following:

```
cdo highpass,x -del29feb ifile ofile
```

Accordingly you might use the following to suppress variability on timescales shorter than one year:

```
cdo lowpass,1 -del29feb ifile ofile
```

Finally you might be interested in 2-year variability. If you want to suppress the seasonal cycle as well as say the longer cycles in climate system you might use

```
cdo bandpass,x,y -del29feb ifile ofile
```

with x<=0.5 and y >=0.5.

2.15.4. GRIDCELL - Grid cell quantities

Synopsis

<*operator*> `ifile ofile`

Description

This module reads the grid cell area of the first grid from the input stream. If the grid cell area is missing it will be computed from the grid description. Depending on the chosen operator the grid cell area or weights are written to the output stream.

Operators

gridarea Grid cell area
Writes the grid cell area to the output stream. If the grid cell area have to be computed it is scaled with the earth radius to square meters.

gridweights Grid cell weights
Writes the grid cell area weights to the output stream.

Environment

`PLANET_RADIUS` This variable is used to scale the computed grid cell areas to square meters. By default PLANET_RADIUS is set to an earth radius of 6371000 meter.

2.15.5. SMOOTH9 - 9 point smoothing

Synopsis

smooth9 `ifile ofile`

Description

Performs a 9 point smoothing on all fields with a quadrilateral curvilinear grid. The result at each grid point is a weighted average of the grid point plus the 8 surrounding points. The center point receives a weight of 1.0, the points at each side and above and below receive a weight of 0.5, and corner points receive a weight of 0.3. All 9 points are multiplied by their weights and summed, then divided by the total weight to obtain the smoothed value. Any missing data points are not included in the sum; points beyond the grid boundary are considered to be missing. Thus the final result may be the result of an averaging with less than 9 points.

2.15.6. REPLACEVALUES - Replace variable values

Synopsis

setvals,*oldval,newval[,...]* `ifile ofile`

setrtoc,*rmin,rmax,c* `ifile ofile`

setrtoc2,*rmin,rmax,c,c2* `ifile ofile`

Description

This module replaces old variable values with new values, depending on the operator.

Operators

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

setrtoc Set range to constant

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

setrtoc2 Set range to constant others to constant2

$$o(t,x) = \begin{cases} \text{c} & \text{if } i(t,x) \geq \mathit{rmin} \wedge i(t,x) \leq \mathit{rmax} \\ \text{c2} & \text{if } i(t,x) < \mathit{rmin} \vee i(t,x) > \mathit{rmax} \end{cases}$$

Parameter

oldval,newval,...	FLOAT	Pairs of old and new values
rmin	FLOAT	Lower bound
rmax	FLOAT	Upper bound
c	FLOAT	New value - inside range
c2	FLOAT	New value - outside range

2.15.7. TIMSORT - Timsort

Synopsis

timsort `ifile ofile`

Description

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

$o(t_1, x) <= o(t_2, x) \qquad \forall (t_1 < t_2), x$

Example

To sort all field elements of a dataset over all timesteps use:

```
cdo timsort ifile ofile
```

2.15.8. VARGEN - Generate a field

Synopsis

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

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

topo[*,grid*] `ofile`

for,*start*,*end[,inc]* `ofile`

stdatm,*levels* `ofile`

Description

Generates a dataset with one or more fields

Operators

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 numbers
Creates a field with rectangularly distrubuted random numbers in the interval [0,1].

topo Create a field with topography
Creates a field with topography data, per default on a global half degree grid.

for Create a time series
Creates a time series with field size 1 and field elements beginning with a start value in time step 1 which is increased from one time step to the next.

stdatm Create values for pressure and temperature for hydrostatic atmosphere
Creates pressure and temperature values for the given list of vertical levels. The formulars are:

$$P(z) = P_0 \exp\left(-\frac{g}{R}\frac{H}{T_0}\log\left(\frac{\exp\left(\frac{z}{H}\right)T_0 + \Delta T}{T_0 + \Delta T}\right)\right)$$

$$T(z) = T_0 + \Delta T \exp\left(-\frac{z}{H}\right)$$

with the following constants

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

This is the solution for the hydrostatic equations and is only valid for the troposphere (constant positive lapse rate). The temperature increase in the stratosphere and other effects of the upper atmosphere are not taken into account.

Parameter

const	FLOAT	Constant
seed	INTEGER	The seed for a new sequence of pseudo-random numbers [default: 1]
grid	STRING	Target grid description file or name
start	FLOAT	Start value of the loop
end	FLOAT	End value of the loop
inc	FLOAT	Increment of the loop [default: 1]
levels	FLOAT	Target levels in metre above surface

Example

To create a standard atmosphere dataset on a given horizontal grid:

```
cdo enlarge,gridfile -stdatm,10000,8000,5000,3000,2000,1000,500,200,0 ofile
```

2.15.9. ROTUVB - Rotation

Synopsis

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

Description

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

Parameter

u,*v*,... STRING Pairs of zonal and meridional velocity components (use variable names or code numbers)

Example

To transform the u and v velocity of a dataset from a rotated spherical system to a geographical system use:

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

2.15.10. MASTRFU - Mass stream function

Synopsis

mastrfu `ifile ofile`

Description

This is a special operator for the post processing of the atmospheric general circulation model ECHAM. It computes the mass stream function (code=272). The input dataset have to be a zonal mean of v-velocity [m/s] (code=132) on pressure levels.

Example

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

```
cdo mastrfu ifile ofile
```

2.15.11. DERIVEPAR - Sea level pressure

Synopsis

sealevelpressure `ifile ofile`

Description

This operator computes the sea level pressure (air_pressure_at_sea_level). Required input fields are surface_air_pressure, surface_geopotential and air_temperature on hybrid sigma pressure levels.

2.15.12. ADISIT - Potential temperature to in-situ temperature and vice versa

Synopsis

adisit*[,pressure]* `ifile ofile`

adipot `ifile ofile`

Description

Operators

adisit Potential temperature to in-situ temperature
This is a special operator for the post processing of the ocean and sea ice model output. It converts potential temperature adiabatically to in-situ temperature to(t, s, p). Required input fields are sea water potential temperature (name=tho; code=2) and sea water salinity (name=sao; code=5). Pressure is calculated from the level information or can be specified by the optional parameter. Output fields are sea water temperature (name=to; code=20) and sea water salinity (name=s; code=5).

adipot In-situ temperature to potential temperature
This is a special operator for the post processing of the ocean and sea ice model outpu. It converts in-situ temperature to potential temperature tho(to, s, p). Required input fields are sea water in-situ temperature (name=t; code=2) and sea water salinity (name=sao,s; code=5). Pressure is calculated from the level information or can be specified by the optional parameter. Output fields are sea water temperature (name=tho; code=2) and sea water salinity (name=s; code=5).

Parameter

pressure	FLOAT	Pressure in bar (constant value assigned to all levels)

2.15.13. RHOPOT - Calculates potential density

Synopsis

rhopot*[,pressure]* `ifile ofile`

Description

This is a special operator for the post processing of the ocean and sea ice model MPIOM. It calculates the sea water potential density (name=rhopoto; code=18). Required input fields are sea water in-situ temperature (name=to; code=20) and sea water salinity (name=sao; code=5). Pressure is calculated from the level information or can be specified by the optional parameter.

Parameter

pressure	FLOAT	Pressure in bar (constant value assigned to all levels)

Example

To compute the sea water potential density from the potential temperature use this operator in combination with adisit:

```
cdo rhopot -adisit ifile ofile
```

2.15.14. HISTOGRAM - Histogram

Synopsis

<operator>,bounds `ifile ofile`

Description

This module creates bins for a histogram of the input data. The bins have to be adjacent and have non-overlapping intervals. The user has to define the bounds of the bins. The first value is the lower bound and the second value the upper bound of the first bin. The bounds of the second bin are defined by the second and third value, aso. Only 2-dimensional input fields are allowed. The ouput file contains one vertical level for each of the bins requested.

Operators

histcount	Histogram count Number of elements in the bin range.
histsum	Histogram sum Sum of elements in the bin range.
histmean	Histogram mean Mean of elements in the bin range.
histfreq	Histogram frequency Relative frequency of elements in the bin range.

Parameter

bounds	FLOAT	Comma separated list of the bin bounds (-inf and inf valid)

2.15.15. SETHALO - Set the left and right bounds of a field

Synopsis

sethalo,*lhalo,rhalo* `ifile ofile`

Description

This operator sets the left and right bounds of the rectangularly understood fields. Positive numbers of the parameter *lhalo* enlarges the left bound by the given number of columns from the right bound. The parameter *rhalo* does the similar for the right bound. Negative numbers of the parameter *lhalo*/*rhalo* can be used to remove the given number of columns of the left and right bounds.

Parameter

lhalo	INTEGER	Left halo
rhalo	INTEGER	Right halo

2.15.16. WCT - Windchill temperature

Synopsis

wct `ifile1 ifile2 ofile`

Description

Let `ifile1` and `ifile2` be time series of temperature and wind speed records, then a corresponding time series of resulting windchill temperatures is written to `ofile`. The wind chill temperature calculation is only valid for a temperature of T <= 33 °C and a wind speed of v >= 1.39 m/s. Whenever these conditions are not satisfied, a missing value is written to `ofile`. Note that temperature and wind speed records have to be given in units of °C and m/s, respectively.

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

Synopsis

fdns `ifile1 ifile2 ofile`

Description

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

2.15.18. STRWIN - Strong wind days index per time period

Synopsis

strwin*[,v]* `ifile ofile`

Description

Let `ifile` be a time series of the daily maximum horizontal wind speed VX, then the number of days where VX > v is counted. The horizontal wind speed v is an optional parameter with default v = 10.5 m/s. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to v. Note that both VX and v have to be given in units of m/s. Also note that the horizontal wind speed is defined as the square root of the sum of squares of the zonal and meridional wind speeds. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile`.

Parameter

v	FLOAT	Horizontal wind speed threshold (m/s, default v = 10.5 m/s)

2.15.19. STRBRE - Strong breeze days index per time period

Synopsis

strbre `ifile ofile`

Description

Let `ifile` be a time series of the daily maximum horizontal wind speed VX, then the number of days where VX is greater than or equal to 10.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 10.5 m/s. Note that VX is defined as the square root of the sum of squares of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile`.

2.15.20. STRGAL - Strong gale days index per time period

Synopsis

strgal `ifile ofile`

Description

Let `ifile` be a time series of the daily maximum horizontal wind speed VX, then the number of days where VX is greater than or equal to 20.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 20.5 m/s. Note that VX is defined as the square root of the sum of square of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile`.

2.15.21. HURR - Hurricane days index per time period

Synopsis

hurr `ifile ofile`

Description

Let `ifile` be a time series of the daily maximum horizontal wind speed VX, then the number of days where VX is greater than or equal to 32.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 32.5 m/s. Note that VX is defined as the square root of the sum of squares of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile`.

Bibliography

[CDI]
Climate Data Interface, from the Max Planck Institute for Meteorologie

[CM-SAF]
Satellite Application Facility on Climate Monitoring, from the German Weather Service (Deutscher Wetterdienst, DWD)

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

[GMT]
The Generic Mapping Tool, from the School of Ocean and Earth Science and Technology (SOEST)

[GrADS]
Grid Analysis and Display System, from the Center for Ocean-Land-Atmosphere Studies (COLA)

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

[GRIBAPI]
GRIB API decoding/encoding, from the European Centre for Medium-Range Weather Forecasts (ECMWF)

[HDF5]
HDF version 5, from the HDF Group

[INTERA]
INTERA Software Package, from the Max Planck Institute for Meteorologie

[Magics]
Magics Software Package, from the European Centre for Medium-Range Weather Forecasts (ECMWF)

[MPIOM]
Ocean and sea ice model, from the Max Planck Institute for Meteorologie

[NetCDF]
NetCDF Software Package, from the UNIDATA Program Center of the University Corporation for Atmospheric Research

[PINGO]
The PINGO package, from the Model & Data group at the Max Planck Institute for Meteorologie

[REMO]
Regional Model, from the Max Planck Institute for Meteorologie

[Peisendorfer]
Rudolph W. Peisendorfer: Principal Component Analysis, Elsevier (1988)

[PROJ.4]
Cartographic Projections Library, originally written by Gerald Evenden then of the USGS.

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

[szip]
Szip compression software, developed at University of New Mexico.

[vonStorch]
Hans von Storch, Walter Zwiers: Statistical Analysis in Climate Research, Cambridge University Press (1999)

[YAC]
YAC - Yet Another Coupler Software Package, from DKRZ and MPI for Meteorologie

A. Environment Variables

The following table describes the environment variables that affect **CDO**.

Variable name	Default	Description
CDO_FILE_SUFFIX	None	Default file suffix. This suffix will be added to the output file name instead of the filename extension derived from the file format. NULL will disable the adding of a file suffix.
CDO_HISTORY_INFO	1	Append NetCDF global attribute histroy
CDO_PCTL_NBINS	101	Number of histogram bins.
CDO_RESET_HISTORY	0	Set to 1 to reset the NetCDF *history* global attribute.
CDO_REMAP_NORM	fracarea	Choose the normalization for the conservative interpolation
CDO_GRIDSEARCH_RADIUS	180	Grid search radius in degree. Used by the operators setmisstonn, remapdis and remapnn.
CDO_TIMESTAT_DATE	None	Set the date information of a time statistic operator to the "first", "middle" or "last" contributing timestep.
CDO_USE_FFTW	1	Set to 0 to switch off usage of FFTW. Used in the Filter module.
CDO_VERSION_INFO	1	Set to 0 to disable NetCDF global attribute CDO

B. Parallelized operators

Some of the **CDO** operators are parallelized with OpenMP. To use **CDO** with multiple OpenMP threads, you have to set the number of threads with the option '-P'. Here is an example to distribute the bilinear interpolation on 8 OpenMP threads:

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

The following **CDO** operators are parallelized with OpenMP:

Module	Operator	Description
Detrend	detrend	Detrend
Ensstat	ensmin	Ensemble minimum
Ensstat	ensmax	Ensemble maximum
Ensstat	enssum	Ensemble sum
Ensstat	ensmean	Ensemble mean
Ensstat	ensavg	Ensemble average
Ensstat	ensvar	Ensemble variance
Ensstat	ensstd	Ensemble standard deviation
Ensstat	enspctl	Ensemble percentiles
Filter	bandpass	Bandpass filtering
Filter	lowpass	Lowpass filtering
Filter	highpass	Highpass filtering
Fourier	fourier	Fourier transformation
Genweights	genbil	Generate bilinear interpolation weights
Genweights	genbic	Generate bicubic interpolation weights
Genweights	gendis	Generate distance-weighted average remap weights
Genweights	gennn	Generate nearest neighbor remap weights
Genweights	gencon	Generate 1st order conservative remap weights
Genweights	gencon2	Generate 2nd order conservative remap weights
Genweights	genlaf	Generate largest area fraction remap weights
Gridboxstat	gridboxmin	Gridbox minimum
Gridboxstat	gridboxmax	Gridbox maximum
Gridboxstat	gridboxsum	Gridbox sum
Gridboxstat	gridboxmean	Gridbox mean
Gridboxstat	gridboxavg	Gridbox average
Gridboxstat	gridboxvar	Gridbox variance
Gridboxstat	gridboxstd	Gridbox standard deviation
Remapeta	remapeta	Remap vertical hybrid level
Remap	remapbil	Bilinear interpolation
Remap	remapbic	Bicubic interpolation
Remap	remapdis	Distance-weighted average remapping
Remap	remapnn	Nearest neighbor remapping
Remap	remapcon	First order conservative remapping
Remap	remapcon2	Second order conservative remapping
Remap	remaplaf	Largest area fraction remapping

C. Standard name table

The following CF standard names are supported by **CDO**.

CF standard name	Units	**GRIB 1 code**	**variable name**
surface_geopotential	m2 s-2	129	geosp
air_temperature	K	130	ta
specific_humidity	1	133	hus
surface_air_pressure	Pa	134	aps
air_pressure_at_sea_level	Pa	151	psl
geopotential_height	m	156	zg

D. Grid description examples

D.1. Example of a curvilinear grid description

Here is an example for the **CDO** description of a curvilinear grid. xvals/yvals describe the positions of the 6x5 quadrilateral grid cells. The first 4 values of xbounds/ybounds are the corners of the first grid cell.

```
gridtype  = curvilinear
gridsize  = 30
xsize     = 6
ysize     = 5
xvals     = -21  -11    0   11   21   30  -25  -13    0   13
             25   36  -31  -16    0   16   31   43  -38  -21
              0   21   38   52  -51  -30    0   30   51   64
xbounds   = -23  -14  -17  -28       -14   -5   -6  -17        -5    5    6   -6
              5   14   17    6        14   23   28   17        23   32   38   28
            -28  -17  -21  -34       -17   -6   -7  -21        -6    6    7   -7
              6   17   21    7        17   28   34   21        28   38   44   34
            -34  -21  -27  -41       -21   -7   -9  -27        -7    7    9   -9
              7   21   27    9        21   34   41   27        34   44   52   41
            -41  -27  -35  -51       -27   -9  -13  -35        -9    9   13  -13
              9   27   35   13        27   41   51   35        41   52   63   51
            -51  -35  -51  -67       -35  -13  -21  -51       -13   13   21  -21
             13   35   51   21        35   51   67   51        51   63   77   67
yvals     =  29   32   32   32   29   26   39   42   42   42
             39   35   48   51   52   51   48   43   57   61
             62   61   57   51   65   70   72   70   65   58
ybounds   =  23   26   36   32        26   27   37   36        27   27   37   37
             27   26   36   37        26   23   32   36        23   19   28   32
             32   36   45   41        36   37   47   45        37   37   47   47
             37   36   45   47        36   32   41   45        32   28   36   41
             41   45   55   50        45   47   57   55        47   47   57   57
             47   45   55   57        45   41   50   55        41   36   44   50
             50   55   64   58        55   57   67   64        57   57   67   67
             57   55   64   67        55   50   58   64        50   44   51   58
             58   64   72   64        64   67   77   72        67   67   77   77
             67   64   72   77        64   58   64   72        58   51   56   64
```

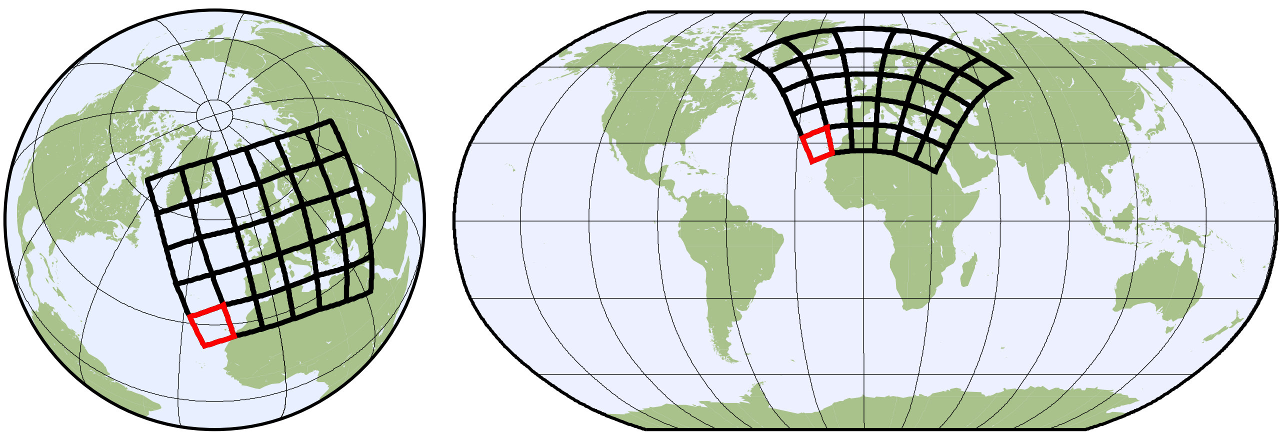


Figure D.1.: Orthographic and Robinson projection of the curvilinear grid, the first grid cell is colored red

D.2. Example description for an unstructured grid

Here is an example of the **CDO** description for an unstructured grid. xvals/yvals describe the positions of 30 independent hexagonal grid cells. The first 6 values of xbounds/ybounds are the corners of the first grid cell. The grid cell corners have to rotate counterclockwise. The first grid cell is colored red.

```
gridtype  = unstructured
gridsize  = 30
nvertex   = 6
xvals     =  -36   36    0  -18   18  108   72   54   90  180  144  126  162 -108 -144
            -162 -126  -72  -90  -54    0   72   36  144  108 -144  180  -72 -108  -36
xbounds   =  339    0    0  288  288  309        21   51   72   72    0    0
               0   16   21    0  339  344       340    0   -0  344  324  324
              20   36   36   16    0    0        93  123  144  144   72   72
              72   88   93   72   51   56        52   72   72   56   36   36
              92  108  108   88   72   72       165  195  216  216  144  144
             144  160  165  144  123  128       124  144  144  128  108  108
             164  180  180  160  144  144       237  267  288  288  216  216
             216  232  237  216  195  200       196  216  216  200  180  180
             236  252  252  232  216  216       288  304  309  288  267  272
             268  288  288  272  252  252       308  324  324  304  288  288
             345  324  324   36   36   15        36   36  108  108   87   57
              20   15   36   57   52   36       108  108  180  180  159  129
              92   87  108  129  124  108       180  180  252  252  231  201
             164  159  180  201  196  180       252  252  324  324  303  273
             236  231  252  273  268  252       308  303  324  345  340  324
yvals     =   58   58   32    0    0   58   32    0    0   58   32    0    0   58   32
               0    0   32    0    0  -58  -58  -32  -58  -32  -58  -32  -58  -32  -32
ybounds   =   41   53   71   71   53   41        41   41   53   71   71   53
              11   19   41   53   41   19       -19   -7   11   19    7  -11
             -19  -11    7   19   11   -7        41   41   53   71   71   53
              11   19   41   53   41   19       -19   -7   11   19    7  -11
             -19  -11    7   19   11   -7        41   41   53   71   71   53
              11   19   41   53   41   19       -19   -7   11   19    7  -11
             -19  -11    7   19   11   -7        41   41   53   71   71   53
              11   19   41   53   41   19       -19   -7   11   19    7  -11
             -19  -11    7   19   11   -7        11   19   41   53   41   19
             -19   -7   11   19    7  -11       -19  -11    7   19   11   -7
             -41  -53  -71  -71  -53  -41       -53  -71  -71  -53  -41  -41
             -19  -41  -53  -41  -19  -11       -53  -71  -71  -53  -41  -41
             -19  -41  -53  -41  -19  -11       -53  -71  -71  -53  -41  -41
             -19  -41  -53  -41  -19  -11       -53  -71  -71  -53  -41  -41
             -19  -41  -53  -41  -19  -11       -19  -41  -53  -41  -19  -11
```

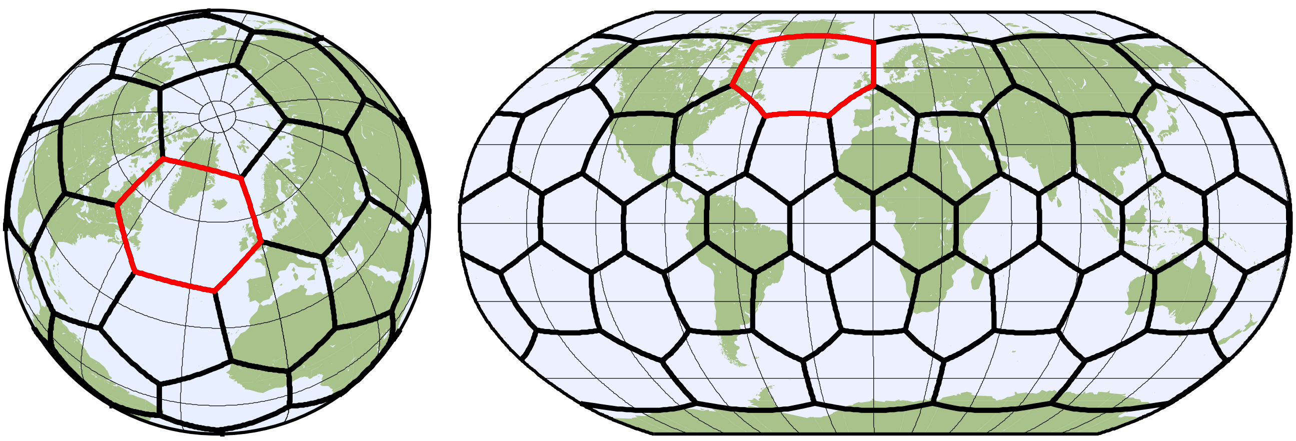


Figure D.2.: Orthographic and Robinson projection of the unstructured grid

Operator index

Z