## Contents

### 1. Introduction
1.1. Building from sources .............................................. 6
   1.1.1. Compilation .................................................. 7
   1.1.2. Installation ................................................. 7
1.2. Usage .................................................................. 7
   1.2.1. Options .......................................................... 8
   1.2.2. Operators ....................................................... 8
   1.2.3. Combining operators ......................................... 9
   1.2.4. Operator parameter ......................................... 9
1.3. Horizontal grids ..................................................... 9
   1.3.1. Grid area weights ............................................. 10
   1.3.2. Grid description ............................................... 10
1.4. Z-axis description ................................................... 13
1.5. Time axis ................................................................ 14
   1.5.1. Absolute time ................................................... 14
   1.5.2. Relative time .................................................... 14
   1.5.3. Conversion of the time ....................................... 14
1.6. Parameter table ....................................................... 14
1.7. Missing values ....................................................... 15
   1.7.1. Mean and average .............................................. 15

### 2. Reference manual .................................................. 16
2.1. Information .......................................................... 17
   2.1.1. INFO - Information and simple statistics ................. 18
   2.1.2. SINFO - Short information .................................. 19
   2.1.3. DIFF - Compare two datasets field by field .............. 20
   2.1.4. NINFO - Print the number of parameters, levels or times 21
   2.1.5. SHOWINFO - Show variables, levels or times .......... 22
   2.1.6. FILEDES - Dataset description ............................. 23
2.2. File operations ...................................................... 24
   2.2.1. COPY - Copy datasets ....................................... 25
   2.2.2. REPLACE - Replace variables ............................... 26
   2.2.3. MERGE - Merge datasets .................................... 27
   2.2.4. SPLIT - Split a dataset ...................................... 28
   2.2.5. SPLITTIME - Split timesteps of a dataset ............... 30
   2.2.6. SPLITSEL - Split selected timesteps ...................... 31
2.3. Selection ................................................................ 32
   2.3.1. SELVAR - Select fields ...................................... 33
   2.3.2. SELTIME - Select timesteps ................................. 36
   2.3.3. SELBOX - Select a box of a field ......................... 38
2.4. Conditional selection ............................................... 39
   2.4.1. COND - Conditional select one field ....................... 40
   2.4.2. COND2 - Conditional select two fields ................. 40
   2.4.3. CONDC - Conditional select a constant .................. 41
2.5. Comparison .......................................................... 42
   2.5.1. COMP - Comparison of two fields ........................ 43
   2.5.2. COMPC - Comparison of a field with a constant ....... 44
2.6. Modification ........................................................ 45
   2.6.1. SET - Set field info .......................................... 47
2.8. Statistical values

2.8.1. CONSECSTAT - Consecutive timestep periods
2.8.2. ENSSSTAT - Statistical values over an ensemble
2.8.3. ENSSSTAT2 - Statistical values over an ensemble
2.8.4. ENSVAL - Ensemble validation tools
2.8.5. FLDSTAT - Statistical values over a field
2.8.6. ZONSTAT - Zonal statistical values
2.8.7. MERSTAT - Meridional statistical values
2.8.8. GRIDBOXSTAT - Statistical values over grid boxes
2.8.9. VERTSTAT - Vertical statistical values
2.8.10. TIMSELEXSTAT - Time range statistical values
2.8.11. TIMSELEXPCTL - Time range percentile values
2.8.12. RUNSTAT - Running statistical values
2.8.13. RUNPCTL - Running percentile values
2.8.14. TIMSTAT - Statistical values over all timesteps
2.8.15. TIMPCTL - Percentile values over all timesteps
2.8.16. HOURSTAT - Hourly statistical values
2.8.17. HOURPCTL - Hourly percentile values
2.8.18. DAYSTAT - Daily statistical values
2.8.19. DAYPCTL - Daily percentile values
2.8.20. MONSTAT - Monthly statistical values
2.8.21. MONPCTL - Monthly percentile values
2.8.22. YEARSTAT - Yearly statistical values
2.8.23. YEARPCTL - Yearly percentile values
2.8.24. SEASSTAT - Seasonal statistical values
2.8.25. SEASPCTL - Seasonal percentile values
2.8.26. YHOURSTAT - Multi-year hourly statistical values
2.8.27. YDAYSTAT - Multi-year daily statistical values
2.8.28. YMONSTAT - Multi-year monthly statistical values
2.8.29. YMONPCTL - Multi-year monthly percentile values
2.8.30. YDAYPCTL - Multi-year daily percentile values
2.8.31. YSEASSTAT - Multi-year seasonal statistical values
2.8.32. YSEASPCTL - Multi-year seasonal percentile values
2.8.33. YDRUNSTAT - Multi-year daily running statistical values
2.8.34. YDRUNPCTL - Multi-year daily running percentile values
## Contents

2.9. Correlation and co.

2.9.1. FLDCOR - Correlation in grid space ........................................ 120
2.9.2. TIMCOR - Correlation over time ............................................. 120
2.9.3. FLDCOVAR - Covariance in grid space ..................................... 121
2.9.4. TIMCOVAR - Covariance over time .......................................... 121

2.10. Regression

2.10.1. REGRES - Regression .......................................................... 123
2.10.2. DETREND - Detrend time series ........................................... 123
2.10.3. TRENDB - Trend of time series ............................................ 124
2.10.4. SUBTREND - Subtract a trend .............................................. 124

2.11. EOFs

2.11.1. EOFS - Empirical Orthogonal Functions ................................... 126
2.11.2. EOFCOEFF - Principal coefficients of EOFs ............................. 128

2.12. Interpolation

2.12.1. REMAPGRID - SCRIP grid interpolation .................................... 130
2.12.2. GENWEIGHTS - Generate SCRIP grid interpolation weights ............ 132
2.12.3. REMAP - SCRIP grid remapping ............................................. 134
2.12.4. REMAPETA - Remap vertical hybrid level ................................ 135
2.12.5. INTVERT - Vertical interpolation .......................................... 137
2.12.6. INTLEVEL - Linear level interpolation ..................................... 138
2.12.7. INTLEVEL3D - Linear level interpolation from/to 3d vertical coordinates 139
2.12.8. INTTIME - Time interpolation .............................................. 140
2.12.9. INTYEAR - Year interpolation .............................................. 141

2.13. Transformation

2.13.1. SPECTRAL - Spectral transformation ...................................... 143
2.13.2. WIND - Wind transformation ................................................ 145


2.14.1. IMPORTBINARY - Import binary data sets ................................. 147
2.14.2. IMPORTCM5SAF - Import CM-SAF HDF5 files .............................. 148
2.14.3. IMPORTAMSR - Import AMSR binary files .................................. 149
2.14.4. INPUT - Formatted input ...................................................... 150
2.14.5. OUTPUT - Formatted output .................................................. 151

2.15. Miscellaneous

2.15.1. GRADSDES - GrADS data descriptor file ................................... 154
2.15.2. FILTER - Time series filtering .............................................. 155
2.15.3. GRIDCELL - Grid cell quantities .......................................... 156
2.15.4. SMOOTH9 - 9 point smoothing .............................................. 157
2.15.5. REPLACEVALUES - Replace variable values .............................. 157
2.15.6. TIMSORT - Timsort ............................................................ 158
2.15.7. VARGEN - Generate a field .................................................. 158
2.15.8. ROTUV - Rotation ............................................................. 160
2.15.9. MASTRFU - Mass stream function ......................................... 160
2.15.10. HISTOGRAM - Histogram .................................................... 161
2.15.11. SETHALO - Set the left and right bounds of a field .................... 161
2.15.12. WCT - Windchill temperature .............................................. 162
2.15.13. FDNS - Frost days where no snow index per time period ............. 162
2.15.14. STRWIN - Strong wind days index per time period ..................... 162
2.15.15. STRBRE - Strong breeze days index per time period ................... 163
2.15.16. STRGAL - Strong gale days index per time period ..................... 163
2.15.17. HURR - Hurricane days index per time period ........................... 163

2.16. Climate indices

2.16.1. ECACDD - Consecutive dry days index per time period ................ 166
2.16.2. ECACFD - Consecutive frost days index per time period ............... 166
2.16.3. ECACSU - Consecutive summer days index per time period ............ 167
2.16.4. ECACWD - Consecutive wet days index per time period ................. 167
2.16.5. ECACWDI - Cold wave duration index w.r.t. mean of reference period 169
2.16.6. ECACWFI - Cold-spell days index w.r.t. 10th percentile of reference period 169
2.16.7. ECAETR - Intra-period extreme temperature range ....................... 171
1. Introduction

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

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

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

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

The main CDO features are:

- More than 400 operators available
- Modular design and easily extendable with new operators
- Very simple UNIX command line interface
- A dataset can be processed by several operators, without storing the interim results in files
- Most operators handle datasets with missing values
- Fast processing of large datasets
- Support of many different grid types
- Tested on many UNIX/Linux systems, Cygwin, and MacOS-X

1.1. Building from sources

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

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

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

- Unidata netCDF library (http://www.unidata.ucar.edu/packages/netcdf) version 3 or higher. This is needed to process netCDF [netCDF] files with CDO.
- The ECMWF GRIB_API (http://www.ecmwf.int/products/data/software/grib_api.html) version 1.9.5 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 is needed to process szip compressed GRIB [GRIB] files with CDO.
- HDF5 library (http://www.hdfgroup.org/HDF5) version 1.6 or higher. This is needed to import CM-SAF [CM-SAF] HDF5 files with the CDO operator import_cmsaf.
Introduction

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

CDO is a multi-threaded application. Therefore 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>
     ```
   - The GRIB2 configuration depends on the GRIB_API installation! Here is an example GRIB2 configuration with a JASPER enabled GRIB_API version:

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

   For an overview of other configuration options use

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

3. Compile the program by running make:

   ```
   make
   ```

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

1.1.2. Installation

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

```make install
```

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

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

1.2. Usage

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

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:

<table>
<thead>
<tr>
<th>&lt;format&gt;</th>
<th>&lt;nbits&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>grb, grb2</td>
<td>P1 - P24</td>
</tr>
<tr>
<td>nc, nc2, nc4, nc4c</td>
<td>I8/I16/I32/F32/F64</td>
</tr>
<tr>
<td>grb2, srv, ext, ieg</td>
<td>F32/F64</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>File format</th>
<th>&lt;format&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRIB version 1</td>
<td>grb</td>
</tr>
<tr>
<td>GRIB version 2</td>
<td>grb2</td>
</tr>
<tr>
<td>netCDF</td>
<td>nc</td>
</tr>
<tr>
<td>netCDF version 2 (64-bit)</td>
<td>nc2</td>
</tr>
<tr>
<td>netCDF-4 (HDF5)</td>
<td>nc4</td>
</tr>
<tr>
<td>netCDF-4 classic</td>
<td>nc4c</td>
</tr>
<tr>
<td>SERVICE</td>
<td>srv</td>
</tr>
<tr>
<td>EXTRA</td>
<td>ext</td>
</tr>
<tr>
<td>IEG</td>
<td>ieg</td>
</tr>
</tbody>
</table>

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

- **-g <grid>** Define the default grid description by name or from file (see chapter 1.3 on page 10).
  Available grid names are: r<NX>x<NY>, lon=<LON>/lat=<LAT>, n<NI>.

- **-h** Help information for the operators.
- **-M** Switch to indicate that the I/O streams have missing values.
- **-m <missval>** Set the default missing value (default: -9e+33).
- **-O** Overwrite existing output file, if checked.
  Existing output file is checked only for: ens<STAT>, merge, mergetime
- **-P <nthreads>** Set number of OpenMP threads.
- **-Q** Alphanumeric sorting of netCDF parameter names.
- **-R** Convert GRIB1 data from reduced to regular grid (only with cgribex lib).
- **-r** Generate a relative time axis.
- **-s** Silent mode.
- **-t <partab>** Set the default parameter table name or file (see chapter 1.6 on page 14).
  Predefined tables are: echam4 echam5 echam6 mpiom1 ecmwf remo
- **-V** Print the version number.
- **-v** Print extra details for some operators.
- **-z szip jpeg zip[1-9]** SZIP compression of GRIB1 records.
  JPEG compression of GRIB2 records.
  Deflate compression of netCDF4 variables.

1.2.2. Operators

There are more than 400 operators available. A detailed description of all operators can be found in the **Reference Manual** section.
1.2.3. Combining operators

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

- reducing unnecessary disk I/O
- parallel processing

Use

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

instead of

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

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

1.2.4. Operator parameter

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

- **STRING**
  
  Unquoted characters without blanks and tabs. The following command select variables with the name `pressure` and `tsurf`:
  
  ```
cdo selvar,pressure,tsurf ifile ofile
  ```

- **FLOAT**
  
  Floating point number in any representation. The following command sets the range between 0 and 273.15 of all fields to missing value:
  
  ```
cdo setrtomiss,0,273.15 ifile ofile
  ```

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

  The result is the same as:
  
  ```
cdo selday,5,6,7,8,9 ifile ofile
  ```

1.3. Horizontal grids

Physical quantities of climate models are typically stored on a horizontal grid.
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.:

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

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

1.3.2. Grid description

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

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

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

1.3.2.1. Predefined grids

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

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

`r<NX>x<NY>` defines a global regular lon/lat grid. The number of the longitudes `<NX>` and the latitudes `<NY>` can be selected at will. The longitudes start at 0° with an increment of `(360/<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:
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gridtype</td>
<td>STRING</td>
<td>Type of the grid (gaussian, lonlat, curvilinear, unstructured).</td>
</tr>
<tr>
<td>gridsize</td>
<td>INTEGER</td>
<td>Size of the grid.</td>
</tr>
<tr>
<td>xsize</td>
<td>INTEGER</td>
<td>Size in x direction (number of longitudes).</td>
</tr>
<tr>
<td>ysize</td>
<td>INTEGER</td>
<td>Size in y direction (number of latitudes).</td>
</tr>
<tr>
<td>xvals</td>
<td>FLOAT ARRAY</td>
<td>X values of the grid cell center.</td>
</tr>
<tr>
<td>yvals</td>
<td>FLOAT ARRAY</td>
<td>Y values of the grid cell center.</td>
</tr>
<tr>
<td>xnpole</td>
<td>FLOAT</td>
<td>X value of the north pole (rotated grid).</td>
</tr>
<tr>
<td>ynpole</td>
<td>FLOAT</td>
<td>Y value of the north pole (rotated grid).</td>
</tr>
<tr>
<td>nvertex</td>
<td>INTEGER</td>
<td>Number of the vertices for all grid cells.</td>
</tr>
<tr>
<td>xbounds</td>
<td>FLOAT ARRAY</td>
<td>X bounds of each gridbox.</td>
</tr>
<tr>
<td>ybounds</td>
<td>FLOAT ARRAY</td>
<td>Y bounds of each gridbox.</td>
</tr>
<tr>
<td>xfirst, xinc</td>
<td>FLOAT, FLOAT</td>
<td>Macros to define xvals with a constant increment,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xfirst is the x value of the first grid cell center.</td>
</tr>
<tr>
<td>yfirst, yinc</td>
<td>FLOAT, FLOAT</td>
<td>Macros to define yvals with a constant increment,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>yfirst is the y value of the first grid cell center.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>gridtype</th>
<th>lonlat</th>
<th>gaussian</th>
<th>curvilinear</th>
<th>unstructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>gridsize</td>
<td>xsize*ysize</td>
<td>xsize*ysize</td>
<td>xsize*ysize</td>
<td>ncell</td>
</tr>
<tr>
<td>xsize</td>
<td>nlon</td>
<td>nlon</td>
<td>nlon</td>
<td>gridsize</td>
</tr>
<tr>
<td>ysize</td>
<td>nlat</td>
<td>nlat</td>
<td>nlat</td>
<td>gridsize</td>
</tr>
<tr>
<td>xvals</td>
<td>xsize</td>
<td>xsize</td>
<td>gridsize</td>
<td>gridsize</td>
</tr>
<tr>
<td>yvals</td>
<td>ysize</td>
<td>ysize</td>
<td>gridsize</td>
<td>gridsize</td>
</tr>
<tr>
<td>xnpole</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ynpole</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nvertex</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>nv</td>
</tr>
<tr>
<td>xbounds</td>
<td>2*xsize</td>
<td>2*xsize</td>
<td>4* gridsize</td>
<td>nv* gridsize</td>
</tr>
<tr>
<td>ybounds</td>
<td>2*ysize</td>
<td>2*ysize</td>
<td>4* gridsize</td>
<td>nv* gridsize</td>
</tr>
</tbody>
</table>

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:

```
grids = 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:

```
grids = 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:**

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

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

### 1.4. Z-axis description

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

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

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

Available z-axis types:

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

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

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

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

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

1.5. Time axis

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

1.5.1. Absolute time

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

1.5.2. Relative time

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

1.5.3. Conversion of the time

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

1.6. Parameter table

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

Example of a CDO parameter table:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>134</td>
<td>aps</td>
<td>surface pressure [Pa]</td>
<td>141</td>
<td>sn</td>
<td>snow depth [m]</td>
<td>147</td>
<td>ahfl</td>
</tr>
<tr>
<td>172</td>
<td>slm</td>
<td>land sea mask</td>
<td>175</td>
<td>albedo</td>
<td>surface albedo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>211</td>
<td>siced</td>
<td>ice depth [m]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 `--n <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 the missing value results in $miss$.

<table>
<thead>
<tr>
<th>Operation</th>
<th>b</th>
<th>$a$</th>
<th>$miss$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>addition</strong></td>
<td>$b$</td>
<td>$a + b$</td>
<td>$miss$</td>
</tr>
<tr>
<td></td>
<td>$miss$</td>
<td>$miss$</td>
<td>$miss$</td>
</tr>
<tr>
<td><strong>subtraction</strong></td>
<td>$b$</td>
<td>$a - b$</td>
<td>$miss$</td>
</tr>
<tr>
<td></td>
<td>$miss$</td>
<td>$miss$</td>
<td>$miss$</td>
</tr>
<tr>
<td><strong>multiplication</strong></td>
<td>$b$</td>
<td>$0$</td>
<td>$miss$</td>
</tr>
<tr>
<td></td>
<td>$a$</td>
<td>$a \cdot b$</td>
<td>$0$</td>
</tr>
<tr>
<td></td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td></td>
<td>$miss$</td>
<td>$miss$</td>
<td>$miss$</td>
</tr>
<tr>
<td><strong>division</strong></td>
<td>$b$</td>
<td>$0$</td>
<td>$miss$</td>
</tr>
<tr>
<td></td>
<td>$a$</td>
<td>$a/b$</td>
<td>$miss$</td>
</tr>
<tr>
<td></td>
<td>$0$</td>
<td>$0$</td>
<td>$miss$</td>
</tr>
<tr>
<td></td>
<td>$miss$</td>
<td>$miss$</td>
<td>$miss$</td>
</tr>
<tr>
<td><strong>maximum</strong></td>
<td>$b$</td>
<td>$\max(a, b)$</td>
<td>$a$</td>
</tr>
<tr>
<td></td>
<td>$miss$</td>
<td>$b$</td>
<td>$miss$</td>
</tr>
<tr>
<td><strong>minimum</strong></td>
<td>$b$</td>
<td>$\min(a, b)$</td>
<td>$a$</td>
</tr>
<tr>
<td></td>
<td>$miss$</td>
<td>$b$</td>
<td>$miss$</td>
</tr>
<tr>
<td><strong>sum</strong></td>
<td>$b$</td>
<td>$a + b$</td>
<td>$miss$</td>
</tr>
<tr>
<td></td>
<td>$miss$</td>
<td>$b$</td>
<td>$miss$</td>
</tr>
</tbody>
</table>

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. \texttt{log}, \texttt{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.
2. Reference manual

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

\[ i(t) \quad \text{Timestep } t \text{ of ifile} \]
\[ i(t,x) \quad \text{Element number } x \text{ of the field at timestep } t \text{ of ifile} \]
\[ o(t) \quad \text{Timestep } t \text{ of ofile} \]
\[ o(t,x) \quad \text{Element number } x \text{ of the field at timestep } t \text{ of ofile} \]
2.1. Information

This section contains modules to print information about datasets. All operators print their 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

\(<\text{operator}>\) 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:

\texttt{cdo infon ifile}

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

\begin{verbatim}
<table>
<thead>
<tr>
<th></th>
<th>Date</th>
<th>Time</th>
<th>Level</th>
<th>Size</th>
<th>Miss</th>
<th>Minimum</th>
<th>Mean</th>
<th>Maximum</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1987-01-31 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>232.77</td>
<td>266.65</td>
<td>305.31</td>
<td>SST</td>
</tr>
<tr>
<td>2</td>
<td>1987-02-28 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>233.64</td>
<td>267.11</td>
<td>307.15</td>
<td>SST</td>
</tr>
<tr>
<td>3</td>
<td>1987-03-31 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>225.31</td>
<td>267.52</td>
<td>307.67</td>
<td>SST</td>
</tr>
<tr>
<td>4</td>
<td>1987-04-30 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>215.68</td>
<td>268.65</td>
<td>310.47</td>
<td>SST</td>
</tr>
<tr>
<td>5</td>
<td>1987-05-31 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>215.78</td>
<td>271.53</td>
<td>312.49</td>
<td>SST</td>
</tr>
<tr>
<td>6</td>
<td>1987-06-30 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>212.89</td>
<td>272.80</td>
<td>314.18</td>
<td>SST</td>
</tr>
<tr>
<td>7</td>
<td>1987-07-31 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>209.52</td>
<td>274.29</td>
<td>316.34</td>
<td>SST</td>
</tr>
<tr>
<td>8</td>
<td>1987-08-31 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>210.48</td>
<td>274.41</td>
<td>315.83</td>
<td>SST</td>
</tr>
<tr>
<td>9</td>
<td>1987-09-30 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>210.48</td>
<td>272.37</td>
<td>312.86</td>
<td>SST</td>
</tr>
<tr>
<td>10</td>
<td>1987-10-31 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>219.46</td>
<td>270.53</td>
<td>309.51</td>
<td>SST</td>
</tr>
<tr>
<td>11</td>
<td>1987-11-30 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>230.98</td>
<td>269.85</td>
<td>308.61</td>
<td>SST</td>
</tr>
<tr>
<td>12</td>
<td>1987-12-31 12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td></td>
<td>241.25</td>
<td>269.94</td>
<td>309.27</td>
<td>SST</td>
</tr>
</tbody>
</table>
\end{verbatim}
2.1.2. SINFO - Short information

Synopsis

<operator> ifiles

Description

This module writes information about the structure of ifiles to standard output. ifiles is an unlimited number of input files. All input files need to have the same structure with the same variables on different 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 : Institute Source TType Levels Num Gridsize Num Dtype : Name |
|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| 1 : MPI/ET ECHAM5 constant 1 1 2048 1 F32 : GEOSP |
| 2 : MPI/ET ECHAM5 instan 4 2 2048 1 F32 : T |
| 3 : MPI/ET ECHAM5 instan 1 1 2048 1 F32 : TSURF |

Grid coordinates:
1: gaussian > size : dim = 2048 nlon = 64 nlat = 32
longitude : first = 0 last = 354.375 inc = 5.625
latitude : first = 85.7605871 last = -85.7605871

Vertical coordinates:
1 : surface : 0
2 : pressure : Pa : 92500 85000 50000 20000

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
- 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

\[
\text{Absdiff}(t, x) = |i_1(t, x) - i_2(t, x)|
\]

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

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:

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Level</th>
<th>Size</th>
<th>Miss</th>
<th>S</th>
<th>Z</th>
<th>Max.Absdiff</th>
<th>Max.Reldiff</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1987−01−31</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>0.000010681</td>
<td>4.1660e−07</td>
<td>SST</td>
</tr>
<tr>
<td>2:1987−02−28</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>6.1035e−05</td>
<td>2.3742e−07</td>
<td>SST</td>
</tr>
<tr>
<td>3:1987−03−31</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>7.6294e−05</td>
<td>3.3784e−07</td>
<td>SST</td>
</tr>
<tr>
<td>4:1987−04−30</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>7.6294e−05</td>
<td>3.5117e−07</td>
<td>SST</td>
</tr>
<tr>
<td>5:1987−05−31</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>0.000010681</td>
<td>4.0307e−07</td>
<td>SST</td>
</tr>
<tr>
<td>6:1987−06−30</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>0.000010681</td>
<td>4.2510e−07</td>
<td>SST</td>
</tr>
<tr>
<td>7:1987−07−31</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>9.1553e−05</td>
<td>3.5634e−07</td>
<td>SST</td>
</tr>
<tr>
<td>8:1987−08−31</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>7.6294e−05</td>
<td>3.7624e−07</td>
<td>SST</td>
</tr>
<tr>
<td>9:1987−09−30</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>7.6294e−05</td>
<td>3.839e−07</td>
<td>SST</td>
</tr>
<tr>
<td>10:1987−10−31</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>9.1553e−05</td>
<td>3.5001e−07</td>
<td>SST</td>
</tr>
<tr>
<td>11:1987−11−30</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>6.1035e−05</td>
<td>2.3839e−07</td>
<td>SST</td>
</tr>
<tr>
<td>12:1987−12−31</td>
<td>12:00:00</td>
<td>0</td>
<td>2048</td>
<td>1361</td>
<td>F</td>
<td>F</td>
<td>9.3553e−05</td>
<td>3.7624e−07</td>
<td>SST</td>
</tr>
</tbody>
</table>
2.1.4. NINFO - Print the number of parameters, levels or times

Synopsis

\texttt{\textless operator\textgreater \ ifile}

Description

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

Operators

\begin{itemize}
  \item \texttt{npar} \hspace{1cm} Number of parameters
  \hspace{1cm} Prints the number of parameters (variables).
  \item \texttt{nlevel} \hspace{1cm} Number of levels
  \hspace{1cm} Prints the number of levels for each variable.
  \item \texttt{nyear} \hspace{1cm} Number of years
  \hspace{1cm} Prints the number of different years.
  \item \texttt{nmon} \hspace{1cm} Number of months
  \hspace{1cm} Prints the number of different combinations of years and months.
  \item \texttt{ndate} \hspace{1cm} Number of dates
  \hspace{1cm} Prints the number of different dates.
  \item \texttt{ntime} \hspace{1cm} Number of timesteps
  \hspace{1cm} Prints the number of timesteps.
\end{itemize}

Example

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

\begin{verbatim}
cdo npar ifile
\end{verbatim}

To print the number of months in a dataset use:

\begin{verbatim}
cdo nmon ifile
\end{verbatim}
2.1.5. SHOWINFO - Show variables, levels or times

Synopsis

\texttt{<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:

\texttt{cdo showcode ifile}

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

\begin{verbatim}
129 130 139
\end{verbatim}

To print all months in a dataset use:

\texttt{cdo showmon ifile}

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

\begin{verbatim}
1 2 3 4 5 6 7 8 9 10 11 12
\end{verbatim}
2.1.6. FILEDES - Dataset description

Synopsis

\[ <\text{operator}> \text{ ifile} \]

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 Gaussian N16 grid. To print the grid description of this dataset use:

```
cdo griddes ifile
```

Result:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gridtype</td>
<td>gaussian</td>
</tr>
<tr>
<td>gridsize</td>
<td>2048</td>
</tr>
<tr>
<td>xname</td>
<td>lon</td>
</tr>
<tr>
<td>xlongname</td>
<td>longitude</td>
</tr>
<tr>
<td>xunits</td>
<td>degrees_east</td>
</tr>
<tr>
<td>yname</td>
<td>lat</td>
</tr>
<tr>
<td>ylongname</td>
<td>latitude</td>
</tr>
<tr>
<td>yunits</td>
<td>degrees_north</td>
</tr>
<tr>
<td>xsize</td>
<td>64</td>
</tr>
<tr>
<td>ysize</td>
<td>32</td>
</tr>
<tr>
<td>xfirst</td>
<td>0</td>
</tr>
<tr>
<td>xinc</td>
<td>5.625</td>
</tr>
</tbody>
</table>
### 2.2. File operations

This section contains modules to perform operations on files.

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy</td>
<td>Copy datasets</td>
</tr>
<tr>
<td>cat</td>
<td>Concatenate datasets</td>
</tr>
<tr>
<td>replace</td>
<td>Replace variables</td>
</tr>
<tr>
<td>merge</td>
<td>Merge datasets with different fields</td>
</tr>
<tr>
<td>mergetime</td>
<td>Merge datasets sorted by date and time</td>
</tr>
<tr>
<td>splitcode</td>
<td>Split code numbers</td>
</tr>
<tr>
<td>splitparam</td>
<td>Split parameter identifiers</td>
</tr>
<tr>
<td>splitname</td>
<td>Split variable names</td>
</tr>
<tr>
<td>splitlevel</td>
<td>Split levels</td>
</tr>
<tr>
<td>splitgrid</td>
<td>Split grids</td>
</tr>
<tr>
<td>splitzaxis</td>
<td>Split z-axes</td>
</tr>
<tr>
<td>splittabnum</td>
<td>Split parameter table numbers</td>
</tr>
<tr>
<td>splithour</td>
<td>Split hours</td>
</tr>
<tr>
<td>splitday</td>
<td>Split days</td>
</tr>
<tr>
<td>splitmon</td>
<td>Split months</td>
</tr>
<tr>
<td>splitseas</td>
<td>Split seasons</td>
</tr>
<tr>
<td>splityear</td>
<td>Split years</td>
</tr>
<tr>
<td>splitsel</td>
<td>Split time selection</td>
</tr>
</tbody>
</table>
2.2.1. COPY - Copy datasets

Synopsis

<operator> ifiles ofile

Description

This module contains operators to copy or concatenate datasets. `ifiles` is an unlimited number of input files. All input files need to have the same structure with the same variables on different 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 of ifile1 with 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 tsml1. To replace the variable tsml1 in ifile1 with tsml1 from ifile2 use:

cdo replace ifile1 ifile2 ofile
2.2.3. MERGE - Merge datasets

Synopsis

\[ <operator> \text{ ifiles ofile} \]

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 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
```
2.2.4. SPLIT - Split a dataset

Synopsis

<operator>[,swap] ifile obase

Description

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

Operators

- **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.

- **splitslevel**  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

Environment

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

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

```
cdo splitcode ifile code
```

Result of `dir code*`:

```
code129.grb  code130.grb  code139.grb
```
2.2.5. SPLITTIME - Split timesteps of a dataset

Synopsis

<operator> 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.

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

- **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.

Environment

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

Example

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
```
2.2.6. SPLITSEL - Split selected timesteps

Synopsis

\texttt{splitsel,nsets[,noffset[,nskip]] ifile obase}

Description

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

Parameter

\begin{itemize}
  \item \texttt{nsets} \hspace{1cm} \texttt{INTEGER} \hspace{1cm} Number of input timesteps for each output file
  \item \texttt{noffset} \hspace{1cm} \texttt{INTEGER} \hspace{1cm} Number of input timesteps skipped before the first timestep range (optional)
  \item \texttt{nskip} \hspace{1cm} \texttt{INTEGER} \hspace{1cm} Number of input timesteps skipped between timestep ranges (optional)
\end{itemize}

Environment

\texttt{CDO\_FILE\_SUFFIX} \hspace{1cm} This environment variable can be used to set the default file suffix. This suffix will be added to the output file names instead of the filename extension derived from the file format. Set this variable to NULL to disable the adding of a file suffix.
2.3. Selection

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

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

- **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
- **selltype**: Select GRIB level types
- **seltabnum**: Select parameter table numbers
- **seltimestep**: Select timesteps
- **seltime**: Select times
- **selhour**: Select hours
- **selday**: Select days
- **selmon**: Select months
- **selyear**: Select years
- **selseas**: Select seasons
- **seldate**: Select dates
- **selsmon**: Select single month
- **sellonlatbox**: Select a longitude/latitude box
- **selindexbox**: Select an index box
2.3.1. SELVAR - Select fields

Synopsis

```
selparam, params ifile ofile

delparam, params ifile ofile

selcode, codes ifile ofile

delcode, codes ifile ofile

selname, names ifile ofile

delname, names ifile ofile

selstdname, stdnames ifile ofile

sellevel, levels ifile ofile

sellevidx, levidx ifile ofile

selgrid, grids ifile ofile

selzaxis, zaxes ifile ofile

selltype, ltypes ifile ofile

seltabnum, tabnums ifile ofile
```

Description

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

- **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.

- **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 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 names or numbers
- **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
```
2.3.2. SELTIME - Select timesteps

Synopsis

```
seltimestep,timesteps ifile ofile
seltime,times ifile ofile
selhour,hours ifile ofile
selday,days ifile ofile
selmon,months ifile ofile
selyear,years ifile ofile
selseas,seasons ifile ofile
seldate,date1[,date2] ifile ofile
selsmon,month[,nts1[,nts2]] ifile ofile
```

Description

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

Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seltimestep</td>
<td>Select timesteps</td>
</tr>
<tr>
<td>seltime</td>
<td>Select times</td>
</tr>
<tr>
<td>selhour</td>
<td>Select hours</td>
</tr>
<tr>
<td>selday</td>
<td>Select days</td>
</tr>
<tr>
<td>selmon</td>
<td>Select months</td>
</tr>
<tr>
<td>selyear</td>
<td>Select years</td>
</tr>
<tr>
<td>selseas</td>
<td>Select seasons</td>
</tr>
<tr>
<td>seldate</td>
<td>Select dates</td>
</tr>
<tr>
<td>selsmon</td>
<td>Select single month</td>
</tr>
</tbody>
</table>

- Seltimestep selects all timesteps with a timestep in a user given list.
- Seltime selects all timesteps with a time in a user given list.
- Selhour selects all timesteps with a hour in a user given list.
- Selday selects all timesteps with a day in a user given list.
- Selmon selects all timesteps with a month in a user given list.
- Selyear selects all timesteps with a year in a user given list.
- Selseas selects all timesteps with a month of a season in a user given list.
- Seldate selects all timesteps with a date in a user given range.
- Selsmon selects a month and optional an unlimited number of timesteps before and after this month.
Parameter

- **timesteps**: INTEGER, Comma separated list of timesteps
- **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 (DJF, MAM, JJA, SON)
- **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]
2.3.3. SELBOX - Select a box of a field

Synopsis

```
sellonlatbox,lon1,lon2,lat1,lat2  ifile ofile
selindexbox,idx1,idx2,idxy1,idxy2  ifile ofile
```

Description

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

Operators

```
sellonlatbox
Select a longitude/latitude box
Selects a longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box. Considered are only those grid cells with the grid center inside the lon/lat box.
```

```
selindexbox
Select an index box
Selects an index box. The user has to give the indexes of the edges of the box. The index of the left edge may be greater than 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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifthen</td>
<td>If then</td>
</tr>
<tr>
<td>ifnotthen</td>
<td>If not then</td>
</tr>
<tr>
<td>ifthenelse</td>
<td>If then else</td>
</tr>
<tr>
<td>ifthenc</td>
<td>If then constant</td>
</tr>
<tr>
<td>ifnotthenc</td>
<td>If not then constant</td>
</tr>
</tbody>
</table>
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

\textbf{ifthen} \quad \text{If then}

\[
o(t, x) = \begin{cases} 
  i_2(t, x) & \text{if } i_1([t, x]) \neq 0 \land i_1([t, x]) \neq \text{miss} \\
  \text{miss} & \text{if } i_1([t, x]) = 0 \lor i_1([t, x]) = \text{miss}
\end{cases}
\]

\textbf{ifnotthen} \quad \text{If not then}

\[
o(t, x) = \begin{cases} 
  i_2(t, x) & \text{if } i_1([t, x]) = 0 \land i_1([t, x]) \neq \text{miss} \\
  \text{miss} & \text{if } i_1([t, x]) \neq 0 \lor i_1([t, x]) = \text{miss}
\end{cases}
\]

Example

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

\texttt{cdo ifthen ifile1 ifile2 ofile}

2.4.2. COND2 - Conditional select two fields

Synopsis

\texttt{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.

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

- `ifthenenc`: If then constant
  
  \[
  o(t,x) = \begin{cases} 
  c & \text{if } i(t,x) \neq 0 \land i(t,x) \neq \text{miss} \\
  \text{miss} & \text{if } i(t,x) = 0 \lor i(t,x) = \text{miss}
  \end{cases}
  \]

- `ifnotthenc`: If not then constant
  
  \[
  o(t,x) = \begin{cases} 
  c & \text{if } i(t,x) = 0 \land i(t,x) \neq \text{miss} \\
  \text{miss} & \text{if } i(t,x) \neq 0 \lor i(t,x) = \text{miss}
  \end{cases}
  \]

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 ifthenenc,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

The operators are:

- **eq** (Equal)
  
  \[
  o(t,x) = \begin{cases} 
  1 & \text{if } i_1(t,x) = i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  0 & \text{if } i_1(t,x) \neq i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  \text{miss} & \text{if } i_1(t,x) \text{ or } i_2(t,x) \text{ is miss} 
  \end{cases}
  \]

- **ne** (Not equal)
  
  \[
  o(t,x) = \begin{cases} 
  1 & \text{if } i_1(t,x) \neq i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  0 & \text{if } i_1(t,x) = i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  \text{miss} & \text{if } i_1(t,x) \text{ or } i_2(t,x) \text{ is miss} 
  \end{cases}
  \]

- **le** (Less equal)
  
  \[
  o(t,x) = \begin{cases} 
  1 & \text{if } i_1(t,x) \leq i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  0 & \text{if } i_1(t,x) > i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  \text{miss} & \text{if } i_1(t,x) \text{ or } i_2(t,x) \text{ is miss} 
  \end{cases}
  \]

- **lt** (Less than)
  
  \[
  o(t,x) = \begin{cases} 
  1 & \text{if } i_1(t,x) < i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  0 & \text{if } i_1(t,x) \geq i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  \text{miss} & \text{if } i_1(t,x) \text{ or } i_2(t,x) \text{ is miss} 
  \end{cases}
  \]

- **ge** (Greater equal)
  
  \[
  o(t,x) = \begin{cases} 
  1 & \text{if } i_1(t,x) \geq i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  0 & \text{if } i_1(t,x) < i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  \text{miss} & \text{if } i_1(t,x) \text{ or } i_2(t,x) \text{ is miss} 
  \end{cases}
  \]

- **gt** (Greater than)
  
  \[
  o(t,x) = \begin{cases} 
  1 & \text{if } i_1(t,x) > i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  0 & \text{if } i_1(t,x) \leq i_2(t,x) \land i_1(t,x), i_2(t,x) \neq \text{miss} \\
  \text{miss} & \text{if } i_1(t,x) \text{ or } i_2(t,x) \text{ is miss} 
  \end{cases}
  \]

Example

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

\[
\text{cdo eq ifile1 ifile2 ofile}
\]
2.5.2. COMPC - Comparison of a field with a constant

Synopsis

\(<\text{operator}>,c\ \text{ifile ofile}\)

Description

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

Operators

- **eqc** Equal constant
  
  \[ o(t, x) = \begin{cases} 
  1 & \text{if } i(t, x) = c \land i(t, x) \neq \text{miss} \\
  0 & \text{if } i(t, x) \neq c \land i(t, x) \neq \text{miss} \\
  \text{miss} & \text{if } i(t, x) = \text{miss} \lor c = \text{miss} 
  \end{cases} \]

- **nec** Not equal constant
  
  \[ o(t, x) = \begin{cases} 
  1 & \text{if } i(t, x) \neq c \land i(t, x) \neq \text{miss} \\
  0 & \text{if } i(t, x) = c \land i(t, x) \neq \text{miss} \\
  \text{miss} & \text{if } i(t, x) = \text{miss} \lor c = \text{miss} 
  \end{cases} \]

- **lec** Less equal constant
  
  \[ o(t, x) = \begin{cases} 
  1 & \text{if } i(t, x) \leq c \land i(t, x) \neq \text{miss} \\
  0 & \text{if } i(t, x) > c \land i(t, x) \neq \text{miss} \\
  \text{miss} & \text{if } i(t, x) = \text{miss} \lor c = \text{miss} 
  \end{cases} \]

- **ltc** Less than constant
  
  \[ o(t, x) = \begin{cases} 
  1 & \text{if } i(t, x) < c \land i(t, x) \neq \text{miss} \\
  0 & \text{if } i(t, x) \geq c \land i(t, x) \neq \text{miss} \\
  \text{miss} & \text{if } i(t, x) = \text{miss} \lor c = \text{miss} 
  \end{cases} \]

- **gec** Greater equal constant
  
  \[ o(t, x) = \begin{cases} 
  1 & \text{if } i(t, x) \geq c \land i(t, x) \neq \text{miss} \\
  0 & \text{if } i(t, x) < c \land i(t, x) \neq \text{miss} \\
  \text{miss} & \text{if } i(t, x) = \text{miss} \lor c = \text{miss} 
  \end{cases} \]

- **gtc** Greater than constant
  
  \[ o(t, x) = \begin{cases} 
  1 & \text{if } i(t, x) > c \land i(t, x) \neq \text{miss} \\
  0 & \text{if } i(t, x) \leq c \land i(t, x) \neq \text{miss} \\
  \text{miss} & \text{if } i(t, x) = \text{miss} \lor c = \text{miss} 
  \end{cases} \]

Parameter

- **c** FLOAT Constant

Example

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

\[ \text{cdo gtc,273.15 ifile ofile} \]
## 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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setpartab</code></td>
<td>Set parameter table</td>
</tr>
<tr>
<td><code>setcode</code></td>
<td>Set code number</td>
</tr>
<tr>
<td><code>setparam</code></td>
<td>Set parameter identifier</td>
</tr>
<tr>
<td><code>setname</code></td>
<td>Set variable name</td>
</tr>
<tr>
<td><code>setunit</code></td>
<td>Set variable unit</td>
</tr>
<tr>
<td><code>setlevel</code></td>
<td>Set level</td>
</tr>
<tr>
<td><code>setltype</code></td>
<td>Set GRIB level type</td>
</tr>
<tr>
<td><code>setdate</code></td>
<td>Set date</td>
</tr>
<tr>
<td><code>settime</code></td>
<td>Set time of the day</td>
</tr>
<tr>
<td><code>setday</code></td>
<td>Set day</td>
</tr>
<tr>
<td><code>setmon</code></td>
<td>Set month</td>
</tr>
<tr>
<td><code>setyear</code></td>
<td>Set year</td>
</tr>
<tr>
<td><code>settunits</code></td>
<td>Set time units</td>
</tr>
<tr>
<td><code>settaxis</code></td>
<td>Set time axis</td>
</tr>
<tr>
<td><code>setreftime</code></td>
<td>Set reference time</td>
</tr>
<tr>
<td><code>setcalendar</code></td>
<td>Set calendar</td>
</tr>
<tr>
<td><code>shifttime</code></td>
<td>Shift timesteps</td>
</tr>
<tr>
<td><code>chcode</code></td>
<td>Change code number</td>
</tr>
<tr>
<td><code>chparam</code></td>
<td>Change parameter identifier</td>
</tr>
<tr>
<td><code>chname</code></td>
<td>Change variable name</td>
</tr>
<tr>
<td><code>chunit</code></td>
<td>Change variable unit</td>
</tr>
<tr>
<td><code>chlevel</code></td>
<td>Change level</td>
</tr>
<tr>
<td><code>chlevelc</code></td>
<td>Change level of one code</td>
</tr>
<tr>
<td><code>chlevelv</code></td>
<td>Change level of one variable</td>
</tr>
<tr>
<td><code>setgrid</code></td>
<td>Set grid</td>
</tr>
<tr>
<td><code>setgridtype</code></td>
<td>Set grid type</td>
</tr>
<tr>
<td><code>setgridarea</code></td>
<td>Set grid cell area</td>
</tr>
<tr>
<td><code>setzaxis</code></td>
<td>Set z-axis</td>
</tr>
<tr>
<td><code>setgatt</code></td>
<td>Set global attribute</td>
</tr>
<tr>
<td><code>setgatts</code></td>
<td>Set global attributes</td>
</tr>
<tr>
<td><code>invertlat</code></td>
<td>Invert latitudes</td>
</tr>
<tr>
<td><code>invertlev</code></td>
<td>Invert levels</td>
</tr>
<tr>
<td><code>maskregion</code></td>
<td>Mask regions</td>
</tr>
<tr>
<td><code>masklonlatbox</code></td>
<td>Mask a longitude/latitude box</td>
</tr>
<tr>
<td><code>maskindexbox</code></td>
<td>Mask an index box</td>
</tr>
<tr>
<td><code>setclonlatbox</code></td>
<td>Set a longitude/latitude box to constant</td>
</tr>
<tr>
<td><code>setcindexbox</code></td>
<td>Set an index box to constant</td>
</tr>
<tr>
<td><code>enlarge</code></td>
<td>Enlarge fields</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>setmissval</td>
<td>Set a new missing value</td>
</tr>
<tr>
<td>setctomiss</td>
<td>Set constant to missing value</td>
</tr>
<tr>
<td>setmisstoc</td>
<td>Set missing value to constant</td>
</tr>
<tr>
<td>setrtomiss</td>
<td>Set range to missing value</td>
</tr>
<tr>
<td>setvrange</td>
<td>Set valid range</td>
</tr>
</tbody>
</table>
2.6.1. SET - Set field info

Synopsis

```plaintext
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 (format: code[.tabnum] or num[.cat[.dis]])
- **name** STRING Variable name
- **level** FLOAT New level
- **ltype** INTEGER GRIB level type

Example

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

```plaintext
cdo setpartab,echam5 ifile ofile
```
2.6.2. SETTIME - Set time

Synopsis

\texttt{setdate, date ifile ofile}
\texttt{settime, time ifile ofile}
\texttt{setday, day ifile ofile}
\texttt{setmon, month ifile ofile}
\texttt{setyear, year ifile ofile}
\texttt{setunits, units ifile ofile}
\texttt{settaxis, date,time[,inc] ifile ofile}
\texttt{setreftime, date,time[,units] ifile ofile}
\texttt{setcalendar, calendar ifile ofile}
\texttt{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

\texttt{setdate} \hspace{1cm} Set date
Sets the date in every timestep to the same given value.

\texttt{settime} \hspace{1cm} Set time of the day
Sets the time in every timestep to the same given value.

\texttt{setday} \hspace{1cm} Set day
Sets the day in every timestep to the same given value.

\texttt{setmon} \hspace{1cm} Set month
Sets the month in every timestep to the same given value.

\texttt{setyear} \hspace{1cm} Set year
Sets the year in every timestep to the same given value.

\texttt{setunits} \hspace{1cm} Set time units
Sets the base units of a relative time axis.

\texttt{settaxis} \hspace{1cm} Set time axis
Sets the time axis.

\texttt{setreftime} \hspace{1cm} Set reference time
Sets the reference time of a relative time axis.

\texttt{setcalendar} \hspace{1cm} Set calendar
Sets the calendar of a relative time axis.

\texttt{shifttime} \hspace{1cm} Shift timesteps
Shifts all timesteps by the parameter sval.
Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>INTEGER</td>
<td>Value of the new day</td>
</tr>
<tr>
<td>month</td>
<td>INTEGER</td>
<td>Value of the new month</td>
</tr>
<tr>
<td>year</td>
<td>INTEGER</td>
<td>Value of the new year</td>
</tr>
<tr>
<td>units</td>
<td>STRING</td>
<td>Base units of the time axis (seconds, minutes, hours, days, months, years)</td>
</tr>
<tr>
<td>date</td>
<td>STRING</td>
<td>Date (format: YYYY-MM-DD)</td>
</tr>
<tr>
<td>time</td>
<td>STRING</td>
<td>Time (format: hh:mm:ss)</td>
</tr>
<tr>
<td>inc</td>
<td>STRING</td>
<td>Optional increment (seconds, minutes, hours, days, months, years) [default:</td>
</tr>
<tr>
<td>calendar</td>
<td>STRING</td>
<td>Calendar (standard, proleptic, 360days, 365days, 366days)</td>
</tr>
<tr>
<td>sval</td>
<td>STRING</td>
<td>Shift value (e.g. -3hour)</td>
</tr>
</tbody>
</table>

Example

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

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

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

```
```

To shift this time axis by -15 days use:

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

Result of 'cdo showdate ofile':

```
```
2.6.3. CHANGE - Change field header

Synopsis

\texttt{chcode,oldcode,newcode[,...] ifileofile}
\texttt{chparam,oldparam,newparam,... ifileofile}
\texttt{chname,oldname,newname,... ifileofile}
\texttt{chunit,oldunit,newunit,... ifileofile}
\texttt{chlevel,oldlev,newlev,... ifileofile}
\texttt{chlevelc,code,oldlev,newlev ifileofile}
\texttt{chlevelv,name,oldlev,newlev ifileofile}

Description

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

Operators

- \texttt{chcode} Change code number
  Changes some user given code numbers to new user given values.
- \texttt{chparam} Change parameter identifier
  Changes some user given parameter identifiers to new user given values.
- \texttt{chname} Change variable name
  Changes some user given variable names to new user given names.
- \texttt{chunit} Change variable unit
  Changes some user given variable units to new user given units.
- \texttt{chlevel} Change level
  Changes some user given levels to new user given values.
- \texttt{chlevelc} Change level of one code
  Changes one level of a user given code number.
- \texttt{chlevelv} Change level of one variable
  Changes one level of a user given variable name.

Parameter

\begin{itemize}
  \item \texttt{code} INTEGER Code number
  \item \texttt{oldcode,newcode,...} INTEGER Pairs of old and new code numbers
  \item \texttt{oldparam,newparam,...} STRING Pairs of old and new parameter identifiers
  \item \texttt{name} STRING Variable name
  \item \texttt{oldname,newname,...} STRING Pairs of old and new variable names
  \item \texttt{oldlev} FLOAT Old level
  \item \texttt{newlev} FLOAT New level
  \item \texttt{oldlev,newlev,...} FLOAT Pairs of old and new levels
\end{itemize}
Example

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

```
cdo chcode,98,179,99,211 ifile ofile
```
2.6.4. SETGRID - Set grid information

Synopsis

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

Description

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

Operators

- **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 regular grid to curvilinear grid
    - `unstructured`: Converts grid type to unstructured grid
    - `dereference`: Dereference grid type REFERENCE
    - `regular`: Converts reduced Gaussian grid to regular Gaussian grid

- **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 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
```
2.6.5. SETZAXIS - Set z-axis type

Synopsis

```
setzaxis, zaxis ifile ofile
```

Description

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

Parameter

```
zaxis  STRING  Z-axis description file or name of the target z-axis
```
2.6.6. SETGATT - Set global attribute

Synopsis

setgatt,attname,attstring ifile ofile
setgatts,attfile ifile ofile

Description

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

Operators

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" ;
}
```
2.6.7. INVERT - Invert latitudes

Synopsis

invertlat ifile ofile

Description

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

Example

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

```cdo invertlat ifile ofile```

2.6.8. INVERTLEV - Invert levels

Synopsis

invertlev ifile ofile

Description

This operator inverts the levels of all non hybrid 3D variables.
2.6.9. MASKREGION - Mask regions

Synopsis

\texttt{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

\texttt{regions} \hspace{1cm} \texttt{STRING} \hspace{1cm} 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:

\begin{verbatim}
cdo maskregion,myregion ifile ofile
\end{verbatim}

For this example the polygon description file \texttt{myregion} should contain the following four coordinates:

\begin{verbatim}
120 20
120 -20
270 -20
270 20
\end{verbatim}
2.6.10. MASKBOX - Mask a box

Synopsis

\texttt{masklonlatbox,lon1,lon2,lat1,lat2 \textit{ifile ofile}}
\texttt{maskindexbox,idx1,idx2,idy1,idy2 \textit{ifile ofile}}

Description

Masks a box of the rectangular understood field. The elements inside the box are untouched, the elements outside are set to missing value. All input fields need to have the same horizontal grid. Use \texttt{selonlatbox} or \texttt{selindexbox} if only the data inside the box are needed.

Operators

- \texttt{masklonlatbox} Mask a longitude/latitude box
  Masks a longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box. Considered are only those grid cells with the grid center inside the lon/lat box.

- \texttt{maskindexbox} Mask an index box
  Masks an index box. The user has to give the indexes of the edges of the box. The index of the left edge can be greater then the one of the right edge.

Parameter

- \textit{lon1} FLOAT Western longitude
- \textit{lon2} FLOAT Eastern longitude
- \textit{lat1} FLOAT Southern or northern latitude
- \textit{lat2} FLOAT Northern or southern latitude
- \textit{idx1} INTEGER Index of first longitude
- \textit{idx2} INTEGER Index of last longitude
- \textit{idy1} INTEGER Index of first latitude
- \textit{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:

\texttt{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 \texttt{maskindexbox} by:

\texttt{cdo maskindexbox,23,48,13,20 ifile ofile}
2.6.11. SETBOX - Set a box to constant

Synopsis

```
setclonlatbox,c,lon1,lon2,lat1,lat2  ifile ofile
setcindexbox,c,idx1,idx2,idy1,idy2  ifile ofile
```

Description

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

Operators

- **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
```
2.6.12. ENLARGE - Enlarge fields

Synopsis

\texttt{enlarge,grid ifile ofile}

Description

Enlarge all fields of \texttt{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

\textit{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:

\begin{verbatim}
  cdo enlarge,ifile1 ifile2 tmpfile
  cdo add ifile1 tmpfile ofile
\end{verbatim}

Or shorter using operator piping:

\begin{verbatim}
  cdo add ifile1 -enlarge,ifile1 ifile2 ofile
\end{verbatim}
### 2.6.13. SETMISS - Set missing value

#### Synopsis

```plaintext
setmissval newmiss ifile ofile
setctomiss c ifile ofile
setmisstoc c ifile ofile
setrtomiss rmin rmax ifile ofile
setvrange rmin rmax ifile ofile
```

#### Description

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

#### Operators

**setmissval**  
Set a new missing value  
\[
o(t, x) = \begin{cases} 
  \text{newmiss} & \text{if } i(t, x) = \text{miss} \\
  i(t, x) & \text{if } i(t, x) \neq \text{miss} 
\end{cases}
\]

**setctomiss**  
Set constant to missing value  
\[
o(t, x) = \begin{cases} 
  \text{miss} & \text{if } i(t, x) = c \\
  i(t, x) & \text{if } i(t, x) \neq c 
\end{cases}
\]

**setmisstoc**  
Set missing value to constant  
\[
o(t, x) = \begin{cases} 
  c & \text{if } i(t, x) = \text{miss} \\
  i(t, x) & \text{if } i(t, x) \neq \text{miss} 
\end{cases}
\]

**setrtomiss**  
Set range to missing value  
\[
o(t, x) = \begin{cases} 
  \text{miss} & \text{if } i(t, x) \geq rmin \wedge i(t, x) \leq rmax \\
  i(t, x) & \text{if } i(t, x) < rmin \lor i(t, x) > rmax 
\end{cases}
\]

**setvrange**  
Set valid range  
\[
o(t, x) = \begin{cases} 
  \text{miss} & \text{if } i(t, x) < rmin \lor i(t, x) > rmax \\
  i(t, x) & \text{if } i(t, x) \geq rmin \wedge i(t, x) \leq rmax 
\end{cases}
\]

#### Parameter

- **newmiss**  
  FLOAT  
  New missing value
- **c**  
  FLOAT  
  Constant
- **rmin**  
  FLOAT  
  Lower bound
- **rmax**  
  FLOAT  
  Upper bound
Example

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

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

Result of `cdo info ifile`:

<table>
<thead>
<tr>
<th>Date Time Code Level</th>
<th>Size</th>
<th>Miss</th>
<th>Minimum</th>
<th>Mean</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987−12−31 12:00:00 139</td>
<td>0</td>
<td>2048</td>
<td>0</td>
<td>246.27</td>
<td>276.75</td>
</tr>
</tbody>
</table>

Result of `cdo info ofile`:

<table>
<thead>
<tr>
<th>Date Time Code Level</th>
<th>Size</th>
<th>Miss</th>
<th>Minimum</th>
<th>Mean</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987−12−31 12:00:00 139</td>
<td>0</td>
<td>2048</td>
<td>871</td>
<td>273.16</td>
<td>287.08</td>
</tr>
</tbody>
</table>
2.7. Arithmetic

This section contains modules to arithmetically process datasets.

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expr</td>
<td>Evaluate expressions</td>
</tr>
<tr>
<td>exprf</td>
<td>Evaluate expressions from script file</td>
</tr>
<tr>
<td>abs</td>
<td>Absolute value</td>
</tr>
<tr>
<td>int</td>
<td>Integer value</td>
</tr>
<tr>
<td>nint</td>
<td>Nearest integer value</td>
</tr>
<tr>
<td>pow</td>
<td>Power</td>
</tr>
<tr>
<td>sqr</td>
<td>Square</td>
</tr>
<tr>
<td>sqrt</td>
<td>Square root</td>
</tr>
<tr>
<td>exp</td>
<td>Exponential</td>
</tr>
<tr>
<td>ln</td>
<td>Natural logarithm</td>
</tr>
<tr>
<td>log10</td>
<td>Base 10 logarithm</td>
</tr>
<tr>
<td>sin</td>
<td>Sine</td>
</tr>
<tr>
<td>cos</td>
<td>Cosine</td>
</tr>
<tr>
<td>tan</td>
<td>Tangent</td>
</tr>
<tr>
<td>asin</td>
<td>Arc sine</td>
</tr>
<tr>
<td>acos</td>
<td>Arc cosine</td>
</tr>
<tr>
<td>reci</td>
<td>Reciprocal value</td>
</tr>
<tr>
<td>addc</td>
<td>Add a constant</td>
</tr>
<tr>
<td>subc</td>
<td>Subtract a constant</td>
</tr>
<tr>
<td>mulc</td>
<td>Multiply with a constant</td>
</tr>
<tr>
<td>divc</td>
<td>Divide by a constant</td>
</tr>
<tr>
<td>add</td>
<td>Add two fields</td>
</tr>
<tr>
<td>sub</td>
<td>Subtract two fields</td>
</tr>
<tr>
<td>mul</td>
<td>Multiply two fields</td>
</tr>
<tr>
<td>div</td>
<td>Divide two fields</td>
</tr>
<tr>
<td>min</td>
<td>Minimum of two fields</td>
</tr>
<tr>
<td>max</td>
<td>Maximum of two fields</td>
</tr>
<tr>
<td>atan2</td>
<td>Arc tangent of two fields</td>
</tr>
<tr>
<td>monadd</td>
<td>Add monthly time series</td>
</tr>
<tr>
<td>monsub</td>
<td>Subtract monthly time series</td>
</tr>
<tr>
<td>monmul</td>
<td>Multiply monthly time series</td>
</tr>
<tr>
<td>mondiv</td>
<td>Divide monthly time series</td>
</tr>
<tr>
<td>ymonadd</td>
<td>Add multi-year monthly time series</td>
</tr>
<tr>
<td>ymonsub</td>
<td>Subtract multi-year monthly time series</td>
</tr>
<tr>
<td>ymonmul</td>
<td>Multiply multi-year monthly time series</td>
</tr>
<tr>
<td>ymondiv</td>
<td>Divide multi-year monthly time series</td>
</tr>
<tr>
<td>ydayadd</td>
<td>Add multi-year daily time series</td>
</tr>
<tr>
<td>ydaysub</td>
<td>Subtract multi-year daily time series</td>
</tr>
<tr>
<td>ydaymul</td>
<td>Multiply multi-year daily time series</td>
</tr>
<tr>
<td>ydaydiv</td>
<td>Divide multi-year daily time series</td>
</tr>
<tr>
<td>yhouradd</td>
<td>Add multi-year hourly time series</td>
</tr>
<tr>
<td>yhoursub</td>
<td>Subtract multi-year hourly time series</td>
</tr>
<tr>
<td>yhourmul</td>
<td>Multiply multi-year hourly time series</td>
</tr>
<tr>
<td>yhourdiv</td>
<td>Divide multi-year hourly time series</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td><code>muldpm</code></td>
<td>Multiply with days per month</td>
</tr>
<tr>
<td><code>divdpm</code></td>
<td>Divide by days per month</td>
</tr>
<tr>
<td><code>muldpy</code></td>
<td>Multiply with days per year</td>
</tr>
<tr>
<td><code>divdpy</code></td>
<td>Divide by days per year</td>
</tr>
</tbody>
</table>
2.7.1. EXPR - Evaluate expressions

Synopsis

expr, instr ifile ofile

exprf, filename ifile ofile

Description

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

abs(x) Absolute value of x
int(x) Integer value of x
nint(x) Nearest integer value of x
sqr(x) Square of x
sqrt(x) Square Root of x
exp(x) Exponential of x
log(x) Natural logarithm of x
log10(x) Base 10 logarithm of x
sin(x) Sine of x, where x is specified in radians
cos(x) Cosine of x, where x is specified in radians
tan(x) Tangent of x, where x is specified in radians
asin(x) Arc-sine of x, where x is specified in radians
acos(x) Arc-cosine of x, where x is specified in radians
atan(x) Arc-tangent of x, where x is specified in radians

Operators

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

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

Parameter

instr STRING Processing instructions (without spaces!)
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

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

Example

To calculate the square root for all field elements use:

```bash
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
  \[ o(t,x) = i(t,x) \times c \]
- **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:

```bash
cdo addc,-273.15 ifile ofile
```
2.7.4. ARITH - Arithmetic on two datasets

Synopsis

\langle operator \rangle \text{ ifile1 ifile2 ofile}

Description

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

Operators

\begin{itemize}
  \item \textbf{add} \quad \text{Add two fields} \quad o(t, x) = i_1(t, x) + i_2(t, x)
  \item \textbf{sub} \quad \text{Subtract two fields} \quad o(t, x) = i_1(t, x) - i_2(t, x)
  \item \textbf{mul} \quad \text{Multiply two fields} \quad o(t, x) = i_1(t, x) \times i_2(t, x)
  \item \textbf{div} \quad \text{Divide two fields} \quad o(t, x) = \frac{i_1(t, x)}{i_2(t, x)}
  \item \textbf{min} \quad \text{Minimum of two fields} \quad o(t, x) = \min(i_1(t, x), i_2(t, x))
  \item \textbf{max} \quad \text{Maximum of two fields} \quad o(t, x) = \max(i_1(t, x), i_2(t, x))
  \item \textbf{atan2} \quad \text{Arc tangent of two fields}
  \quad \text{The atan2 operator calculates the arc tangent of two fields. The result is in radians, which is between -PI and PI (inclusive).} \quad o(t, x) = \text{atan2}(i_1(t, x), i_2(t, x))
\end{itemize}

Example

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

\texttt{cdo add ifile1 ifile2 ofile}
2.7.5. MONARITH - Monthly arithmetic

Synopsis

\[ <\text{operator}> \text{ifile1 ifile2 ofile} \]

Description

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

Operators

- \text{monadd}: Add monthly time series
  - Adds a time series and a monthly time series.
- \text{monsub}: Subtract monthly time series
  - Subtracts a time series and a monthly time series.
- \text{monmul}: Multiply monthly time series
  - Multiplies a time series and a monthly time series.
- \text{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:

\[
\text{cdo monsub ifile -monavg ifile ofile}
\]
2.7.6. YMONARITH - Multi-year monthly arithmetic

Synopsis

<operator> ifile1 ifile2 ofile

Description

This module performs simple arithmetic of a time series and one 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
2.7.7. **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 average from a time series use:

```cdo ydaysub ifile -ydayavg ifile ofile```
2.7.8. YHOURARITH - Multi-year hourly arithmetic

Synopsis

\[ \text{<operator> ifile1 ifile2 ofile} \]

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 \text{ifile1} the corresponding field of the timestep in \text{ifile2} with the same hour and day of year is used. The header information in \text{ifile1} have to be the same as in \text{ifile2}. Usually \text{ifile2} is generated by an operator of the module \text{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
```
2.7.9. 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

- **muldpdm**: Multiply with days per month
  
  \[ o(t,x) = i(t,x) \times \text{days\_per\_month} \]

- **divdpdm**: Divide by days per month
  
  \[ o(t,x) = \frac{i(t,x)}{\text{days\_per\_month}} \]

- **muldpdy**: Multiply with days per year
  
  \[ o(t,x) = i(t,x) \times \text{days\_per\_year} \]

- **divdpdy**: Divide by days per year
  
  \[ o(t,x) = \frac{i(t,x)}{\text{days\_per\_year}} \]

Example

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

```
cdo muldpdm ifile tmpfile1
cdo yearsum tmpfile1 tmpfile2
cdo divdpdy tmpfile2 ofile
```

Or all in one command line:

```
cdo divdpdy -yearsum -muldpdm ifile ofile
```
2.8. Statistical values

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

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

<table>
<thead>
<tr>
<th>sum</th>
<th>(\sum_{i=1}^{n} x_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean resp. avg</td>
<td>(n^{-1} \sum_{i=1}^{n} x_i)</td>
</tr>
<tr>
<td>mean resp. avg weighted by ({w_i, i = 1, \ldots, n})</td>
<td>((\sum_{j=1}^{n} w_j)^{-1} \sum_{i=1}^{n} w_i x_i)</td>
</tr>
<tr>
<td>Variance</td>
<td>(n^{-1} \sum_{i=1}^{n} (x_i - \bar{x})^2)</td>
</tr>
<tr>
<td>var weighted by ({w_i, i = 1, \ldots, n})</td>
<td>((\sum_{j=1}^{n} w_j)^{-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)</td>
</tr>
<tr>
<td>Standard deviation std</td>
<td>(\sqrt{n^{-1} \sum_{i=1}^{n} (x_i - \bar{x})^2})</td>
</tr>
<tr>
<td>std weighted by ({w_i, i = 1, \ldots, n})</td>
<td>(\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)</td>
</tr>
<tr>
<td>Cumulative Ranked Probability Score crps</td>
<td>(\int_{-\infty}^{\infty}</td>
</tr>
</tbody>
</table>

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

and \(H(x)\) the Heaviside function jumping at \(x\).

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

| consecsum | Consecutive Sum |
| conseqts  | Consecutive Timesteps |
| **ensmin** | Ensemble minimum |
| **ensmax** | Ensemble maximum |
| **enssum** | Ensemble sum |
| **ensmean** | Ensemble mean |
| **ensavg** | Ensemble average |
| **ensvar** | Ensemble variance |
| **ensstd** | Ensemble standard deviation |
| **enspctl** | Ensemble percentiles |

| **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 |
| **fldvar** | Field variance |
| **fldstd** | Field standard deviation |
| **fldpctl** | Field percentiles |

| **zonmin** | Zonal minimum |
| **zonmax** | Zonal maximum |
| **zonsum** | Zonal sum |
| **zonmean** | Zonal mean |
| **zonavg** | Zonal average |
| **zonestd** | Zonal standard deviation |
| **zonpctl** | Zonal percentiles |

| **mermin** | Meridional minimum |
| **mermax** | Meridional maximum |
| **mersum** | Meridional sum |
| **mermean** | Meridional mean |
| **meravg** | Meridional average |
| **mervar** | Meridional variance |
| **merstd** | Meridional standard deviation |
| **merpctl** | Meridional percentiles |

<p>| <strong>gridboxmin</strong> | Gridbox minimum |
| <strong>gridboxmax</strong> | Gridbox maximum |
| <strong>gridboxsum</strong> | Gridbox sum |
| <strong>gridboxmean</strong> | Gridbox mean |
| <strong>gridboxavg</strong> | Gridbox average |
| <strong>gridboxvar</strong> | Gridbox variance |
| <strong>gridboxstd</strong> | Gridbox standard deviation |</p>
<table>
<thead>
<tr>
<th>Statistical values</th>
<th>Reference manual</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vertmin</strong></td>
<td>Vertical minimum</td>
</tr>
<tr>
<td><strong>vertmax</strong></td>
<td>Vertical maximum</td>
</tr>
<tr>
<td><strong>vertsum</strong></td>
<td>Vertical sum</td>
</tr>
<tr>
<td><strong>vertmean</strong></td>
<td>Vertical mean</td>
</tr>
<tr>
<td><strong>vertavg</strong></td>
<td>Vertical average</td>
</tr>
<tr>
<td><strong>vertvar</strong></td>
<td>Vertical variance</td>
</tr>
<tr>
<td><strong>vertstd</strong></td>
<td>Vertical standard deviation</td>
</tr>
<tr>
<td><strong>timselmin</strong></td>
<td>Time range minimum</td>
</tr>
<tr>
<td><strong>timselmax</strong></td>
<td>Time range maximum</td>
</tr>
<tr>
<td><strong>timselsum</strong></td>
<td>Time range sum</td>
</tr>
<tr>
<td><strong>timselmean</strong></td>
<td>Time range mean</td>
</tr>
<tr>
<td><strong>timselavg</strong></td>
<td>Time range average</td>
</tr>
<tr>
<td><strong>timselvar</strong></td>
<td>Time range variance</td>
</tr>
<tr>
<td><strong>timselstd</strong></td>
<td>Time range standard deviation</td>
</tr>
<tr>
<td><strong>timselpctl</strong></td>
<td>Time range percentiles</td>
</tr>
<tr>
<td><strong>runmin</strong></td>
<td>Running minimum</td>
</tr>
<tr>
<td><strong>runmax</strong></td>
<td>Running maximum</td>
</tr>
<tr>
<td><strong>runsum</strong></td>
<td>Running sum</td>
</tr>
<tr>
<td><strong>runmean</strong></td>
<td>Running mean</td>
</tr>
<tr>
<td><strong>runavg</strong></td>
<td>Running average</td>
</tr>
<tr>
<td><strong>runvar</strong></td>
<td>Running variance</td>
</tr>
<tr>
<td><strong>runstd</strong></td>
<td>Running standard deviation</td>
</tr>
<tr>
<td><strong>runpctl</strong></td>
<td>Running percentiles</td>
</tr>
<tr>
<td><strong>timmin</strong></td>
<td>Time minimum</td>
</tr>
<tr>
<td><strong>timmax</strong></td>
<td>Time maximum</td>
</tr>
<tr>
<td><strong>tisum</strong></td>
<td>Time sum</td>
</tr>
<tr>
<td><strong>timmean</strong></td>
<td>Time mean</td>
</tr>
<tr>
<td><strong>timavg</strong></td>
<td>Time average</td>
</tr>
<tr>
<td><strong>timvar</strong></td>
<td>Time variance</td>
</tr>
<tr>
<td><strong>timstd</strong></td>
<td>Time standard deviation</td>
</tr>
<tr>
<td><strong>timpctl</strong></td>
<td>Time percentiles</td>
</tr>
<tr>
<td><strong>hourmin</strong></td>
<td>Hourly minimum</td>
</tr>
<tr>
<td><strong>hourmax</strong></td>
<td>Hourly maximum</td>
</tr>
<tr>
<td><strong>hoursun</strong></td>
<td>Hourly sum</td>
</tr>
<tr>
<td><strong>hourmean</strong></td>
<td>Hourly mean</td>
</tr>
<tr>
<td><strong>houravg</strong></td>
<td>Hourly average</td>
</tr>
<tr>
<td><strong>hourvar</strong></td>
<td>Hourly variance</td>
</tr>
<tr>
<td><strong>hourstd</strong></td>
<td>Hourly standard deviation</td>
</tr>
<tr>
<td><strong>hourpctl</strong></td>
<td>Hourly percentiles</td>
</tr>
<tr>
<td><strong>daymin</strong></td>
<td>Daily minimum</td>
</tr>
<tr>
<td><strong>daymax</strong></td>
<td>Daily maximum</td>
</tr>
<tr>
<td><strong>daysum</strong></td>
<td>Daily sum</td>
</tr>
<tr>
<td><strong>daymean</strong></td>
<td>Daily mean</td>
</tr>
<tr>
<td><strong>dayavg</strong></td>
<td>Daily average</td>
</tr>
<tr>
<td><strong>dayvar</strong></td>
<td>Daily variance</td>
</tr>
<tr>
<td><strong>daystd</strong></td>
<td>Daily standard deviation</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>daypctl</td>
<td>Daily percentiles</td>
</tr>
<tr>
<td>monmin</td>
<td>Monthly minimum</td>
</tr>
<tr>
<td>monmax</td>
<td>Monthly maximum</td>
</tr>
<tr>
<td>monsum</td>
<td>Monthly sum</td>
</tr>
<tr>
<td>monmean</td>
<td>Monthly mean</td>
</tr>
<tr>
<td>monavg</td>
<td>Monthly average</td>
</tr>
<tr>
<td>monvar</td>
<td>Monthly variance</td>
</tr>
<tr>
<td>monstd</td>
<td>Monthly standard deviation</td>
</tr>
<tr>
<td>yearpctl</td>
<td>Yearly percentiles</td>
</tr>
<tr>
<td>yearmin</td>
<td>Yearly minimum</td>
</tr>
<tr>
<td>yearmax</td>
<td>Yearly maximum</td>
</tr>
<tr>
<td>yearsum</td>
<td>Yearly sum</td>
</tr>
<tr>
<td>yearmean</td>
<td>Yearly mean</td>
</tr>
<tr>
<td>yearavg</td>
<td>Yearly average</td>
</tr>
<tr>
<td>yearvar</td>
<td>Yearly variance</td>
</tr>
<tr>
<td>yearstd</td>
<td>Yearly standard deviation</td>
</tr>
<tr>
<td>seasctl</td>
<td>Seasonal percentiles</td>
</tr>
<tr>
<td>seasmin</td>
<td>Seasonal minimum</td>
</tr>
<tr>
<td>seasmx</td>
<td>Seasonal maximum</td>
</tr>
<tr>
<td>seassum</td>
<td>Seasonal sum</td>
</tr>
<tr>
<td>seasmean</td>
<td>Seasonal mean</td>
</tr>
<tr>
<td>seasavg</td>
<td>Seasonal average</td>
</tr>
<tr>
<td>seavar</td>
<td>Seasonal variance</td>
</tr>
<tr>
<td>seastd</td>
<td>Seasonal standard deviation</td>
</tr>
<tr>
<td>yhourmin</td>
<td>Multi-year hourly minimum</td>
</tr>
<tr>
<td>yhourmax</td>
<td>Multi-year hourly maximum</td>
</tr>
<tr>
<td>yhoursum</td>
<td>Multi-year hourly sum</td>
</tr>
<tr>
<td>yhourmean</td>
<td>Multi-year hourly mean</td>
</tr>
<tr>
<td>yhouravg</td>
<td>Multi-year hourly average</td>
</tr>
<tr>
<td>yhourvar</td>
<td>Multi-year hourly variance</td>
</tr>
<tr>
<td>yhourstd</td>
<td>Multi-year hourly standard deviation</td>
</tr>
<tr>
<td>ydaymin</td>
<td>Multi-year daily minimum</td>
</tr>
<tr>
<td>ydaymax</td>
<td>Multi-year daily maximum</td>
</tr>
<tr>
<td>ydaysum</td>
<td>Multi-year daily sum</td>
</tr>
<tr>
<td>ydaymean</td>
<td>Multi-year daily mean</td>
</tr>
<tr>
<td>ydayavg</td>
<td>Multi-year daily average</td>
</tr>
<tr>
<td>ydayvar</td>
<td>Multi-year daily variance</td>
</tr>
<tr>
<td>ydaystd</td>
<td>Multi-year daily standard deviation</td>
</tr>
<tr>
<td>ydaypctl</td>
<td>Multi-year daily percentiles</td>
</tr>
<tr>
<td>ymonmin</td>
<td>Multi-year monthly minimum</td>
</tr>
<tr>
<td>ymonmax</td>
<td>Multi-year monthly maximum</td>
</tr>
<tr>
<td>ymonsum</td>
<td>Multi-year monthly sum</td>
</tr>
<tr>
<td>ymonmean</td>
<td>Multi-year monthly mean</td>
</tr>
<tr>
<td>ymonavg</td>
<td>Multi-year monthly average</td>
</tr>
<tr>
<td>ymonvar</td>
<td>Multi-year monthly variance</td>
</tr>
<tr>
<td>ymonstd</td>
<td>Multi-year monthly standard deviation</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>ymonpctl</td>
<td>Multi-year monthly percentiles</td>
</tr>
<tr>
<td>yseasmin</td>
<td>Multi-year seasonal minimum</td>
</tr>
<tr>
<td>yseasmax</td>
<td>Multi-year seasonal maximum</td>
</tr>
<tr>
<td>yseassum</td>
<td>Multi-year seasonal sum</td>
</tr>
<tr>
<td>yseasmean</td>
<td>Multi-year seasonal mean</td>
</tr>
<tr>
<td>yseasavg</td>
<td>Multi-year seasonal average</td>
</tr>
<tr>
<td>yseasvar</td>
<td>Multi-year seasonal variance</td>
</tr>
<tr>
<td>yseasstd</td>
<td>Multi-year seasonal standard deviation</td>
</tr>
<tr>
<td>yseaspctl</td>
<td>Multi-year seasonal percentiles</td>
</tr>
<tr>
<td>ydrunmin</td>
<td>Multi-year daily running minimum</td>
</tr>
<tr>
<td>ydrunmax</td>
<td>Multi-year daily running maximum</td>
</tr>
<tr>
<td>ydrunsum</td>
<td>Multi-year daily running sum</td>
</tr>
<tr>
<td>ydrunmean</td>
<td>Multi-year daily running mean</td>
</tr>
<tr>
<td>ydrunavg</td>
<td>Multi-year daily running average</td>
</tr>
<tr>
<td>ydrunvar</td>
<td>Multi-year daily running variance</td>
</tr>
<tr>
<td>ydrunstd</td>
<td>Multi-year daily running standard deviation</td>
</tr>
<tr>
<td>ydrunpctl</td>
<td>Multi-year daily running percentiles</td>
</tr>
</tbody>
</table>
2.8.1. CONSECSTAT - Consecutive timestep periods

Synopsis

<operator> ifile ofile

Description

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

Operators

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 length of each period together with its last timestep. To be able to perform statistical analysis like min, max or mean, everything else is set to missing value.

Example

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

```
cdo consects -gtc,20.0 ifile1 ofile
```
2.8.2. ENSSTAT - Statistical values over an ensemble

Synopsis

\begin{verbatim}
ensmin ifiles ofile
ensmax ifiles ofile
enssum ifiles ofile
ensmean ifiles ofile
ensavg ifiles ofile
ensvar ifiles ofile
ensstd ifiles ofile
enspctl,p ifiles ofile
\end{verbatim}

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 \texttt{ofile}. All input files need to have the same structure with the same variables. The date information of a timestep in \texttt{ofile} is the date of the first input file.

Operators

\begin{verbatim}
ensmin Ensemble minimum
\quad o(t, x) = \min\{i_1(t, x), i_2(t, x), \cdots, i_n(t, x)\}
ensmax Ensemble maximum
\quad o(t, x) = \max\{i_1(t, x), i_2(t, x), \cdots, i_n(t, x)\}
enssum Ensemble sum
\quad o(t, x) = \sum\{i_1(t, x), i_2(t, x), \cdots, i_n(t, x)\}
enmean Ensemble mean
\quad o(t, x) = \mean\{i_1(t, x), i_2(t, x), \cdots, i_n(t, x)\}
enavg Ensemble average
\quad o(t, x) = \avg\{i_1(t, x), i_2(t, x), \cdots, i_n(t, x)\}
envar Ensemble variance
\quad o(t, x) = \var\{i_1(t, x), i_2(t, x), \cdots, i_n(t, x)\}
enstd Ensemble standard deviation
\quad o(t, x) = \std\{i_1(t, x), i_2(t, x), \cdots, i_n(t, x)\}
enpctl Ensemble percentiles
\quad o(t, x) = \texttt{pth percentile}\{i_1(t, x), i_2(t, x), \cdots, i_n(t, x)\}
\end{verbatim}

Parameter

\begin{verbatim}
p FLOAT Percentile number in 0, ..., 100
\end{verbatim}
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 ensptct,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; bresa for the Brier score resolution wrt threshold x; brsuct 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 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 and Decomposition
Example

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

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

or shorter using file name substitution:

```
cdo ensbrs,5 rfile ensfile[1-5] obase
```
2.8.5. FLDSTAT - Statistical values over a field

Synopsis

fldmin ifile ofile
fldmax ifile ofile
fldsum ifile ofile
fldmean ifile ofile
fldavg ifile ofile
fldvar ifile ofile
fldstd ifile ofile
fldpctl,p ifile ofile

Description

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

Operators

fldmin  Field minimum
For every gridpoint $x_1,\ldots,x_n$ of the same field it is:
\[ o(t,1) = \min \{ i(t,x'), x_1 < x' \leq x_n \} \]

fldmax  Field maximum
For every gridpoint $x_1,\ldots,x_n$ of the same field it is:
\[ o(t,1) = \max \{ i(t,x'), x_1 < x' \leq x_n \} \]

fldsum  Field sum
For every gridpoint $x_1,\ldots,x_n$ of the same field it is:
\[ o(t,1) = \sum \{ i(t,x'), x_1 < x' \leq x_n \} \]

fldmean  Field mean
For every gridpoint $x_1,\ldots,x_n$ of the same field it is:
\[ o(t,1) = \text{mean} \{ i(t,x'), x_1 < x' \leq x_n \} \]

fldavg  Field average
For every gridpoint $x_1,\ldots,x_n$ of the same field it is:
\[ o(t,1) = \text{avg} \{ i(t,x'), x_1 < x' \leq x_n \} \]

fldvar  Field variance
For every gridpoint $x_1,\ldots,x_n$ of the same field it is:
\[ o(t,1) = \text{var} \{ i(t,x'), x_1 < x' \leq x_n \} \]

fldstd  Field standard deviation
For every gridpoint $x_1,\ldots,x_n$ of the same field it is:
\[ o(t,1) = \text{std} \{ i(t,x'), x_1 < x' \leq x_n \} \]

fldpctl  Field percentiles
For every gridpoint $x_1,\ldots,x_n$ of the same field it is:
\[ o(t,1) = \text{pth percentile} \{ i(t,x'), x_1 < x' \leq x_n \} \]
**Parameter**

\( p \) FLOAT Percentile number in 0, ..., 100

**Example**

To compute the field mean of all input fields use:

```
cdo fldmean ifile ofile
```

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

```
cdo fldpctl,90 ifile ofile
```
2.8.6. ZONSTAT - Zonal statistical values

Synopsis

- `zonmin ifile ofile`
- `zonmax ifile ofile`
- `zonsum ifile ofile`
- `zonmean ifile ofile`
- `zonavg ifile ofile`
- `zonvar ifile ofile`
- `zonestd ifile ofile`
- `zonpctl,p ifile ofile`

Description

This module computes zonal statistical values of the input fields. According to the chosen operator the zonal minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to `ofile`. All input fields need to have the same regular lonlat grid.

Operators

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

Parameter

- **p**: FLOAT
  Percentile number in 0, ..., 100
Example

To compute the zonal mean of all input fields use:

```plaintext
cdo zonmean ifile ofile
```

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

```plaintext
cdo zonpctl,50 ifile ofile
```
2.8.7. MERSTAT - Meridional statistical values

Synopsis

mermin ifile ofile
mermax ifile ofile
mersum ifile ofile
mermean ifile ofile
meravg ifile ofile
mervar ifile ofile
merstd ifile ofile
merpctl,p ifile ofile

Description

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

Operators

- **mermin**: Meridional minimum
  - For every longitude the minimum over all latitudes is computed.
- **mermax**: Meridional maximum
  - For every longitude the maximum over all latitudes is computed.
- **mersum**: Meridional sum
  - For every longitude the sum over all latitudes is computed.
- **mermean**: Meridional mean
  - For every longitude the area weighted mean over all latitudes is computed.
- **meravg**: Meridional average
  - For every longitude the area weighted average over all latitudes is computed.
- **mervar**: Meridional variance
  - For every longitude the variance over all latitudes is computed.
- **merstd**: Meridional standard deviation
  - For every longitude the standard deviation over all latitudes is computed.
- **merpctl**: Meridional percentiles
  - For every longitude the 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

\[ <\text{operator}>, nx, ny \text{ ifile ofile} \]

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
- `gridboxmax` Gridbox maximum
- `gridboxsum` Gridbox sum
- `gridboxmean` Gridbox mean
- `gridboxavg` Gridbox average
- `gridboxvar` Gridbox variance
- `gridboxstd` Gridbox standard deviation

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 mean over all levels is computed.
- **vertavg** Vertical average
  For every gridpoint the average over all levels is computed.
- **vertvar** Vertical variance
  For every gridpoint the variance over all levels is computed.
- **vertstd** Vertical standard deviation
  For every gridpoint the standard deviation over all levels is computed.

Example

To compute the vertical sum of all input variables use:

```
cdo vertsum ifile ofile
```
2.8.10. TIMSELSTAT - Time range statistical values

Synopsis

\[ <\text{operator}>, \text{nsets}, [\text{noffset}, \text{nskip}] \] 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 date information of a timestep in ofile is the date of the last contributing timestep in ifile.

Operators

- **timselmin** Time range minimum
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same selected time range it is:
  \[ o(t, x) = \min\{i(t', x), t_1 < t' \leq t_n\} \]

- **timselmax** Time range maximum
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same selected time range it is:
  \[ o(t, x) = \max\{i(t', x), t_1 < t' \leq t_n\} \]

- **timselsum** Time range sum
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same selected time range it is:
  \[ o(t, x) = \sum\{i(t', x), t_1 < t' \leq t_n\} \]

- **timselmean** Time range mean
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same selected time range it is:
  \[ o(t, x) = \text{mean}\{i(t', x), t_1 < t' \leq t_n\} \]

- **timselavg** Time range average
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same selected time range it is:
  \[ o(t, x) = \text{avg}\{i(t', x), t_1 < t' \leq t_n\} \]

- **timselvar** Time range variance
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same selected time range it is:
  \[ o(t, x) = \text{var}\{i(t', x), t_1 < t' \leq t_n\} \]

- **timselstd** Time range standard deviation
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same selected time range it is:
  \[ o(t, x) = \text{std}\{i(t', x), t_1 < t' \leq t_n\} \]

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 time steps in ifile1. The algorithm uses histograms with minimum and maximum bounds given in ifile2 and ifile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files ifile2 and ifile3 should be the result of corresponding `timselmin` and `timselmax` operations, respectively. The date information of a timestep in ofile is the date of the last contributing timestep in ifile.

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

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

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>FLOAT</td>
<td>Percentile number in 0, ..., 100</td>
</tr>
<tr>
<td>nsets</td>
<td>INTEGER</td>
<td>Number of input timesteps for each output timestep</td>
</tr>
<tr>
<td>noffset</td>
<td>INTEGER</td>
<td>Number of input timesteps skipped before the first timestep range (optional)</td>
</tr>
<tr>
<td>nskip</td>
<td>INTEGER</td>
<td>Number of input timesteps skipped between timestep ranges (optional)</td>
</tr>
</tbody>
</table>

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 date information in ofile is the date of the middle contributing timestep in ifile.

Operators

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

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

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

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

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

- **runvar**  
  Running variance
  \[ o(t + (nts - 1)/2, x) = \std\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\} \]

- **runstd**  
  Running standard deviation
  \[ o(t + (nts - 1)/2, x) = \std\{i(t, x), i(t + 1, x), ..., i(t + nts - 1, x)\} \]

Parameter

- **nts**  INTEGER  Number of timesteps

Environment

- **RUNSTAT_DATE**  
  Sets the date information in ofile to the "first", "last" or "middle" contributing timestep in ifile.

Example

To compute the running mean over 9 timesteps use:

```
cdo runmean,9 ifile ofile
```
2.8.13. RUNPCTL - Running percentile values

Synopsis

\texttt{runpctl},\texttt{p,nts ifile1 ofile}

Description

This module computes running percentiles over a selected number of time steps in \texttt{ifile1}. The date information in \texttt{ofile} is the date of the medium contributing timestep in \texttt{ifile1}.

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

Parameter

\begin{array}{ll}
  p & \text{FLOAT} \quad \text{Percentile number in 0, ..., 100} \\
  nts & \text{INTEGER} \quad \text{Number of timesteps} \\
\end{array}

Example

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

\texttt{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 date information of a timestep in ofile is the date of the last contributing timestep in ifile.

Operators

- **timmin** Time minimum
  
  \[ o(1, x) = \min \{i(t', x), t_1 < t' \leq t_n\} \]

- **timmax** Time maximum
  
  \[ o(1, x) = \max \{i(t', x), t_1 < t' \leq t_n\} \]

- **timsum** Time sum
  
  \[ o(1, x) = \sum \{i(t', x), t_1 < t' \leq t_n\} \]

- **timmean** Time mean
  
  \[ o(1, x) = \text{mean}\{i(t', x), t_1 < t' \leq t_n\} \]

- **timavg** Time average
  
  \[ o(1, x) = \text{avg}\{i(t', x), t_1 < t' \leq t_n\} \]

- **timvar** Time variance
  
  \[ o(1, x) = \text{var}\{i(t', x), t_1 < t' \leq t_n\} \]

- **timstd** Time standard deviation
  
  \[ o(1, x) = \text{std}\{i(t', x), t_1 < t' \leq t_n\} \]

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 date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile1`.

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

Parameter

\[ p \quad \text{FLOAT} \quad \text{Percentile number in 0, ..., 100} \]

Environment

\[ \text{CDO\_PCTL\_NBINS} \quad \text{Sets the number of histogram bins. The default number is 101.} \]

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 date information of a timestep in ofile is the date of the last contributing timestep in ifile.

Operators

- **hourmin** Hourly minimum
  For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same hour it is:
  \[
  o(t, x) = \min \{ i(t', x), t_1 < t' \leq t_n \}
  \]

- **hourmax** Hourly maximum
  For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same hour it is:
  \[
  o(t, x) = \max \{ i(t', x), t_1 < t' \leq t_n \}
  \]

- **hoursum** Hourly sum
  For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same hour it is:
  \[
  o(t, x) = \sum \{ i(t', x), t_1 < t' \leq t_n \}
  \]

- **hourmean** Hourly mean
  For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same hour it is:
  \[
  o(t, x) = \text{mean} \{ i(t', x), t_1 < t' \leq t_n \}
  \]

- **houravg** Hourly average
  For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same hour it is:
  \[
  o(t, x) = \text{avg} \{ i(t', x), t_1 < t' \leq t_n \}
  \]

- **hourvar** Hourly variance
  For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same hour it is:
  \[
  o(t, x) = \text{var} \{ i(t', x), t_1 < t' \leq t_n \}
  \]

- **hourstd** Hourly standard deviation
  For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same hour it is:
  \[
  o(t, x) = \text{std} \{ i(t', x), t_1 < t' \leq t_n \}
  \]

Example

To compute the hourly mean of a time series use:

```
cdo hourmean ifile ofile
```
2.8.17. HOURPCTL - Hourly percentile values

Synopsis

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

Description

This operator computes percentiles over all 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 date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile1`. For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same hour it is:

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

Parameter

\( p \) 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 date information of a timestep in ofile is the date of the last
contributing timestep in ifile.

Operators

daymin  Daily minimum
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same day it is:
\[
o(t,x) = \min\{i(t',x), t_1 < t' \leq t_n\}
\]
daymax  Daily maximum
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same day it is:
\[
o(t,x) = \max\{i(t',x), t_1 < t' \leq t_n\}
\]
daysum  Daily sum
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same day it is:
\[
o(t,x) = \sum\{i(t',x), t_1 < t' \leq t_n\}
\]
daymean Daily mean
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same day it is:
\[
o(t,x) = \mean\{i(t',x), t_1 < t' \leq t_n\}
\]
dayavg  Daily average
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same day it is:
\[
o(t,x) = \avg\{i(t',x), t_1 < t' \leq t_n\}
\]
dayvar  Daily variance
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same day it is:
\[
o(t,x) = \var\{i(t',x), t_1 < t' \leq t_n\}
\]
daystd  Daily standard deviation
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same day it is:
\[
o(t,x) = \std\{i(t',x), t_1 < t' \leq t_n\}
\]

Example

To compute the daily mean of a time series use:

```
cdo daymean ifile ofile
```
2.8.19. DAYPCTL - Daily percentile values

Synopsis

daypctl,p ifile1 ifile2 ifile3 ofile

Description

This operator computes percentiles over all 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 date information of a timestep in ofile is the date of the last contributing timestep in ifile1.

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

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

Parameter

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

Environment

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

Example

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

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

Or shorter using operator piping:

```
cdo daypctl,90 ifile -daymin ifile -daymax ifile ofile
```
2.8.20. MONSTAT - Monthly statistical values

Synopsis

\[ \text{<operator>} 	ext{ ifile ofile} \]

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 date information of a timestep in ofile is the date of the last contributing timestep in ifile.

Operators

- **monmin**  
  Monthly minimum  
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same month it is:  
  \[ o(t, x) = \min \{i(t', x), t_1 < t' \leq t_n \} \]

- **monmax**  
  Monthly maximum  
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same month it is:  
  \[ o(t, x) = \max \{i(t', x), t_1 < t' \leq t_n \} \]

- **monsum**  
  Monthly sum  
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same month it is:  
  \[ o(t, x) = \sum \{i(t', x), t_1 < t' \leq t_n \} \]

- **monmean**  
  Monthly mean  
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same month it is:  
  \[ o(t, x) = \mean \{i(t', x), t_1 < t' \leq t_n \} \]

- **monavg**  
  Monthly average  
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same month it is:  
  \[ o(t, x) = \avg \{i(t', x), t_1 < t' \leq t_n \} \]

- **monvar**  
  Monthly variance  
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same month it is:  
  \[ o(t, x) = \var \{i(t', x), t_1 < t' \leq t_n \} \]

- **monstd**  
  Monthly standard deviation  
  For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same month it is:  
  \[ o(t, x) = \std \{i(t', x), t_1 < t' \leq t_n \} \]

Example

To compute the monthly mean of a time series use:

\[ \text{cdo monmean ifile ofile} \]
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 date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile1`. For every adjacent sequence \( t_1, \ldots, t_n \) of timesteps of the same month it is:

\[
o(t, x) = p\text{th percentile}\{i(t', x), t_1 < t' \leq t_n\}
\]

**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. 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 date information of a timestep in ofile is the date of the last contributing timestep in ifile.

Operators

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

yearmax  Yearly maximum
For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year it is:
\[ o(t, x) = \max\{i(t', x), t_1 < t' \leq t_n\} \]

yearsum  Yearly sum
For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year it is:
\[ o(t, x) = \sum\{i(t', x), t_1 < t' \leq t_n\} \]

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

yearavg  Yearly average
For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year it is:
\[ o(t, x) = \text{avg}\{i(t', x), t_1 < t' \leq t_n\} \]

yearvar  Yearly variance
For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year it is:
\[ o(t, x) = \text{var}\{i(t', x), t_1 < t' \leq t_n\} \]

yearstd  Yearly standard deviation
For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year it is:
\[ o(t, x) = \text{std}\{i(t', x), t_1 < t' \leq t_n\} \]

Note

The operators yearmean and yearavg compute only arithmetical means!

Example

To compute the yearly mean of a time series use:

```
cdo yearmean ifile ofile
```

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

```
cdo divdpy -yearsum -muldpm ifile ofile
```
2.8.23. YEARPCTL - Yearly percentile values

Synopsis

\texttt{yearpctl,p ifile1 ifile2 ifile3 ofile}

Description

This operator computes percentiles over all timesteps of the same year in \texttt{ifile1}. The algorithm uses histograms with minimum and maximum bounds given in \texttt{ifile2} and \texttt{ifile3}, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable \texttt{CDO\_PCTL\_NBINS} to a different value. The files \texttt{ifile2} and \texttt{ifile3} should be the result of corresponding \texttt{yearmin} and \texttt{yearmax} operations, respectively. The date information of a timestep in \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile1}. For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same year it is:

\[
o(t, x) = p\text{th percentile}\{i(t', x), t_1 < t' \leq t_n\}
\]

Parameter

\( p \) FLOAT Percentile number in 0, ..., 100

Environment

\texttt{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:

\begin{verbatim}
cdo yearmin ifile minfile
cdo yearmax ifile maxfile
cdo yearpctl,90 ifile minfile maxfile ofile
\end{verbatim}

Or shorter using operator piping:

\begin{verbatim}
cdo yearpctl,90 ifile -yearmin ifile -yearmax ifile ofile
\end{verbatim}
2.8.24. 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 date information of a timestep in ofile is the date of the last contributing timestep in 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:
\[
o(t, x) = \min\{ i(t', x), t_1 < t' \leq t_n \}
\]

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

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

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

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

seasvar  Seasonal variance
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same season it is:
\[
o(t, x) = \text{var}\{ i(t', x), t_1 < t' \leq t_n \}
\]

seasstd  Seasonal standard deviation
For every adjacent sequence \( t_1, ..., t_n \) of timesteps of the same season it is:
\[
o(t, x) = \text{std}\{ i(t', x), t_1 < t' \leq t_n \}
\]

Example

To compute the seasonal mean of a time series use:

```
cdo seasmean ifile ofile
```
2.8.25. SEASPCTL - Seasonal percentile values

Synopsis

seaspctl,p ifile1 ifile2 ifile3 ofile

Description

This operator computes percentiles over all 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 date information of a timestep in ofile is the date of the last contributing timestep in 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 \( t_1, \ldots, t_n \) of timesteps of the same season it is:

\[
o(t, x) = p\text{th percentile}\{i(t', x), t_1 < t' \leq t_n\}
\]

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:

\[
\begin{align*}
cdo & \text{ seasmin ifile minfile} \\
cdo & \text{ seasmax ifile maxfile} \\
cdo & \text{ seaspctl,90 ifile minfile maxfile ofile}
\end{align*}
\]

Or shorter using operator piping:

\[
\text{cdo seaspctl,90 ifile -seasmin ifile -seasmax ifile ofile}
\]
2.8.26. YHOURSTAT - Multi-year hourly statistical values

Synopsis

<operator> ifile ofile

Description

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

Operators

- **yhourmin**  Multi-year hourly minimum
  
  \( o(0001, x) = \min\{i(t, x), \text{day}(i(t)) = 0001\} \)
  
  \( \vdots \)
  
  \( o(8784, x) = \min\{i(t, x), \text{day}(i(t)) = 8784\} \)

- **yhourmax**  Multi-year hourly maximum
  
  \( o(0001, x) = \max\{i(t, x), \text{day}(i(t)) = 0001\} \)
  
  \( \vdots \)
  
  \( o(8784, x) = \max\{i(t, x), \text{day}(i(t)) = 8784\} \)

- **yhoursum**  Multi-year hourly sum
  
  \( o(0001, x) = \sum\{i(t, x), \text{day}(i(t)) = 0001\} \)
  
  \( \vdots \)
  
  \( o(8784, x) = \sum\{i(t, x), \text{day}(i(t)) = 8784\} \)

- **yhourmean**  Multi-year hourly mean
  
  \( o(0001, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 0001\} \)
  
  \( \vdots \)
  
  \( o(8784, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 8784\} \)

- **yhouravg**  Multi-year hourly average
  
  \( o(0001, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 0001\} \)
  
  \( \vdots \)
  
  \( o(8784, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 8784\} \)

- **yhourvar**  Multi-year hourly variance
  
  \( o(0001, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 0001\} \)
  
  \( \vdots \)
  
  \( o(8784, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 8784\} \)

- **yhourstd**  Multi-year hourly standard deviation
  
  \( o(0001, x) = \text{std}\{i(t, x), \text{day}(i(t)) = 0001\} \)
  
  \( \vdots \)
  
  \( o(8784, x) = \text{std}\{i(t, x), \text{day}(i(t)) = 8784\} \)

Example

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

```
cdo yhourmean ifile ofile
```
2.8.27. YDAYSTAT - Multi-year daily statistical values

Synopsis

<operator> ifile ofile

Description

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

Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ydaymin</td>
<td>Multi-year daily minimum</td>
</tr>
<tr>
<td>ydaymax</td>
<td>Multi-year daily maximum</td>
</tr>
<tr>
<td>ydaysum</td>
<td>Multi-year daily sum</td>
</tr>
<tr>
<td>ydaymean</td>
<td>Multi-year daily mean</td>
</tr>
<tr>
<td>ydayavg</td>
<td>Multi-year daily average</td>
</tr>
<tr>
<td>ydayvar</td>
<td>Multi-year daily variance</td>
</tr>
<tr>
<td>ydaystd</td>
<td>Multi-year daily standard deviation</td>
</tr>
</tbody>
</table>

Example

To compute the daily mean over all input years use:

cdo ydaymean ifile ofile
2.8.28. YDAYPCTL - Multi-year daily percentile values

Synopsis

\texttt{ydaypctl,p ifile1 ifile2 ifile3 ofile}

Description

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

\[
o(001,x) = p\text{th percentile}\{i(t,x), \text{day}(i(t)) = 001\}
\]

\[
\vdots
\]

\[
o(366,x) = p\text{th percentile}\{i(t,x), \text{day}(i(t)) = 366\}
\]

Parameter

\[
p \quad \text{FLOAT} \quad \text{Percentile number in 0, ..., 100}
\]

Environment

\[
\texttt{CDO\_PCTL\_NBINS} \quad \text{Sets the number of histogram bins. The default number is 101.}
\]

Example

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

\begin{verbatim}
cdo ydaymin ifile minfile
cdo ydaymax ifile maxfile
cdo ydaypctl,90 ifile minfile maxfile ofile
\end{verbatim}

Or shorter using operator piping:

\begin{verbatim}
cdo ydaypctl,90 ifile -ydaymin ifile -ydaymax ifile ofile
\end{verbatim}
2.8.29. YMONSTAT - Multi-year monthly statistical values

Synopsis

<operator> ifile ofile

Description

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

Operators

ymonmin  Multi-year monthly minimum
  o(01, x) = min{i(t, x), month(i(t)) = 01} 
  : 
  o(12, x) = min{i(t, x), month(i(t)) = 12}

ymonmax  Multi-year monthly maximum
  o(01, x) = max{i(t, x), month(i(t)) = 01} 
  : 
  o(12, x) = max{i(t, x), month(i(t)) = 12}

ymonsum  Multi-year monthly sum
  o(01, x) = sum{i(t, x), month(i(t)) = 01} 
  : 
  o(12, x) = sum{i(t, x), month(i(t)) = 12}

ymonmean Multi-year monthly mean
  o(01, x) = mean{i(t, x), month(i(t)) = 01} 
  : 
  o(12, x) = mean{i(t, x), month(i(t)) = 12}

ymonavg  Multi-year monthly average
  o(01, x) = avg{i(t, x), month(i(t)) = 01} 
  : 
  o(12, x) = avg{i(t, x), month(i(t)) = 12}

ymonvar  Multi-year monthly variance
  o(01, x) = var{i(t, x), month(i(t)) = 01} 
  : 
  o(12, x) = var{i(t, x), month(i(t)) = 12}

ymonstd  Multi-year monthly standard deviation
  o(01, x) = std{i(t, x), month(i(t)) = 01} 
  : 
  o(12, x) = std{i(t, x), month(i(t)) = 12}

Example

To compute the monthly mean over all input years use:

```
cdo ymonmean ifile ofile
```
2.8.30. YMONPCTL - Multi-year monthly percentile values

Synopsis

ymonpctl,p ifile1 ifile2 ifile3 ofile

Description

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

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

\[
\vdots
\]

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

Parameter

\[p\]  FLOAT  Percentile number in 0, ..., 100

Environment

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

Example

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

```
cdo ymonmin ifile minfile
```

```
cdo ymonmax ifile maxfile
```

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

Or shorter using operator piping:

```
cdo ymonpctl,90 ifile -ymonmin ifile -ymonmax ifile ofile
```
2.8.31. YSEASSTAT - Multi-year seasonal statistical values

Synopsis

<operator> ifile ofile

Description

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

Operators

yseasmin Multi-year seasonal minimum

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

yseasmax Multi-year seasonal maximum

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

yseassum Multi-year seasonal sum

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

yseasmean Multi-year seasonal mean

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

yseasavg Multi-year seasonal average

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

yseasvar Multi-year seasonal variance

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

yseasstd Multi-year seasonal standard deviation

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

To compute the seasonal mean over all input years use:

```
cdo yseasmean ifile ofile
```

2.8.32. YSEASPCTL - Multi-year seasonal percentile values

Synopsis

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

Description

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

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

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

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

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

Parameter

\[
p \quad \text{FLOAT} \quad \text{Percentile number in 0, ..., 100}
\]

Environment

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

Example

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

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

Or shorter using operator piping:

```
cdo yseaspctl,90 ifile -yseasmin ifile -yseasmax ifile ofile
```
2.8.33. YDRUNSTAT - Multi-year daily running statistical values

Synopsis

<operator>,nts ifile ofile

Description

This module writes running statistical values for each day of year in ifile to ofile. Depending on the chosen operator, the minimum, maximum, sum, average, variance or standard deviation of all 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 medium timestep in the last contributing running window. Note that the operator have to be applied to a continuous time series of daily measurements in order to yield physically meaningful results. Also note that the output time series begins (nts-1)/2 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

\[ o(001, x) = \min\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 001\} \]

\[ \vdots \]

\[ o(366, x) = \min\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 366\} \]

ydrunmax Multi-year daily running maximum

\[ o(001, x) = \max\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 001\} \]

\[ \vdots \]

\[ o(366, x) = \max\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 366\} \]

ydrunsum Multi-year daily running sum

\[ o(001, x) = \sum\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 001\} \]

\[ \vdots \]

\[ o(366, x) = \sum\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 366\} \]

ydrunmean Multi-year daily running mean

\[ o(001, x) = \text{mean}\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 001\} \]

\[ \vdots \]

\[ o(366, x) = \text{mean}\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 366\} \]

ydrunavg Multi-year daily running average

\[ o(001, x) = \text{avg}\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 001\} \]

\[ \vdots \]

\[ o(366, x) = \text{avg}\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 366\} \]

ydrunvar Multi-year daily running variance

\[ o(001, x) = \text{var}\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 001\} \]

\[ \vdots \]

\[ o(366, x) = \text{var}\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 366\} \]

ydrunstd Multi-year daily running standard deviation

\[ o(001, x) = \text{std}\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 001\} \]

\[ \vdots \]

\[ o(366, x) = \text{std}\{i(t, x), i(t + 1, x), \ldots, i(t + nts - 1, x); \text{day}[i(t + (nts - 1)/2)] = 366\} \]
Parameter

\texttt{nts} \hspace{1em} \textbf{INTEGER} \hspace{1em} 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:

\texttt{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:

\texttt{cdo ydaymean -runmean,5 ifile ofile}
2.8.34. YDRUNPCTL - Multi-year daily running percentile values

Synopsis

ydrunpctl,p,nts ifile1 ifile2 ifile3 ofile

Description

This operator writes running percentile values for each day of year in ifile1 to ofile. A certain percentile is computed for all 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 medium time step in the last contributing running window. Note that the operator have to be applied to a continuous time series of daily measurements in order to yield physically meaningful results. Also note that the output time series begins (nts-1)/2 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.

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

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

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:

\[
\text{cdo ydrunmin,5 ifile minfile} \\
\text{cdo ydrunmax,5 ifile maxfile} \\
\text{cdo ydrunpctl,90,5 ifile minfile maxfile ofile}
\]

Or shorter using operator piping:

\[
\text{cdo ydrunpctl,90,5 ifile -ydrunmin ifile -ydrunmax ifile ofile}
\]
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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>covar</td>
<td>Covariance</td>
</tr>
<tr>
<td>covar weighted by {w_i, i = 1, ..., n}</td>
<td>Covariance weighted by weights {w_i}</td>
</tr>
</tbody>
</table>

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

Covariance

\[
\text{covar} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})^2
\]

Covariance weighted by \{w_i, i = 1, ..., n\}

\[
\text{covar weighted by } \{w_i, i = 1, ..., n\} = \left( \frac{n}{\sum_{j=1}^{n} w_j} \right)^{-1} \sum_{i=1}^{n} w_j \left( x_i - \left( \frac{n}{\sum_{j=1}^{n} w_j} \right) \sum_{j=1}^{n} w_j x_j \right) \left( y_i - \left( \frac{n}{\sum_{j=1}^{n} w_j} \right) \sum_{j=1}^{n} w_j y_j \right)
\]
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

\[ S(t) = \{ x, i_1(t, x) \neq \text{missval} \land i_2(t, x) \neq \text{missval} \} \]

it is

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

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

2.9.2. TIMCOR - Correlation over time

Synopsis

```
timcor ifile1 ifile2 ofile
```

Description

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

\[ S(x) = \{ t, i_1(t, x) \neq \text{missval} \land i_2(t, x) \neq \text{missval} \} \]

it is

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

For every gridpoint \( x \) only those timesteps \( t \) belong to the sample, which have \( i_1(t, x) \neq \text{missval} \) and \( i_2(t, x) \neq \text{missval} \).
2.9.3. 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

\[ S(t) = \{ x, i_1(t,x) \neq \text{missval} \land i_2(t,x) \neq \text{missval} \} \]

it is

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

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

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

\[ S(x) = \{ t, i_1(t,x) \neq \text{missval} \land i_2(t,x) \neq \text{missval} \} \]

it is

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

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

This section 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 \( \sigma^2 \). This operator estimates the parameter \( b \). For every field element \( x \) only those timesteps \( t \) belong to the sample \( S(x) \), which have \( i(t, x) \neq \text{miss} \). It is

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

2.10.2. DETREND - Detrend time series

Synopsis

detrend ifile ofile

Description

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

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

and

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

it is

\[
o(t, x) = i(t, x) - (a(x) + b(x)t)
\]

Note

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

Example

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

cdo detrend ifile ofile
2.10.3. TREND - Trend of time series

Synopsis

trend ifile ofile1 ofile2

Description

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

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

and

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

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]
- Statistical Analysis in Climate Research [von Storch]

EOFs are defined as the eigenvalues 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.

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

the scatter matrix \(S\) can be written as

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

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

The matrix \(S\) 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 \(S\) and not the weighted eigenvector which would be \(W e_j\).) Let the corresponding eigenvalues be denoted \(\lambda_j\). The vectors \(e_j\) are spatial patterns which explain a certain amount of variance of the time series \(z(t)\) that is related linearly to \(\lambda_j\). Thus, the spatial pattern defined by the first eigenvector (the one with the largest eigenvalue) is the pattern which explains a maximum possible amount of variance of the sample \(z(t)\). The orthonormality of eigenvectors reads as

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

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

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

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

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

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
- `eofs spatial` 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

\(<\text{operator}>,\text{neof} \> \text{ifile ofile1 ofile2}\)

Description

This module calculates empirical orthogonal functions of the data in \text{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.

\textbf{Please note, that the input data are assumed to be anomalies.}

If operator \text{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 \text{eoftime} or \text{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 \text{ifile} are assumed to be anomalies. If they are not, the behavior of this module is \textbf{not well defined}. After execution \text{ofile1} will contain all eigen-values and \text{ofile2} the eigenvectors $e_j$. All EOFs and eigen-values are computed. However, only the first \text{neof} EOFs are written to \text{ofile2}. Nonetheless, \text{ofile1} contains all eigen-values. Note, that the resulting EOF in \text{ofile2} is $e_j$ and thus \textbf{not weighted} for consistency.

Missing values are not fully supported. Support is only checked for non-changing masks of missing values in time. Although there still will be results, they are not trustworthy, and a warning will occur.

In the latter case we suggest to replace missing values by 0 in \text{ifile}.

Operators

\text{eof} \hspace{1cm} \text{Calculate EOFs in spatial or time space}

\text{eoftime} \hspace{1cm} \text{Calculate EOFs in time space}

\text{eofspatial} \hspace{1cm} \text{Calculate EOFs in spatial space}

\text{eof3d} \hspace{1cm} \text{Calculate 3-Dimensional EOFs in time space}

Parameter

\text{neof} \hspace{1cm} \text{INTEGER} \hspace{1cm} \text{Number of eigen functions}

Environment

\text{CDO\_SVD\_MODE} \hspace{1cm} \text{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’}.

\text{MAX\_JACOBI\_ITER} \hspace{1cm} \text{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.}

\text{FNORM\_PRECISION} \hspace{1cm} \text{Is the Frobenius norm of the matrix consisting of an annihilation pair of eigenvectors that is used to determine if the eigenvectors have reached a sufficient level of convergence. If all annihilation-pairs of vectors have a norm below this value, the computation is considered to have converged properly. Otherwise, a warning will occur. The default value 1e-12.}
Example

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

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

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

```
cdo sub ifile1 -timmean ifile1 anom_file
cdo eof,40 anom_file ofile1 ofile2
```
2.11.2. EOFCOEFF - Principal coefficients of EOFs

Synopsis

eofcoeff ifile1 ifile2 obase

Description

This module calculates the time series of the principal coefficients for given EOF (empirical orthogonal functions) and data. Time steps in ifile1 are assumed to be the EOFs, Time steps in ifile2 are assumed to be the time series. Weights are taken into account, which is why EOF output is not weighted. Note, that this operator calculates a weighted dot product of the fields in ifile1 and ifile2. Given a set of EOFs $e_j$ and a time series of data $z(t)$ with $p$ entries for each timestep from which $e_j$ have been calculated, this operator calculates the time series of the projections of data onto each EOF

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

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

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

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

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

Example

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

```
cdo eof,40 anom_file eval_file eof_file
cdo eofcoeff eof_file anom_file obase
```

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

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

```
cdo sub ifile -timmean ifile anom_file
cdo eof,40 anom_file eval_file eof_file
cdo eofcoeff eof_file anom_file obase
```

128
# 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:

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

Synopsis

<operator>,grid ifile ofile

Description

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

Operators

- **remapbil**  
  Bilinear interpolation  
  Performs a bilinear interpolation on all input fields. This interpolation method only works on quadrilateral curvilinear grids.

- **remapbic**  
  Bicubic interpolation  
  Performs a bicubic interpolation on all input fields. This interpolation method only works on quadrilateral curvilinear grids.

- **remapdis**  
  Distance-weighted average remapping  
  Performs a distance-weighted average remapping of the four nearest neighbor values on all input fields.

- **remapnn**  
  Nearest neighbor remapping  
  Performs a nearest neighbor remapping on all input fields.

- **remapcon**  
  First order conservative remapping  
  Performs a first order conservative remapping on all input fields.

- **remapcon2**  
  Second order conservative remapping  
  Performs a second order conservative remapping on all input fields.

- **remaplaf**  
  Largest area fraction remapping  
  Performs a largest area fraction remapping on all input fields.

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 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.

Note

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

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

```
cdo remapbil,n32 ifile ofile
```
2.12.2. GENWEIGHTS - Generate SCRIP grid interpolation weights

Synopsis

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

Description

Interpolation between different horizontal grids can be a very time-consuming process. Especially if the data are on an unstructured or a large grid. In this case the SCRIP interpolation process can be split into two parts. Firstly the generation of the interpolation weights, which is the most time-consuming part. These interpolation weights can be reused for every remapping process with the operator `remap`. This method should be used only if all input fields are on the same grid and a possibly mask (missing values) does not change. This module contains operators to generate SCRIP interpolation weights of the first input field. Each operator is using a different interpolation method.

Operators

- **genbil**: Generate bilinear interpolation weights
  Generates bilinear interpolation weights and writes the result to a file. This interpolation method only works on quadrilateral curvilinear grids.

- **genbic**: Generate bicubic interpolation weights
  Generates bicubic interpolation weights and writes the result to a file. This interpolation method only works on quadrilateral curvilinear grids.

- **gendis**: Generate distance-weighted average remap weights
  Generates distance-weighted average remapping weights of the four nearest neighbor values and writes the result to a file.

- **gennn**: Generate nearest neighbor remap weights
  Generates nearest neighbor remapping weights and writes the result to a file.

- **gencon**: Generate 1st order conservative remap weights
  Generates first order conservative remapping weights and writes the result to a file.

- **gencon2**: Generate 2nd order conservative remap weights
  Generates second order conservative remapping weights and writes the result to a file.

- **genlaf**: Generate largest area fraction remap weights
  Generates largest area fraction remapping weights and writes the result to a file.

Parameter

- **grid** STRING  Target grid description file or name

Environment

- **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.
Note

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

Example

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

```
cdo genbil,n32 ifile remapweights.nc
cdo remap,n32,remapweights.nc ifile ofile
```
2.12.3. REMAP - SCRIP grid remapping

Synopsis

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

Description

This operator remaps all input fields to a new horizontal grid. The remap type and the interpolation weights of one input grid are read from a netCDF file. More weights are computed if the input fields are on different grids. The netCDF file with the weights should follow the SCRIP convention. Normally these weights come from a previous call to module GENWEIGHTS or were created by the original SCRIP package.

Parameter

- **grid**: STRING
  - Target grid description file or name
- **weights**: STRING
  - Interpolation weights (SCRIP netCDF file)

Environment

- **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.

Note

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

Example

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

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

The result will be the same as:

```
cdo remapbil,n32 ifile ofile
```
2.12.4. REMAPETA - Remap vertical hybrid level

Synopsis

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

Description

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

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

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

Parameter

- **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:

```plaintext
cdo remapeta,vct ifile ofile
```

Here is an example vct file with 19 hybrid model level:

```
0 0.00000000000000000000 0.00000000000000000000
1 2000.00000000000000000 0.00000000000000000000
2 4000.00000000000000000 0.00000000000000000000
3 6046.10937500000000000 0.00000000000000000000
4 8267.92968750000000000 0.00000000000000000000
5 10609.5117187500000000 0.00000000000000000000
6 12851.1015625000000000 0.00000000000000000000
7 14698.5000000000000000 0.00000000000000000000
8 15861.1289062500000000 0.00000000000000000000
9 16116.2382812500000000 0.00000000000000000000
10 15356.9218750000000000 0.00000000000000000000
11 13621.4609375000000000 0.00000000000000000000
12 11101.5585937500000000 0.00000000000000000000
13 8127.1445312500000000 0.00000000000000000000
14 5125.1406250000000000 0.00000000000000000000
15 2549.9689941406250000 0.00000000000000000000
16 783.1950683593750000 0.00000000000000000000
17 0.00000000000000000000 0.00000000000000000000
18 0.00000000000000000000 0.00000000000000000000
19 0.00000000000000000000 0.00000000000000000000
```
2.12.5. INTVERT - Vertical interpolation

Synopsis

\texttt{ml2pl,plevels ifile ofile}  
\texttt{ml2hl,hlevels ifile ofile}

Description

Interpolate 3D variables on hybrid model levels to pressure or height levels. The input file should contain the log. surface pressure or the surface pressure. To interpolate the temperature, the orography (surface geopotential) is also needed. The pressure, temperature, and orography are identified by their code numbers. Supported parameter tables are: WMO standard table number 2 and ECMWF local table number 128. Use the alias \texttt{ml2plx/ml2hlx} or the environment variable \texttt{EXTRAPOLATE} to extrapolate missing values. All input fields have to be on the same horizontal grid.

Operators

\begin{itemize}
  \item \texttt{ml2pl} Model to pressure level interpolation  
  Interpolates 3D variables on hybrid model levels to pressure levels.
  \item \texttt{ml2hl} Model to height level interpolation  
  Interpolates 3D variables on hybrid model levels to height levels. The procedure is the same as for the operator \texttt{ml2pl} except for the pressure levels being calculated from the heights by: \( plevel = 101325 \ast \exp(hlevel/ - 7000) \)
\end{itemize}

Parameter

\begin{itemize}
  \item \texttt{plevels} FLOAT  
  Pressure levels in pascal
  \item \texttt{hlevels} FLOAT  
  Height levels in meter (max level: 65535 m)
\end{itemize}

Environment

\begin{itemize}
  \item \texttt{EXTRAPOLATE}  
  If set to 1 extrapolate missing values.
\end{itemize}

Note

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

Example

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

\begin{verbatim}
cdo ml2pl,92500,85000,50000,20000 ifile ofile
\end{verbatim}
2.12.6. INTLEVEL - Linear level interpolation

Synopsis

\texttt{intlevel,levels ifile ofile}

Description

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

Parameter

\begin{itemize}
  \item \textit{levels} \texttt{FLOAT} \hspace{1cm} \text{Target levels}
\end{itemize}

Example

To interpolate 3D variables on height levels to a new set of height levels use:

\begin{verbatim}
cdo intlevel,10,50,100,500,1000 ifile ofile
\end{verbatim}
2.12.7. INTLEVEL3D - Linear level interpolation from/to 3d vertical coordinates

Synopsis

\[ \text{operator}, \text{icoordinate} \text{ ifile1 ifile2 ofile} \]

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  
  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 vert. coordinate
- ifile1 contains the source data, which the vertical coordinate from icoordinate belongs to
- ifile2 only contains the target 3d height levels

```
  cdo intlevel3d,icoordinate ifile1 ifile2 ofile
```
2.12.8. **INTTIME - Time interpolation**

**Synopsis**

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

**Description**

This module performs linear interpolation between 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
```
2.12.9. INTYEAR - Year interpolation

Synopsis

\texttt{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 \texttt{<obase><yyyy><suffix>} where \texttt{yyyy} will be the year and \texttt{suffix} is the filename extension derived from the file format.

Parameter

\begin{itemize}
  \item \texttt{years} INTEGER Comma separated list of years
\end{itemize}

Environment

\begin{itemize}
  \item \texttt{CDO\_FILE\_SUFFIX} This environment variable can be used to set the default file suffix. This suffix will be added to the output file names instead of the filename extension derived from the file format. Set this variable to NULL to disable the adding of a file suffix.
\end{itemize}

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:

\begin{verbatim}
\end{verbatim}

Example result of \texttt{dir year*} for netCDF datasets:

\begin{verbatim}
\end{verbatim}
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

sp2gp ifile ofile
sp2gpl ifile ofile
gp2sp ifile ofile
gp2spl ifile ofile
sp2sp,trunc ifile ofile

Description

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

Operators

sp2gp  Spectral to gridpoint
Convert all fields with spectral coefficients to a regular Gaussian grid. The number of
latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:
\[ n_{\text{lat}} = \text{NINT} \left( \frac{3 \times \text{trunc} + 1}{2} \right) \]

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:
\[ n_{\text{lat}} = \text{NINT} \left( \frac{2 \times \text{trunc} + 1}{2} \right) \]
Use this operator to convert ERA40 data e.g. from TL159 to N80.

gp2sp  Gridpoint to spectral
Convert all Gaussian gridpoint fields to spectral coefficients. The triangular truncation
of the resulting spherical harmonics is calculated from the number of latitudes by:
\[ \text{trunc} = \frac{n_{\text{lat}} \times 2 - 1}{3} \]

gp2spl  Gridpoint to spectral (linear)
Convert all Gaussian gridpoint fields to spectral coefficients. The triangular truncation
of the resulting spherical harmonics is calculated from the number of latitudes by:
\[ \text{trunc} = \frac{n_{\text{lat}} \times 2 - 1}{2} \]
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
wnums  INTEGER  Comma separated list of wave numbers
Example

To transform spectral coefficients from T106 to N80 Gaussian grid use:
```
cdo sp2gp ifile ofile
```
To transform spectral coefficients from TL159 to N80 Gaussian grid use:
```
cdo sp2gpl ifile ofile
```
2.13.2. WIND - Wind transformation

Synopsis

<operator> ifile ofile

Description

This module converts relative divergence and vorticity to U and V wind and vice versa. Divergence and vorticity are spherical harmonic coefficients in spectral space and U and V are on a regular Gaussian grid.

Operators

- **dv2uv**  
  Divergence and vorticity to U and V wind  
  Calculate U and V wind on a Gaussian grid from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need to have the names sd and svo or code numbers 155 and 138. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:  
  \[ nlat = \text{NINT}((\text{trunc} \times \frac{3}{2} + 1.)/2. \]  

- **dv2uvl**  
  Divergence and vorticity to U and V wind (linear)  
  Calculate U and V wind on a Gaussian grid from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need to have the names sd and svo or code numbers 155 and 138. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:  
  \[ nlat = \text{NINT}((\text{trunc} \times \frac{2}{2} + 1.)/2. \]  

- **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:  
  \[ \text{trunc} = (nlat \times 2 - 1)/3 \]  

- **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:  
  \[ \text{trunc} = (nlat \times 2 - 1)/2 \]  

- **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
```

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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>import_binary</td>
<td>Import binary data sets</td>
</tr>
<tr>
<td>import_cmsaf</td>
<td>Import CM-SAF HDF5 files</td>
</tr>
<tr>
<td>import AMSR</td>
<td>Import AMSR binary files</td>
</tr>
<tr>
<td>input</td>
<td>ASCII input</td>
</tr>
<tr>
<td>inputsrv</td>
<td>SERVICE ASCII input</td>
</tr>
<tr>
<td>inputext</td>
<td>EXTRA ASCII input</td>
</tr>
<tr>
<td>output</td>
<td>ASCII output</td>
</tr>
<tr>
<td>outputf</td>
<td>Formatted output</td>
</tr>
<tr>
<td>outputint</td>
<td>Integer output</td>
</tr>
<tr>
<td>outputsrv</td>
<td>SERVICE ASCII output</td>
</tr>
<tr>
<td>outputext</td>
<td>EXTRA ASCII output</td>
</tr>
</tbody>
</table>
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 00:00 Z15jun1989 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
golocation 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/amsr). 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 output.

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 outputsrv.

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 outputext.

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.

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 (nelem max = gridsize) |
Example

To print all field elements of a dataset formatted with "%8.4g" and 8 values per line use:

```
cdo outputf,%8.4g,8 ifile
```

Example result of a dataset with one field on 64 grid points:

```
   261.7  262  257.8  252.5  248.8  247.7  246.3  246.1
   250.6  252.6  253.9  254.8  252  246.6  249.7  257.9
   273.4  266.2  259.8  261.6  257.2  253.4  251  263.7
   267.5  267.4  272.2  266.7  259.6  255.2  272.9  277.1
   275.3  275.5  276.4  278.4  282  269.6  278.7  279.5
   282.3  284.5  280.3  280.3  280  281.5  284.7  283.6
   292.9  290.5  293.9  292.6  292.7  292.8  294.1  293.6
   293.8  292.6  291.2  292.6  293.2  292.8  291  291.2
```
2.15. Miscellaneous

This section contains miscellaneous modules which do not fit to the other sections before.

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

- **gradsdes1**: GrADS data descriptor file (version 1 GRIB map)
- **gradsdes2**: GrADS data descriptor file (version 2 GRIB map)
- **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
- **stdatm**: Create values for pressure and temperature for hydrostatic atmosphere
- **rotuvb**: Backward rotation
- **mastrfu**: Mass stream function
- **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

\texttt{<operator> ifile}

Description

Creates a GrADS data descriptor file. Supported file formats are GRIB1, SERVICE, EXTRA and IEG. For GRIB1 files the GrADS map file is also generated. For SERVICE and EXTRA files the grid have to be specified with the CDO option `\texttt{-g <grid>`'. This module takes \texttt{ifile} in order to create filenames for the descriptor (\texttt{ifile.ctl}) and the map (\texttt{ifile.gmp}) file. "gradsdes" is an alias for \texttt{gradsdes2}.

Operators

\textbf{gradsdes1} GrADS data descriptor file (version 1 GRIB map)

Creates a GrADS data descriptor file. Generated a machine specific version 1 GrADS map file for GRIB1 datasets.

\textbf{gradsdes2} GrADS data descriptor file (version 2 GRIB map)

Creates a GrADS data descriptor file. Generated a machine independent version 2 GrADS map file for GRIB1 datasets. This map file can be used only with GrADS version 1.8 or newer.

Example

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

\begin{verbatim}
cdo gradsdes2 ifile.grb
\end{verbatim}

This will create a descriptor file with the name \texttt{ifile.ctl} and the map file \texttt{ifile.gmp}. Assumed the input GRIB1 dataset has 3 variables over 12 timesteps on a Gaussian N16 grid. The contents of the resulting GrADS description file is approximately:

\begin{verbatim}
DSET `ifile.grb
DTYPE GRIB
INDEX `ifile.gmp
XDEF 64 LINEAR 0.000000 5.625000
YDEF 32 LEVELS

ZDEF 4 LEVELS 925 850 500 200
TDEF 12 LINEAR 12:00 Z1jan1987 1mo
TITLE ifile.grb T21 grid
OPTIONS yrev
UNDEF -9e+33
VARS 3
geosp 0 129,1,0 surface geopotential (orography) [m^2/s^2]
t 4 130,99,0 temperature [K]
tsml1 0 139,1,0 surface temperature of land [K]
ENDVARS
\end{verbatim}
2.15.2. FILTER - Time series filtering

Synopsis

\[
\begin{align*}
\text{bandpass, } & f_{\text{min}}, f_{\text{max}} \quad \text{ifile ofile} \\
\text{lowpass, } & f_{\text{max}} \quad \text{ifile ofile} \\
\text{highpass, } & f_{\text{min}} \quad \text{ifile ofile}
\end{align*}
\]

Description

This module takes the time series for each gridpoint in ifile and fills it with zeros (zero-padding) up to the next time-step-number that is a power of 2. Then it (fast fourier) transforms the time series with \(2^n\) elements into the frequency domain. According to the particular operator and its parameters certain frequencies are filtered (set to zero) in the frequency domain and the spectrum is (inverse fast fourier) transformed back into the time domain. This time series is cut to the original number of timesteps from ifile and written to ofile. To determine the frequency the time-axis of ifile is used. (Data should have a constant time increment since this assumption applies for transformation. However, the time increment has to be different from zero.) All frequencies given as parameter are interpreted per year. This is done by the assumption of a 365-day calendar. Consequently if you want to perform multiyear-filtering accurately you have to delete the 29th of February. If your ifile has a 360 year calendar the frequency parameters fmin respectively fmax should be multiplied with a factor of 360/365 in order to obtain accurate results. For the set up of a frequency filter the frequency parameters have to be adjusted to a frequency in the data. Here fmin is rounded down and fmax is always rounded up. Consequently it is possible to use bandpass with fmin=fmax without getting a zero-field for ofile. Hints for efficient usage:

- to avoid effects of zero-padding cut or extend your time series down/up to the nearest power of two
- to get reliable results the time-series has to be detrended (cdo detrend)
- the lowest frequency greater zero that can be contained in ifile is \(1/(N*dT)\),
- the greatest frequency is \(1/(2dT)\) (Nyquist frequency),

with N the number of 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 \([f_{\text{min}}, f_{\text{max}}]\).
- **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 variability 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). Therefore you can use the following:

```
cdo highpass,x -del29feb ifile ofile
```

Accordingly you might use the following to suppress variability on timescales shorter than one year:

```
cdo lowpass,1 -del29feb ifile ofile
```

Finally you might be interested in 2-year variability. If you want to suppress the seasonal cycle as well as say the longer cycles in climate system you might use

```
cdo bandpass,x,y -del29feb ifile ofile
```

with x<=0.5 and y >=0.5.

2.15.3. GRIDCELL - Grid cell quantities

Synopsis

```
<operator> ifile ofile
```

Description

This module reads the grid cell area of the first grid from the input stream. If the grid cell area is missing it will be computed from the grid description. Depending on the chosen operator the grid cell area or weights are written to the output stream.

Operators

- **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.
2.15.4. SMOOTH9 - 9 point smoothing

Synopsis

smooth9 ifile ofile

Description

Performs a 9 point smoothing on all fields with a quadrilateral curvilinear grid. The result at each grid point is a weighted average of the grid point plus the 8 surrounding points. The center point receives a weight of 1.0, the points at each side and above and below receive a weight of 0.5, and corner points receive a weight of 0.3. All 9 points are multiplied by their weights and summed, then divided by the total weight to obtain the smoothed value. Any missing data points are not included in the sum; points beyond the grid boundary are considered to be missing. Thus the final result may be the result of an averaging with less than 9 points.

2.15.5. REPLACEVALUES - Replace variable values

Synopsis

setvals,oldval,newval[,...] ifile ofile
setrtoc,rmin,rmax,c ifile ofile
setrtoc2,rmin,rmax,c,c2 ifile ofile

Description

This module replaces old variable values with new values, depending on the operator.

Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setvals</td>
<td>Set list of old values to new values</td>
</tr>
<tr>
<td></td>
<td>Supply a list of n pairs of old and new values.</td>
</tr>
<tr>
<td>setrtoc</td>
<td>Set range to constant</td>
</tr>
</tbody>
</table>
|          | \( o(t,x) = \begin{cases} 
  c & \text{if } i(t,x) \geq \text{rmin} \land i(t,x) \leq \text{rmax} \\
  i(t,x) & \text{if } i(t,x) < \text{rmin} \lor i(t,x) > \text{rmax} 
\end{cases} \) |
| setrtoc2 | Set range to constant others to constant2 |
|          | \( o(t,x) = \begin{cases} 
  c & \text{if } i(t,x) \geq \text{rmin} \land i(t,x) \leq \text{rmax} \\
  c2 & \text{if } i(t,x) < \text{rmin} \lor i(t,x) > \text{rmax} 
\end{cases} \) |

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oldval,newval,...</td>
<td>FLOAT</td>
<td>Pairs of old and new values</td>
</tr>
<tr>
<td>rmin</td>
<td>FLOAT</td>
<td>Lower bound</td>
</tr>
<tr>
<td>rmax</td>
<td>FLOAT</td>
<td>Upper bound</td>
</tr>
<tr>
<td>c</td>
<td>FLOAT</td>
<td>New value - inside range</td>
</tr>
<tr>
<td>c2</td>
<td>FLOAT</td>
<td>New value - outside range</td>
</tr>
</tbody>
</table>
2.15.6. TIMSORT - Timsort

Synopsis

timsort ifile ofile

Description

Sorts the elements in ascending order over all timesteps for every field position. After sorting it is:
\[ o(t_1, x) \leq o(t_2, x) \quad \forall (t_1 < t_2), x \]

Example

To sort all field elements of a dataset over all timesteps use:

cdo timsort ifile ofile

2.15.7. VARGEN - Generate a field

Synopsis

\[ \text{const, const, grid ofile} \]
\[ \text{random, grid[, seed] ofile} \]
\[ \text{stdatm, levels ofile} \]

Description

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

**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 distributed random numbers in the interval [0,1].

**stdatm**
Create values for pressure and temperature for hydrostatic atmosphere
Creates pressure and temperature values for the given list of vertical levels. The formulas are:

\[
P(z) = P_0 \exp \left( -\frac{g \, H}{T_0} \log \left( \frac{\exp \left( \frac{H \, T_0}{T_0 + \Delta T} \right)}{T_0 + \Delta T} \right) \right)
\]

\[
T(z) = T_0 + \Delta T \exp \left( -\frac{z}{H} \right)
\]

with the following constants

\[
T_0 = 213 \text{K} : \text{offset to get a surface temperature of 288K}
\]

\[
\Delta T = 75 \text{K} : \text{Temperature lapse rate for 10Km}
\]

\[
P_0 = 1013.25 \text{hPa} : \text{surface pressure}
\]

\[
H = 10000.0 \text{m} : \text{scale height}
\]

\[
g = 9.80665 \frac{\text{m}}{\text{s}^2} : \text{earth gravity}
\]

\[
R = 287.05 \frac{\text{J}}{\text{kgK}} : \text{gas constant for air}
\]

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const</td>
<td>FLOAT</td>
<td>Constant</td>
</tr>
<tr>
<td>seed</td>
<td>INTEGER</td>
<td>The seed for a new sequence of pseudo-random numbers [default: 1]</td>
</tr>
<tr>
<td>grid</td>
<td>STRING</td>
<td>Target grid description file or name</td>
</tr>
<tr>
<td>levels</td>
<td>FLOAT</td>
<td>Target levels in metre above surface</td>
</tr>
</tbody>
</table>

Example

To create a standard atmosphere dataset on a given horizontal grid:

```
cdo enlarge,gridfile -stdatm,10000,8000,5000,3000,2000,1000,500,200,0 ofile
```
2.15.8. ROTUV - Rotation

Synopsis

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

Description

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

Parameter

`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
```

2.15.9. MASTRFU - Mass stream function

Synopsis

`mastrfu ifile ofile`

Description

This is a special operator for the post processing of the atmospheric general circulation model ECHAM. It computes the mass stream function (code number 272). The input dataset have to be a zonal mean of v-velocity [m/s] (code number 132) on pressure levels.

Example

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

```
cdo mastrfu ifile ofile
```
2.15.10. HISTOGRAM - Histogram

Synopsis

<operator>,bounds ifile ofile

Description

This module creates bins for a histogram of the input data. The bins have to be adjacent and have non-overlapping intervals. The user has to define the bounds of the bins. The first value is the lower bound and the second value the upper bound of the first bin. The bounds of the second bin are defined by the second and third value, aso. Only 2-dimensional input fields are allowed. The output file contains one vertical level for each of the bins requested.

Operators

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)

2.15.11. SETHALO - Set the left and right bounds of a field

Synopsis

sethalo,lhalo,rhalo ifile ofile

Description

This operator sets the left and right bounds of the rectangularly understood fields. Positive numbers of the parameter lhalo enlarges the left bound by the given number of columns from the right bound. The parameter rhalo does the similar for the right bound. Negative numbers of the parameter lhalo/rhalo can be used to remove the given number of columns of the left and right bounds.

Parameter

lhalo INTEGER Left halo
rhalo INTEGER Right halo
2.15.12. WCT - Windchill temperature

Synopsis

\texttt{wct ifile1 ifile2 ofile}

Description

Let \texttt{ifile1} and \texttt{ifile2} be time series of temperature and wind speed records, then a corresponding time series of resulting windchill temperatures is written to \texttt{ofile}. The wind chill temperature calculation is only valid for a temperature of $T < 33^\circ C$ and a wind speed of $v > 1.39 \text{ m/s}$. Whenever these conditions are not satisfied, a missing value is written to \texttt{ofile}. Note that temperature and wind speed records have to be given in units of $^\circ C$ and m/s, respectively.

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

Synopsis

\texttt{fdns ifile1 ifile2 ofile}

Description

Let \texttt{ifile1} be a time series of the daily minimum temperature $T_N$ and \texttt{ifile2} be a corresponding series of daily surface snow amounts. Then the number of days where $T_N < 0^\circ C$ and the surface snow amount is less than 1 cm is counted. The temperature $T_N$ have to be given in units of Kelvin. The date information of a timestep in \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile}.

2.15.14. STRWIN - Strong wind days index per time period

Synopsis

\texttt{strwin[\textit{v}] ifile ofile}

Description

Let \texttt{ifile} be a time series of the daily maximum horizontal wind speed $V_X$, then the number of days where $V_X > v$ is counted. The horizontal wind speed $v$ is an optional parameter with default $v = 10.5 \text{ m/s}$. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to $v$. Note that both $V_X$ 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 \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile}.

Parameter

\begin{itemize}
  \item $v$ FLOAT Horizontal wind speed threshold (m/s, default $v = 10.5 \text{ m/s}$)
\end{itemize}
2.15.15. STRBRE - Strong breeze days index per time period

Synopsis

\texttt{strbre ifile ofile}

Description

Let \texttt{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 \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile}.

2.15.16. STRGAL - Strong gale days index per time period

Synopsis

\texttt{strgal ifile ofile}

Description

Let \texttt{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 \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile}.

2.15.17. HURR - Hurricane days index per time period

Synopsis

\texttt{hurr ifile ofile}

Description

Let \texttt{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 \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile}.
2.16. **Climate indices**

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

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

- **eca_cdd**: Consecutive dry days index per time period
- **eca_cfd**: Consecutive frost days index per time period
- **eca_csu**: Consecutive summer days index per time period
- **eca_cwd**: Consecutive wet days index per time period
- **eca_cwdi**: Cold wave duration index w.r.t. mean of reference period
- **eca_cwfi**: Cold-spell days index w.r.t. 10th percentile of reference period
- **eca_etru**: Intra-period extreme temperature range
- **eca_fd**: Frost days index per time period
- **eca_gsl**: Growing season length index
- **eca_hd**: Heating degree days per time period
- **eca_hwdi**: Heat wave duration index w.r.t. mean of reference period
- **eca_hwfi**: Warm spell days index w.r.t. 90th percentile of reference period
- **eca_id**: Ice days index per time period
- **eca_r75p**: Moderate wet days w.r.t. 75th percentile of reference period
- **eca_r75ptot**: Precipitation percent due to R75p days
- **eca_r90p**: Wet days w.r.t. 90th percentile of reference period
- **eca_r90ptot**: Precipitation percent due to R90p days
- **eca_r95p**: Very wet days w.r.t. 95th percentile of reference period
- **eca_r95ptot**: Precipitation percent due to R95p days
- **eca_r99p**: Extremely wet days w.r.t. 99th percentile of reference period
- **eca_r99ptot**: Precipitation percent due to R99p days
- **eca_pd**: Precipitation days index per time period
- **eca_r10mm**: Heavy precipitation days index per time period
- **eca_r20mm**: Very heavy precipitation days index per time period
- **eca_rx1**: Wet days index per time period
- **eca_rx1day**: Highest one day precipitation amount per time period
- **eca_rx5day**: Highest five-day precipitation amount per time period
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eca_sdii</td>
<td>Simple daily intensity index per time period</td>
</tr>
<tr>
<td>eca_su</td>
<td>Summer days index per time period</td>
</tr>
<tr>
<td>eca_tg10p</td>
<td>Cold days percent w.r.t. 10th percentile of reference period</td>
</tr>
<tr>
<td>eca_tg90p</td>
<td>Warm days percent w.r.t. 90th percentile of reference period</td>
</tr>
<tr>
<td>eca_tn10p</td>
<td>Cold nights percent w.r.t. 10th percentile of reference period</td>
</tr>
<tr>
<td>eca_tn90p</td>
<td>Warm nights percent w.r.t. 90th percentile of reference period</td>
</tr>
<tr>
<td>eca_tr</td>
<td>Tropical nights index per time period</td>
</tr>
<tr>
<td>eca_tx10p</td>
<td>Very cold days percent w.r.t. 10th percentile of reference period</td>
</tr>
<tr>
<td>eca_tx90p</td>
<td>Very warm days percent w.r.t. 90th percentile of reference period</td>
</tr>
</tbody>
</table>
2.16.1. ECACDD - Consecutive dry days index per time period

Synopsis

\texttt{eca\_cdd}[/R] ifile ofile

Description

Let \texttt{ifile} be a time series of the daily precipitation amount RR, then the largest number of consecutive days where RR is less than \( R \) is counted. \( R \) is an optional parameter with default \( R = 1 \) mm. A further output variable is the number of dry periods of more than 5 days. The date information of a timestep in \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile}. The following variables are created:

- consecutive\_dry\_days\_index\_per\_time\_period
- number\_of\_cdd\_periods\_with\_more\_than\_5\_days\_per\_time\_period

Parameter

\[ R \quad \text{FLOAT} \quad \text{Precipitation threshold (mm, default: } R = 1 \text{ mm)} \]

Example

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

\texttt{cdo eca\_cdd rrfile ofile}

2.16.2. ECACFD - Consecutive frost days index per time period

Synopsis

\texttt{eca\_cfd} ifile ofile

Description

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

- consecutive\_frost\_days\_index\_per\_time\_period

Example

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

\texttt{cdo eca\_cfd tnfile ofile}
2.16.3. ECACSU - Consecutive summer days index per time period

Synopsis

\texttt{eca\_csu}, [\textit{T}] \textit{ifile ofile}

Description

Let \textit{ifile} be a time series of the daily maximum temperature \textit{TX}, then the largest number of consecutive days where \textit{TX} \textgreater{} \textit{T} is counted. The number \textit{T} is an optional parameter with default \textit{T} = 25 \textdegree{} Celsius. Note that \textit{TN} have to be given in units of Kelvin, whereas \textit{T} have to be given in degrees Celsius. The date information of a timestep in \textit{ofile} is the date of the last contributing timestep in \textit{ifile}. The following variables are created:

- \texttt{consecutive\_summer\_days\_index\_per\_time\_period}

Parameter

\texttt{T FLOAT} \hspace{1cm} \text{Temperature threshold (Celsius, default: T = 25 Celsius)}

Example

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

\begin{verbatim}
cdo eca\_csu txfile ofile
\end{verbatim}

2.16.4. ECACWD - Consecutive wet days index per time period

Synopsis

\texttt{eca\_cwd}, [\textit{R}] \textit{ifile ofile}

Description

Let \textit{ifile} be a time series of the daily precipitation amount \textit{RR}, then the largest number of consecutive days where \textit{RR} is at least \textit{R} is counted. \textit{R} is an optional parameter with default \textit{R} = 1 mm. A further output variable is the number of wet periods of more than 5 days. The date information of a timestep in \textit{ofile} is the date of the last contributing timestep in \textit{ifile}. The following variables are created:

- \texttt{consecutive\_wet\_days\_index\_per\_time\_period}
- \texttt{number\_of\_cwd\_periods\_with\_more\_than\_5\_days\_per\_time\_period}

Parameter

\texttt{R FLOAT} \hspace{1cm} \text{Precipitation threshold (mm, default: R = 1 mm)}
Example

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

```
cdo eca_cwd rrf ile of ile
```
2.16.5. ECACWDI - Cold wave duration index w.r.t. mean of reference period

Synopsis

\[ \text{eca}_{\text{cwdi}}[n\text{day},T] \text{ ifile1 ifile2 ofile} \]

Description

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

- cold_wave_duration_index_wrt_mean_of_reference_period
- cold_waves_per_time_period

Parameter

- \( n\text{day} \) INTEGER Number of consecutive days (default: \( n\text{day} = 6 \))
- \( T \) FLOAT Temperature offset (Celsius, default: \( T = 5 \) Celsius)

Example

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

\[ \text{cdo eca}_{\text{cwdi}} tnf_{ile} tnnorm_{ile} o_{ile} \]

2.16.6. ECACWFI - Cold-spell days index w.r.t. 10th percentile of reference period

Synopsis

\[ \text{eca}_{\text{cwfi}}[n\text{day}] \text{ ifile1 ifile2 ofile} \]

Description

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

- cold_spell_days_index_wrt_10th_percentile_of_reference_period
- cold_spell_periods_per_time_period
Parameter

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

Example

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

```
cdo eca_cwfi tgfle tgn10file ofile
```
2.16.7. ECAETR - Intra-period extreme temperature range

Synopsis

eca_etra ifile1 ifile2 ofile

Description

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

- intra_period_extreme_temperature_range

Example

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

cdo eca_etra txfile tnfile ofile

2.16.8. ECAFD - Frost days index per time period

Synopsis

eca_fd ifile ofile

Description

Let ifile be a time series of the daily minimum temperature TN, then the number of days where TN < 0 °C is counted. Note that 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. The following variables are created:

- frost_days_index_per_time_period

Example

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

cdo eca_fd tnfile ofile
2.16.9. ECAGSL - Thermal Growing season length index

Synopsis

\texttt{eca\_gsl[\textit{nday},\textit{T},\textit{fland}]] ifile1 ifile2 ofile}

Description

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

- first occurrence of at least \textit{nday} consecutive days with \textit{TG} \textgreater@math{\textit{T}}
- first occurrence of at least \textit{nday} consecutive days with \textit{TG} \textless@math{\textit{T}} within the last 6 months

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

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

The date information of a timestep in \textit{ofile} is the date of the last contributing timestep in \textit{ifile}. The following variables are created:

- \textit{thermal\_growing\_season\_length}
- \textit{day\_of\_year\_of\_growing\_season\_start}

Parameter

\begin{itemize}
  \item \textit{nday} INTEGER Number of consecutive days (default: \textit{nday} = 6)
  \item \textit{T} FLOAT Temperature threshold (degree Celsius, default: \textit{T} = 5 Celsius)
  \item \textit{fland} FLOAT Land fraction threshold (default: \textit{fland} = 0.5)
\end{itemize}

Example

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

\texttt{cdo eca\_gsl tgf\_file mask\_file of\_file}
2.16.10. ECAHD - Heating degree days per time period

**Synopsis**

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

**Description**

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

- `heating_degree_days_per_time_period`

**Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>T1</code></td>
<td>FLOAT</td>
<td>Temperature limit (Celsius, default: <code>T1 = 17 Celsius</code>)</td>
</tr>
<tr>
<td><code>T2</code></td>
<td>FLOAT</td>
<td>Temperature limit (Celsius, default: <code>T2 = T1</code>)</td>
</tr>
</tbody>
</table>

**Example**

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

```
cdo eca_hd tgfile ofile
```

2.16.11. ECAHWDI - Heat wave duration index w.r.t. mean of reference period

**Synopsis**

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

**Description**

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

- `heat_wave_duration_index_wrt_mean_of_reference_period`
- `heat_waves_per_time_period`
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{nday}$</td>
<td>INTEGER</td>
<td>Number of consecutive days (default: $\text{nday} = 6$)</td>
</tr>
<tr>
<td>$T$</td>
<td>FLOAT</td>
<td>Temperature offset (Celsius, default: $T = 5$ Celsius)</td>
</tr>
</tbody>
</table>
2.16.12. ECAHWFI - Warm spell days index w.r.t. 90th percentile of reference period

Synopsis

`eca_hwfi[nday] ifile1 ifile2 ofile`

Description

Let `ifile1` be a time series of the daily mean temperature $T_G$, and `ifile2` be the 90th percentile $T_Gn90$ of daily mean temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least $nday$ consecutive days, $T_G > T_Gn90$. The number $nday$ is an optional parameter with default $nday = 6$. A further output variable is the number of warm-spell periods longer than or equal to $nday$ days. $T_Gn90$ is calculated as the 90th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both $T_G$ and $T_Gn90$ have to be given in the same units. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile1`. The following variables are created:

- `warm_spell_days_index_wrt_90th_percentile_of_reference_period`
- `warm_spell_periods_per_time_period`

Parameter

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

Example

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

```
cdo eca_hwfi tgfile tgn90file ofile
```

2.16.13. ECAID - Ice days index per time period

Synopsis

`eca_id ifile ofile`

Description

Let `ifile` be a time series of the daily maximum temperature $T_X$, then the number of days where $T_X < 0$ °C is counted. Note that $T_X$ 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`. The following variables are created:

- `ice_days_index_per_time_period`

Example

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

```
cdo eca_id txfile ofile
```
2.16.14. ECAR75P - Moderate wet days w.r.t. 75th percentile of reference period

Synopsis

`eca_r75p ifile1 ifile2 ofile`

Description

Let `ifile1` be a time series RR of the daily precipitation amount at wet days (precipitation $\geq 1$ mm) and `ifile2` be the 75th percentile RRn75 of the daily precipitation amount at wet days for any period used as reference. Then the percentage of wet days with RR $> RRn75$ is calculated. RRn75 is calculated as the 75th percentile of all wet days of a given climate reference period. Usually `ifile2` is generated by the operator `ydaypctl,75`. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile1`. The following variables are created:

- moderate_wet_days_wrt_75th_percentile_of_reference_period

Example

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

```
cdo eca_r75p rrfile rrn75file ofile
```

2.16.15. ECAR75PTOT - Precipitation percent due to R75p days

Synopsis

`eca_r75ptot ifile1 ifile2 ofile`

Description

Let `ifile1` be a time series RR of the daily precipitation amount at wet days (precipitation $\geq 1$ mm) and `ifile2` be the 75th percentile RRn75 of the daily precipitation amount at wet days for any period used as reference. Then the ratio of the precipitation sum at wet days with RR $> RRn75$ to the total precipitation sum is calculated. RRn75 is calculated as the 75th percentile of all wet days of a given climate reference period. Usually `ifile2` is generated by the operator `ydaypctl,75`. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile1`. The following variables are created:

- precipitation_percent_due_to_R75p_days
2.16.16. ECAR90P - Wet days w.r.t. 90th percentile of reference period

Synopsis

eca_r90p ifile1 ifile2 ofile

Description

Let ifile1 be a time series RR of the daily precipitation amount at wet days (precipitation $\geq 1$ mm) and ifile2 be the 90th percentile RRn90 of the daily precipitation amount at wet days for any period used as reference. Then the percentage of wet days with RR > RRn90 is calculated. RRn90 is calculated as the 90th percentile of all wet days of a given climate reference period. Usually ifile2 is generated by the operator ydaypctl,90. The date information of a timestep in ofile is the date of the last contributing timestep in ifile1. The following variables are created:

- wet_days_wrt_90th_percentile_of_reference_period

Example

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

```cdo eca_r90p rrfile rrn90file ofile```

2.16.17. ECAR90PTOT - Precipitation percent due to R90p days

Synopsis

eca_r90ptot ifile1 ifile2 ofile

Description

Let ifile1 be a time series RR of the daily precipitation amount at wet days (precipitation $\geq 1$ mm) and ifile2 be the 90th percentile RRn90 of the daily precipitation amount at wet days for any period used as reference. Then the ratio of the precipitation sum at wet days with RR > RRn90 to the total precipitation sum is calculated. RRn90 is calculated as the 90th percentile of all wet days of a given climate reference period. Usually ifile2 is generated by the operator ydaypctl,90. The date information of a timestep in ofile is the date of the last contributing timestep in ifile1. The following variables are created:

- precipitation_percent_due_to_R90p_days
2.16.18. ECAR95P - Very wet days w.r.t. 95th percentile of reference period

Synopsis

\texttt{eca\_r95p ifile1 ifile2 ofile}

Description

Let \texttt{ifile1} be a time series RR of the daily precipitation amount at wet days (precipitation \(\geq 1\) mm) and \texttt{ifile2} be the 95th percentile RRn95 of the daily precipitation amount at wet days for any period used as reference. Then the percentage of wet days with RR > RRn95 is calculated. RRn95 is calculated as the 95th percentile of all wet days of a given climate reference period. Usually \texttt{ifile2} is generated by the operator \texttt{ydaypctl,95}. The date information of a timestep in \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile1}. The following variables are created:

- very\_wet\_days\_wrt\_95th\_percentile\_of\_reference\_period

Example

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

\texttt{cdo eca\_r95p rrfile rrn95file ofile}

2.16.19. ECAR95PTOT - Precipitation percent due to R95p days

Synopsis

\texttt{eca\_r95ptot ifile1 ifile2 ofile}

Description

Let \texttt{ifile1} be a time series RR of the daily precipitation amount at wet days (precipitation \(\geq 1\) mm) and \texttt{ifile2} be the 95th percentile RRn95 of the daily precipitation amount at wet days for any period used as reference. Then the ratio of the precipitation sum at wet days with RR > RRn95 to the total precipitation sum is calculated. RRn95 is calculated as the 95th percentile of all wet days of a given climate reference period. Usually \texttt{ifile2} is generated by the operator \texttt{ydaypctl,95}. The date information of a timestep in \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile1}. The following variables are created:

- precipitation\_percent\_due\_to\_R95p\_days
2.16.20. ECAR99P - Extremely wet days w.r.t. 99th percentile of reference period

Synopsis

```plaintext
txt  eca_r99p  ifile1  ifile2  ofile
```

Description

Let `ifile1` be a time series RR of the daily precipitation amount at wet days (precipitation $\geq 1$ mm) and `ifile2` be the 99th percentile RRn99 of the daily precipitation amount at wet days for any period used as reference. Then the percentage of wet days with RR $\geq$ RRn99 is calculated. RRn99 is calculated as the 99th percentile of all wet days of a given climate reference period. Usually `ifile2` is generated by the operator `ydaypctl,99`. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile1`. The following variables are created:

- `extremely_wet_days_wrt_99th_percentile_of_reference_period`

Example

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

```plaintext
txt  cdo  eca_r99p  rrfile  rrn99file  ofile
```

2.16.21. ECAR99PTOT - Precipitation percent due to R99p days

Synopsis

```plaintext
txt  eca_r99ptot  ifile1  ifile2  ofile
```

Description

Let `ifile1` be a time series RR of the daily precipitation amount at wet days (precipitation $\geq 1$ mm) and `ifile2` be the 99th percentile RRn99 of the daily precipitation amount at wet days for any period used as reference. Then the ratio of the precipitation sum at wet days with RR $\geq$ RRn99 to the total precipitation sum is calculated. RRn99 is calculated as the 99th percentile of all wet days of a given climate reference period. Usually `ifile2` is generated by the operator `ydaypctl,99`. The date information of a timestep in `ofile` is the date of the last contributing timestep in `ifile1`. The following variables are created:

- `precipitation_percent_due_to_R99p_days`
2.16.22. ECAPD - Precipitation days index per time period

Synopsis

\[
\text{eca}_{\text{pd}}, \text{r}10\text{mm}, \text{r}20\text{mm} \text{ ifile ofile}
\]

Description

Let ifile be a time series of the daily precipitation amount RR in [mm] (or alternatively in [kg m^{-2}]), then the number of days where RR is at least \( x \) mm is counted. eca_{r}10mm and eca_{r}20mm are specific ECA operators with a daily precipitation amount of 10 and 20 mm respectively. The date information of a timestep in ofile is the date of the last contributing timestep in ifile. The following variables are created:

- precipitation_days_index_per_time_period

Operators

\[
\begin{align*}
\text{eca}_{\text{pd}} & \quad \text{Precipitation days index per time period} \\
\text{eca}_{\text{r}}10\text{mm} & \quad \text{Heavy precipitation days index per time period} \\
\text{eca}_{\text{r}}20\text{mm} & \quad \text{Very heavy precipitation days index per time period}
\end{align*}
\]

Generic ECA operator with daily precipitation sum exceeding \( x \) mm.
Specific ECA operator with daily precipitation sum exceeding 10 mm.
Specific ECA operator with daily precipitation sum exceeding 20 mm.

Parameter

\[
x \quad \text{FLOAT} \quad \text{Daily precipitation amount threshold in [mm]}
\]

Note

Precipitation rates in [mm/s] have to be converted to precipitation amounts (multiply with 86400 s). Apart from metadata information the result of eca_{pd,1} and eca_{r}1 is the same.

Example

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

\[
\text{cdo eca}_{\text{pd}},25 \text{ ifile ofile}
\]
2.16.23. ECARR1 - Wet days index per time period

Synopsis

eca_r1[R] ifile ofile

Description

Let ifile be a time series of the daily precipitation amount RR in [mm] (or alternatively in [kg m-2]), then the number of days where RR is at least \( R \) is counted. \( R \) is an optional parameter with default \( R = 1 \) mm. The date information of a timestep in ofile is the date of the last contributing timestep in ifile. The following variables are created:

- wet_days_index_per_time_period

Parameter

\( R \) FLOAT Precipitation threshold (mm, default: \( R = 1 \) mm)

Example

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

cdo eca_r1 rrfile ofile

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

Synopsis

eca_rx1day[mode] ifile ofile

Description

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

- highest_one_day_precipitation_amount_per_time_period

Parameter

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

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

```
cdo eca_rx1day rrfile ofile
```

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

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

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

```
cdo timmax rrfile ofile
```
```
cdo monmax rrfile ofile
```
2.16.25. ECARX5DAY - Highest five-day precipitation amount per time period

Synopsis

\texttt{eca\_rx5day[,x] ifile ofile}

Description

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

- \texttt{highest\_five\_day\_precipitation\_amount\_per\_time\_period}
- \texttt{number\_of\_5day\_heavy\_precipitation\_periods\_per\_time\_period}

Parameter

\(x\)  \texttt{FLOAT} \quad \text{Precipitation threshold (mm, default: } x = 50\text{ mm)}

Example

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

\begin{verbatim}
cdo eca\_rx5day rrfile ofile
\end{verbatim}

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

\begin{verbatim}
cdo timmax rrfile ofile
\end{verbatim}

2.16.26. ECASDII - Simple daily intensity index per time period

Synopsis

\texttt{eca\_sdii[,R] ifile ofile}

Description

Let \texttt{ifile} be a time series of the daily precipitation amount RR, then the mean precipitation amount at wet days (\(RR > R\)) is written to \texttt{ofile}. \(R\) is an optional parameter with default \(R = 1\) mm. The date information of a timestep in \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile}. The following variables are created:

- \texttt{simple\_daily\_intensity\_index\_per\_time\_period}

Parameter

\(R\)  \texttt{FLOAT} \quad \text{Precipitation threshold (mm, default: } R = 1\text{ mm)}
Example

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

\[
\text{cdo eca_sdii rrfile ofile}
\]

2.16.27. ECASU - Summer days index per time period

Synopsis

\[
\text{eca_su}[T] \text{ ifile ofile}
\]

Description

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

- summer_days_index_per_time_period

Parameter

\[
T \quad \text{FLOAT} \quad \text{Temperature threshold (degree Celsius, default: } T = 25 \text{ Celsius)}
\]

Example

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

\[
\text{cdo eca_su txfile ofile}
\]
2.16.28. ECATG10P - Cold days percent w.r.t. 10th percentile of reference period

Synopsis

cda_tg10p ifile1 ifile2 ofile

Description

Let ifile1 be a time series of the daily mean temperature TG, and ifile2 be the 10th percentile TGN10 of daily mean temperatures for any period used as reference. Then the percentage of time where TG < TGN10 is calculated. TGN10 is calculated as the 10th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TG and TGN10 have to be given in the same units. The date information of a timestep in ofile is the date of the last contributing timestep in ifile1. The following variables are created:

- cold_days_percent_wrt_10th_percentile_of_reference_period

Example

To compute the percentage of timesteps with a daily mean temperature smaller than the 10th percentile of the daily mean temperatures for a given reference period use:

cdo eca_tg10p tgfile tgn10file ofile

2.16.29. ECATG90P - Warm days percent w.r.t. 90th percentile of reference period

Synopsis

cda_tg90p ifile1 ifile2 ofile

Description

Let ifile1 be a time series of the daily mean temperature TG, and ifile2 be the 90th percentile TGN90 of daily mean temperatures for any period used as reference. Then the percentage of time where TG > TGN90 is calculated. TGN90 is calculated as the 90th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TG and TGN90 have to be given in the same units. The date information of a timestep in ofile is the date of the last contributing timestep in ifile1. The following variables are created:

- warm_days_percent_wrt_90th_percentile_of_reference_period

Example

To compute the percentage of timesteps with a daily mean temperature greater than the 90th percentile of the daily mean temperatures for a given reference period use:

cdo eca_tg90p tgfile tgn90file ofile
2.16.30. ECATN10P - Cold nights percent w.r.t. 10th percentile of reference period

Synopsis

eca_tn10p ifile1 ifile2 ofile

Description

Let ifile1 be a time serie of the daily minimum temperature TN, and ifile2 be the 10th percentile TNN10 of daily minimum temperatures for any period used as reference. Then the percentage of time where TN < TNN10 is calculated. TNN10 is calculated as the 10th percentile of daily minimum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TN and TNN10 have to be given in the same units. The date information of a timestep in ofile is the date of the last contributing timestep in ifile1. The following variables are created:

- cold_nights_percent_wrt_10th_percentile_of_reference_period

Example

To compute the percentage of timesteps with a daily minimum temperature smaller than the 10th percentile of the daily minimum temperatures for a given reference period use:

```
cdo eca_tn10p tnfile tnn10file ofile
```

2.16.31. ECATN90P - Warm nights percent w.r.t. 90th percentile of reference period

Synopsis

eca_tn90p ifile1 ifile2 ofile

Description

Let ifile1 be a time series of the daily minimum temperature TN, and ifile2 be the 90th percentile TNN90 of daily minimum temperatures for any period used as reference. Then the percentage of time where TN > TNN90 is calculated. TNN90 is calculated as the 90th percentile of daily minimum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TN and TNN90 have to be given in the same units. The date information of a timestep in ofile is the date of the last contributing timestep in ifile1. The following variables are created:

- warm_nights_percent_wrt_90th_percentile_of_reference_period

Example

To compute the percentage of timesteps with a daily minimum temperature greater than the 90th percentile of the daily minimum temperatures for a given reference period use:

```
cdo eca_tn90p tnfile tnn90file ofile
```
2.16.32. ECATR - Tropical nights index per time period

Synopsis

\texttt{eca_tr[,T] ifile ofile}

Description

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

- tropical_nights_index_per_time_period

Parameter

\textbf{T} \hspace{1cm} \textsc{FLOAT} \hspace{1cm} Temperature threshold (Celsius, default: T = 20 Celsius)

Example

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

\texttt{cdo eca_tr tnfile ofile}

2.16.33. ECATX10P - Very cold days percent w.r.t. 10th percentile of reference period

Synopsis

\texttt{eca_tx10p ifile1 ifile2 ofile}

Description

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

- very_cold_days_percent_wrt_10th_percentile_of_reference_period

Example

To compute the percentage of timesteps with a daily maximum temperature smaller than the 10th percentile of the daily maximum temperatures for a given reference period use:

\texttt{cdo eca_tx10p txfile txn10file ofile}
2.16.34. **ECATX90P - Very warm days percent w.r.t. 90th percentile of reference period**

**Synopsis**

\texttt{eca\_tx90p ifile1 ifile2 ofile}

**Description**

Let \texttt{ifile1} be a time series of the daily maximum temperature \texttt{TX}, and \texttt{ifile2} be the 90th percentile \texttt{TXn90} of daily maximum temperatures for any period used as reference. Then the percentage of time where \texttt{TX} > \texttt{TXn90} is calculated. \texttt{TXn90} is calculated as the 90th percentile of daily maximum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both \texttt{TX} and \texttt{TXn90} have to be given in the same units. The date information of a timestep in \texttt{ofile} is the date of the last contributing timestep in \texttt{ifile1}. The following variables are created:

- \texttt{very\_warm\_days\_percent\_wrt\_90th\_percentile\_of\_reference\_period}

**Example**

To compute the percentage of timesteps with a daily maximum temperature greater than the 90th percentile of the daily maximum temperatures for a given reference period use:

\texttt{cdo eca\_tx90p txfile txn90file ofile}
Bibliography

[CDI] Climate Data Interface, from the Max Planck Institute for Meteorologie

[CM-SAF] Satellite Application Facility on Climate Monitoring, from the German Weather Service (Deutscher Wetterdienst, DWD)

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


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

[GRIBAPI] GRIB API decoding/encoding, from the European Centre for Medium-Range Weather Forecasts (ECMWF)

[HDF5] HDF version 5, from the HDF Group

[INTERA] INTERA Software Package, from the Max Planck Institute for Meteorologie

[MPIOM] Ocean and sea ice model, from the Max Planck Institute for Meteorologie

[netCDF] NetCDF Software Package, from the UNIDATA Program Center of the University Corporation for Atmospheric Research

[PINGO] The PINGO package, from the Model & Data group at the Max Planck Institute for Meteorologie

[REMO] Regional Model, from the Max Planck Institute for Meteorologie


[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.

A. Grid description examples

A.1. Example of a curvilinear grid description

Here is an example for the CDO description of a curvilinear grid. xvals/yvals 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
        9 27 35 13 27 41 51 35 41 52 63 51
        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 A.1.: Orthographic and Robinson projection of the curvilinear grid, the first grid cell is colored red
A.2. Example description for an unstructured grid

Here is an example of the CDO description for an unstructured grid. xvals/yvals 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 72 165 195 216 216 144 144 144 162 195 216 216 144 144 144 162 195 216 216 144 144 144 162 195 216 216 144 144 144 162 195 216 216 144 144 144

yvals = 58 58 32 0 0 58 32 0 0 58 32 0 0 58 32

ybounds = 41 53 71 71 53 41 41 41 53 71 71 53 41 41 41 53 71 71 53
           11 19 41 41 53 41 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
```

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>66</td>
</tr>
<tr>
<td>acos</td>
<td>66</td>
</tr>
<tr>
<td>add</td>
<td>68</td>
</tr>
<tr>
<td>addc</td>
<td>67</td>
</tr>
<tr>
<td>asin</td>
<td>66</td>
</tr>
<tr>
<td>atan2</td>
<td>68</td>
</tr>
<tr>
<td>bandpass</td>
<td>155</td>
</tr>
<tr>
<td>cat</td>
<td>25</td>
</tr>
<tr>
<td>chcode</td>
<td>50</td>
</tr>
<tr>
<td>chlevel</td>
<td>50</td>
</tr>
<tr>
<td>chleveL</td>
<td>50</td>
</tr>
<tr>
<td>chname</td>
<td>50</td>
</tr>
<tr>
<td>chparam</td>
<td>50</td>
</tr>
<tr>
<td>chunit</td>
<td>50</td>
</tr>
<tr>
<td>consecsum</td>
<td>79</td>
</tr>
<tr>
<td>consects</td>
<td>79</td>
</tr>
<tr>
<td>const</td>
<td>158</td>
</tr>
<tr>
<td>copy</td>
<td>25</td>
</tr>
<tr>
<td>cos</td>
<td>66</td>
</tr>
<tr>
<td>dayavg</td>
<td>101</td>
</tr>
<tr>
<td>daymax</td>
<td>101</td>
</tr>
<tr>
<td>daymean</td>
<td>101</td>
</tr>
<tr>
<td>daymin</td>
<td>101</td>
</tr>
<tr>
<td>daypctl</td>
<td>102</td>
</tr>
<tr>
<td>daystd</td>
<td>101</td>
</tr>
<tr>
<td>daysum</td>
<td>101</td>
</tr>
<tr>
<td>dayvar</td>
<td>101</td>
</tr>
<tr>
<td>decode</td>
<td>33</td>
</tr>
<tr>
<td>dename</td>
<td>33</td>
</tr>
<tr>
<td>delparam</td>
<td>33</td>
</tr>
<tr>
<td>detrend</td>
<td>123</td>
</tr>
<tr>
<td>diff</td>
<td>20</td>
</tr>
<tr>
<td>diffn</td>
<td>20</td>
</tr>
<tr>
<td>div</td>
<td>68</td>
</tr>
<tr>
<td>divc</td>
<td>67</td>
</tr>
<tr>
<td>divdpm</td>
<td>73</td>
</tr>
<tr>
<td>divdpy</td>
<td>73</td>
</tr>
<tr>
<td>dv2ps</td>
<td>145</td>
</tr>
<tr>
<td>dv2uv</td>
<td>145</td>
</tr>
<tr>
<td>dv2v1</td>
<td>145</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>eca_cdd</td>
<td>166</td>
</tr>
<tr>
<td>eca_cfd</td>
<td>166</td>
</tr>
<tr>
<td>eca_csu</td>
<td>167</td>
</tr>
<tr>
<td>eca_cwd</td>
<td>167</td>
</tr>
<tr>
<td>eca_cwdi</td>
<td>169</td>
</tr>
<tr>
<td>eca_cwfl</td>
<td>169</td>
</tr>
<tr>
<td>eca_et</td>
<td>171</td>
</tr>
<tr>
<td>eca_fd</td>
<td>171</td>
</tr>
<tr>
<td>eca_gsl</td>
<td>172</td>
</tr>
<tr>
<td>eca_hdi</td>
<td>173</td>
</tr>
<tr>
<td>eca_lwdi</td>
<td>173</td>
</tr>
<tr>
<td>eca_lwfi</td>
<td>175</td>
</tr>
<tr>
<td>eca_jd</td>
<td>175</td>
</tr>
<tr>
<td>eca_pd</td>
<td>180</td>
</tr>
<tr>
<td>eca_r10m</td>
<td>180</td>
</tr>
<tr>
<td>eca_r20mm</td>
<td>180</td>
</tr>
<tr>
<td>eca_r75p</td>
<td>176</td>
</tr>
<tr>
<td>eca_r75pptot</td>
<td>176</td>
</tr>
<tr>
<td>eca_r90p</td>
<td>177</td>
</tr>
<tr>
<td>eca_r90pptot</td>
<td>177</td>
</tr>
<tr>
<td>eca_r95p</td>
<td>178</td>
</tr>
<tr>
<td>eca_r95pptot</td>
<td>178</td>
</tr>
<tr>
<td>eca_r99p</td>
<td>179</td>
</tr>
<tr>
<td>eca_r99pptot</td>
<td>179</td>
</tr>
<tr>
<td>eca_r1</td>
<td>181</td>
</tr>
<tr>
<td>eca_rx1day</td>
<td>181</td>
</tr>
<tr>
<td>eca_rx5day</td>
<td>183</td>
</tr>
<tr>
<td>eca_sdl</td>
<td>183</td>
</tr>
<tr>
<td>eca_su</td>
<td>184</td>
</tr>
<tr>
<td>eca_t1g10p</td>
<td>185</td>
</tr>
<tr>
<td>eca_t1g90p</td>
<td>185</td>
</tr>
<tr>
<td>eca_tn10p</td>
<td>186</td>
</tr>
<tr>
<td>eca_tn90p</td>
<td>186</td>
</tr>
<tr>
<td>eca_tr</td>
<td>187</td>
</tr>
<tr>
<td>eca_tx10p</td>
<td>187</td>
</tr>
<tr>
<td>eca_tx90p</td>
<td>188</td>
</tr>
<tr>
<td>enlarge</td>
<td>59</td>
</tr>
<tr>
<td>ensavg</td>
<td>80</td>
</tr>
<tr>
<td>enshr</td>
<td>83</td>
</tr>
<tr>
<td>enscrps</td>
<td>83</td>
</tr>
<tr>
<td>ensmmax</td>
<td>80</td>
</tr>
<tr>
<td>ensmmean</td>
<td>80</td>
</tr>
<tr>
<td>ensmmin</td>
<td>80</td>
</tr>
<tr>
<td>enspctl</td>
<td>80</td>
</tr>
<tr>
<td>ensrkhistspace</td>
<td>82</td>
</tr>
<tr>
<td>ensrkhisttime</td>
<td>82</td>
</tr>
<tr>
<td>ensroc</td>
<td>82</td>
</tr>
<tr>
<td>enssd</td>
<td>80</td>
</tr>
<tr>
<td>enssum</td>
<td>80</td>
</tr>
<tr>
<td>Operator</td>
<td>Page</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>ensvar</td>
<td>80</td>
</tr>
<tr>
<td>eof</td>
<td>126</td>
</tr>
<tr>
<td>eof3d</td>
<td>126</td>
</tr>
<tr>
<td>eofcof</td>
<td>128</td>
</tr>
<tr>
<td>eofspatial</td>
<td>126</td>
</tr>
<tr>
<td>eoftime</td>
<td>126</td>
</tr>
<tr>
<td>eq</td>
<td>43</td>
</tr>
<tr>
<td>eqc</td>
<td>44</td>
</tr>
<tr>
<td>exp</td>
<td>66</td>
</tr>
<tr>
<td>expr</td>
<td>64</td>
</tr>
<tr>
<td>expf</td>
<td>64</td>
</tr>
<tr>
<td>fchns</td>
<td>162</td>
</tr>
<tr>
<td>fldavg</td>
<td>85</td>
</tr>
<tr>
<td>fldcor</td>
<td>120</td>
</tr>
<tr>
<td>fldcovar</td>
<td>121</td>
</tr>
<tr>
<td>fldmax</td>
<td>85</td>
</tr>
<tr>
<td>fldmean</td>
<td>85</td>
</tr>
<tr>
<td>fldmin</td>
<td>85</td>
</tr>
<tr>
<td>fldptcl</td>
<td>85</td>
</tr>
<tr>
<td>fldstd</td>
<td>85</td>
</tr>
<tr>
<td>fldsum</td>
<td>85</td>
</tr>
<tr>
<td>fldvar</td>
<td>85</td>
</tr>
<tr>
<td>ge</td>
<td>43</td>
</tr>
<tr>
<td>gec</td>
<td>44</td>
</tr>
<tr>
<td>genbic</td>
<td>132</td>
</tr>
<tr>
<td>genbil</td>
<td>132</td>
</tr>
<tr>
<td>gencon</td>
<td>132</td>
</tr>
<tr>
<td>gencon2</td>
<td>132</td>
</tr>
<tr>
<td>gendis</td>
<td>132</td>
</tr>
<tr>
<td>genlat</td>
<td>132</td>
</tr>
<tr>
<td>genmm</td>
<td>132</td>
</tr>
<tr>
<td>gp2sp</td>
<td>143</td>
</tr>
<tr>
<td>gp2spl</td>
<td>143</td>
</tr>
<tr>
<td>gradsdes1</td>
<td>154</td>
</tr>
<tr>
<td>gradsdes2</td>
<td>154</td>
</tr>
<tr>
<td>gridarea</td>
<td>156</td>
</tr>
<tr>
<td>gridboxavg</td>
<td>91</td>
</tr>
<tr>
<td>gridboxmax</td>
<td>91</td>
</tr>
<tr>
<td>gridboxmean</td>
<td>91</td>
</tr>
<tr>
<td>gridboxmin</td>
<td>91</td>
</tr>
<tr>
<td>gridboxstd</td>
<td>91</td>
</tr>
<tr>
<td>gridboxsum</td>
<td>91</td>
</tr>
<tr>
<td>gridboxvar</td>
<td>91</td>
</tr>
<tr>
<td>grids</td>
<td>23</td>
</tr>
<tr>
<td>gridweights</td>
<td>156</td>
</tr>
<tr>
<td>gt</td>
<td>43</td>
</tr>
<tr>
<td>gtc</td>
<td>44</td>
</tr>
<tr>
<td>highpass</td>
<td>155</td>
</tr>
<tr>
<td>histcount</td>
<td>161</td>
</tr>
<tr>
<td>histfreq</td>
<td>161</td>
</tr>
<tr>
<td>histmean</td>
<td>161</td>
</tr>
<tr>
<td>histsum</td>
<td>161</td>
</tr>
<tr>
<td>houravg</td>
<td>99</td>
</tr>
<tr>
<td>hourmax</td>
<td>99</td>
</tr>
<tr>
<td>hourmean</td>
<td>99</td>
</tr>
<tr>
<td>hourmin</td>
<td>99</td>
</tr>
<tr>
<td>hourptcl</td>
<td>100</td>
</tr>
<tr>
<td>hourstd</td>
<td>99</td>
</tr>
<tr>
<td>hourstd</td>
<td>99</td>
</tr>
<tr>
<td>hourvar</td>
<td>99</td>
</tr>
<tr>
<td>hurr</td>
<td>163</td>
</tr>
<tr>
<td>ifnotthen</td>
<td>40</td>
</tr>
<tr>
<td>ifnotthenc</td>
<td>41</td>
</tr>
<tr>
<td>ifthen</td>
<td>40</td>
</tr>
<tr>
<td>ifthenelse</td>
<td>40</td>
</tr>
<tr>
<td>import_amsr</td>
<td>149</td>
</tr>
<tr>
<td>import_binary</td>
<td>147</td>
</tr>
<tr>
<td>import_emsaf</td>
<td>148</td>
</tr>
<tr>
<td>info</td>
<td>18</td>
</tr>
<tr>
<td>infor</td>
<td>18</td>
</tr>
<tr>
<td>input</td>
<td>150</td>
</tr>
<tr>
<td>inputext</td>
<td>150</td>
</tr>
<tr>
<td>inputs</td>
<td>150</td>
</tr>
<tr>
<td>int</td>
<td>66</td>
</tr>
<tr>
<td>intlevel</td>
<td>138</td>
</tr>
<tr>
<td>intlevel3d</td>
<td>139</td>
</tr>
<tr>
<td>intlevelx3d</td>
<td>139</td>
</tr>
<tr>
<td>inttime</td>
<td>140</td>
</tr>
<tr>
<td>inttime</td>
<td>140</td>
</tr>
<tr>
<td>intyear</td>
<td>141</td>
</tr>
<tr>
<td>invertlat</td>
<td>55</td>
</tr>
<tr>
<td>inverterl</td>
<td>55</td>
</tr>
<tr>
<td>le</td>
<td>43</td>
</tr>
<tr>
<td>lec</td>
<td>44</td>
</tr>
<tr>
<td>ln</td>
<td>66</td>
</tr>
<tr>
<td>log10</td>
<td>66</td>
</tr>
<tr>
<td>lowpass</td>
<td>155</td>
</tr>
<tr>
<td>lt</td>
<td>43</td>
</tr>
<tr>
<td>ltc</td>
<td>44</td>
</tr>
<tr>
<td>map</td>
<td>18</td>
</tr>
<tr>
<td>maskindexbox</td>
<td>57</td>
</tr>
<tr>
<td>masklonlatbox</td>
<td>57</td>
</tr>
<tr>
<td>maskregion</td>
<td>56</td>
</tr>
<tr>
<td>mastrfu</td>
<td>160</td>
</tr>
<tr>
<td>max</td>
<td>68</td>
</tr>
<tr>
<td>meravg</td>
<td>89</td>
</tr>
<tr>
<td>merge</td>
<td>27</td>
</tr>
<tr>
<td>mergetime</td>
<td>27</td>
</tr>
<tr>
<td>mermax</td>
<td>89</td>
</tr>
<tr>
<td>mermean</td>
<td>89</td>
</tr>
<tr>
<td>mermin</td>
<td>89</td>
</tr>
<tr>
<td>merptcl</td>
<td>89</td>
</tr>
<tr>
<td>merstd</td>
<td>89</td>
</tr>
</tbody>
</table>

193
<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>mersum</td>
<td>89</td>
</tr>
<tr>
<td>mersum</td>
<td>89</td>
</tr>
<tr>
<td>min</td>
<td>68</td>
</tr>
<tr>
<td>min2hl</td>
<td>137</td>
</tr>
<tr>
<td>min2pl</td>
<td>137</td>
</tr>
<tr>
<td>monadd</td>
<td>69</td>
</tr>
<tr>
<td>monavg</td>
<td>103</td>
</tr>
<tr>
<td>mondiv</td>
<td>69</td>
</tr>
<tr>
<td>monmax</td>
<td>103</td>
</tr>
<tr>
<td>momax</td>
<td>103</td>
</tr>
<tr>
<td>monmean</td>
<td>103</td>
</tr>
<tr>
<td>monmim</td>
<td>103</td>
</tr>
<tr>
<td>monmul</td>
<td>69</td>
</tr>
<tr>
<td>monptd</td>
<td>104</td>
</tr>
<tr>
<td>monstd</td>
<td>103</td>
</tr>
<tr>
<td>monsub</td>
<td>69</td>
</tr>
<tr>
<td>monsum</td>
<td>103</td>
</tr>
<tr>
<td>monvar</td>
<td>103</td>
</tr>
<tr>
<td>mul</td>
<td>68</td>
</tr>
<tr>
<td>mule</td>
<td>67</td>
</tr>
<tr>
<td>mulpm</td>
<td>73</td>
</tr>
<tr>
<td>mulpy</td>
<td>73</td>
</tr>
<tr>
<td>ndate</td>
<td>21</td>
</tr>
<tr>
<td>ne</td>
<td>43</td>
</tr>
<tr>
<td>nec</td>
<td>44</td>
</tr>
<tr>
<td>nint</td>
<td>66</td>
</tr>
<tr>
<td>nlevel</td>
<td>21</td>
</tr>
<tr>
<td>nmon</td>
<td>21</td>
</tr>
<tr>
<td>npar</td>
<td>21</td>
</tr>
<tr>
<td>ntime</td>
<td>21</td>
</tr>
<tr>
<td>nyear</td>
<td>21</td>
</tr>
<tr>
<td>output</td>
<td>151</td>
</tr>
<tr>
<td>outputf</td>
<td>151</td>
</tr>
<tr>
<td>outputint</td>
<td>151</td>
</tr>
<tr>
<td>outputsrv</td>
<td>151</td>
</tr>
<tr>
<td>pardes</td>
<td>23</td>
</tr>
<tr>
<td>pow</td>
<td>66</td>
</tr>
<tr>
<td>random</td>
<td>158</td>
</tr>
<tr>
<td>reci</td>
<td>66</td>
</tr>
<tr>
<td>regres</td>
<td>123</td>
</tr>
<tr>
<td>remap</td>
<td>134</td>
</tr>
<tr>
<td>remapbic</td>
<td>130</td>
</tr>
<tr>
<td>remapbil</td>
<td>130</td>
</tr>
<tr>
<td>remapcon</td>
<td>130</td>
</tr>
<tr>
<td>remapcon2</td>
<td>130</td>
</tr>
<tr>
<td>remapdis</td>
<td>130</td>
</tr>
<tr>
<td>remapeta</td>
<td>135</td>
</tr>
<tr>
<td>remapf</td>
<td>130</td>
</tr>
<tr>
<td>remapm</td>
<td>130</td>
</tr>
<tr>
<td>replace</td>
<td>26</td>
</tr>
<tr>
<td>rotuvb</td>
<td>160</td>
</tr>
<tr>
<td>runavg</td>
<td>95</td>
</tr>
<tr>
<td>runmax</td>
<td>95</td>
</tr>
<tr>
<td>runmean</td>
<td>95</td>
</tr>
<tr>
<td>runmin</td>
<td>95</td>
</tr>
<tr>
<td>runpctl</td>
<td>96</td>
</tr>
<tr>
<td>runstd</td>
<td>95</td>
</tr>
<tr>
<td>runsum</td>
<td>95</td>
</tr>
<tr>
<td>runvar</td>
<td>95</td>
</tr>
<tr>
<td>seasavg</td>
<td>107</td>
</tr>
<tr>
<td>seasmax</td>
<td>107</td>
</tr>
<tr>
<td>seasmean</td>
<td>107</td>
</tr>
<tr>
<td>seasmin</td>
<td>107</td>
</tr>
<tr>
<td>seaspek1</td>
<td>108</td>
</tr>
<tr>
<td>seasstd</td>
<td>107</td>
</tr>
<tr>
<td>seassum</td>
<td>107</td>
</tr>
<tr>
<td>seasvar</td>
<td>107</td>
</tr>
<tr>
<td>selecte</td>
<td>33</td>
</tr>
<tr>
<td>seldate</td>
<td>36</td>
</tr>
<tr>
<td>selday</td>
<td>36</td>
</tr>
<tr>
<td>selgird</td>
<td>33</td>
</tr>
<tr>
<td>selhour</td>
<td>36</td>
</tr>
<tr>
<td>selindexb</td>
<td>38</td>
</tr>
<tr>
<td>sellevel</td>
<td>33</td>
</tr>
<tr>
<td>sellevidx</td>
<td>33</td>
</tr>
<tr>
<td>selonlatbox</td>
<td>38</td>
</tr>
<tr>
<td>seltype</td>
<td>33</td>
</tr>
<tr>
<td>selmon</td>
<td>36</td>
</tr>
<tr>
<td>selname</td>
<td>33</td>
</tr>
<tr>
<td>selparam</td>
<td>33</td>
</tr>
<tr>
<td>selseas</td>
<td>36</td>
</tr>
<tr>
<td>selsmon</td>
<td>36</td>
</tr>
<tr>
<td>selstdname</td>
<td>33</td>
</tr>
<tr>
<td>selstbnum</td>
<td>33</td>
</tr>
<tr>
<td>settime</td>
<td>36</td>
</tr>
<tr>
<td>settimep</td>
<td>36</td>
</tr>
<tr>
<td>setyear</td>
<td>36</td>
</tr>
<tr>
<td>selazix</td>
<td>33</td>
</tr>
<tr>
<td>setcalendar</td>
<td>48</td>
</tr>
<tr>
<td>setctomiss</td>
<td>60</td>
</tr>
<tr>
<td>setdate</td>
<td>48</td>
</tr>
<tr>
<td>setday</td>
<td>48</td>
</tr>
<tr>
<td>setgatt</td>
<td>54</td>
</tr>
<tr>
<td>setgatts</td>
<td>54</td>
</tr>
<tr>
<td>setgrid</td>
<td>52</td>
</tr>
<tr>
<td>setgridarea</td>
<td>52</td>
</tr>
<tr>
<td>setgridtype</td>
<td>52</td>
</tr>
<tr>
<td>sethalo</td>
<td>161</td>
</tr>
<tr>
<td>setlevel</td>
<td>47</td>
</tr>
<tr>
<td>settype</td>
<td>47</td>
</tr>
<tr>
<td>setmisstoc</td>
<td>60</td>
</tr>
<tr>
<td>setmissval</td>
<td>60</td>
</tr>
<tr>
<td>setmon</td>
<td>48</td>
</tr>
</tbody>
</table>

194
<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>setname</td>
<td>47</td>
</tr>
<tr>
<td>setparam</td>
<td>47</td>
</tr>
<tr>
<td>setpartab</td>
<td>47</td>
</tr>
<tr>
<td>setrealtime</td>
<td>48</td>
</tr>
<tr>
<td>settoc</td>
<td>157</td>
</tr>
<tr>
<td>settoc2</td>
<td>157</td>
</tr>
<tr>
<td>settomiss</td>
<td>60</td>
</tr>
<tr>
<td>setaxis</td>
<td>48</td>
</tr>
<tr>
<td>settomiss</td>
<td>48</td>
</tr>
<tr>
<td>setunits</td>
<td>48</td>
</tr>
<tr>
<td>setunit</td>
<td>47</td>
</tr>
<tr>
<td>setvals</td>
<td>157</td>
</tr>
<tr>
<td>setrange</td>
<td>60</td>
</tr>
<tr>
<td>setyear</td>
<td>48</td>
</tr>
<tr>
<td>setzaxis</td>
<td>53</td>
</tr>
<tr>
<td>shifttime</td>
<td>48</td>
</tr>
<tr>
<td>showcode</td>
<td>22</td>
</tr>
<tr>
<td>showdate</td>
<td>22</td>
</tr>
<tr>
<td>showformat</td>
<td>22</td>
</tr>
<tr>
<td>showlevel</td>
<td>22</td>
</tr>
<tr>
<td>showtype</td>
<td>22</td>
</tr>
<tr>
<td>showyear</td>
<td>22</td>
</tr>
<tr>
<td>showmon</td>
<td>22</td>
</tr>
<tr>
<td>showstdname</td>
<td>22</td>
</tr>
<tr>
<td>showtime</td>
<td>22</td>
</tr>
<tr>
<td>showtimestamp</td>
<td>22</td>
</tr>
<tr>
<td>sinfo</td>
<td>19</td>
</tr>
<tr>
<td>smooth9</td>
<td>157</td>
</tr>
<tr>
<td>sp2gp</td>
<td>143</td>
</tr>
<tr>
<td>sp2gpl</td>
<td>143</td>
</tr>
<tr>
<td>sp2sp</td>
<td>143</td>
</tr>
<tr>
<td>splitcode</td>
<td>28</td>
</tr>
<tr>
<td>splitday</td>
<td>30</td>
</tr>
<tr>
<td>splitgrid</td>
<td>28</td>
</tr>
<tr>
<td>splithour</td>
<td>30</td>
</tr>
<tr>
<td>splitlevel</td>
<td>28</td>
</tr>
<tr>
<td>splitmon</td>
<td>30</td>
</tr>
<tr>
<td>splitname</td>
<td>28</td>
</tr>
<tr>
<td>splitparam</td>
<td>28</td>
</tr>
<tr>
<td>splitsea</td>
<td>30</td>
</tr>
<tr>
<td>splitsel</td>
<td>31</td>
</tr>
<tr>
<td>splitthbnum</td>
<td>28</td>
</tr>
<tr>
<td>splityear</td>
<td>30</td>
</tr>
<tr>
<td>splitzaxis</td>
<td>28</td>
</tr>
<tr>
<td>sgr</td>
<td>66</td>
</tr>
<tr>
<td>sqrt</td>
<td>66</td>
</tr>
<tr>
<td>stdcm</td>
<td>158</td>
</tr>
<tr>
<td>strbre</td>
<td>163</td>
</tr>
<tr>
<td>strgal</td>
<td>163</td>
</tr>
<tr>
<td>strwin</td>
<td>162</td>
</tr>
<tr>
<td>sub</td>
<td>68</td>
</tr>
<tr>
<td>subc</td>
<td>67</td>
</tr>
<tr>
<td>subtrend</td>
<td>124</td>
</tr>
</tbody>
</table>

**T**

---

<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ydayadd</td>
<td>71</td>
</tr>
<tr>
<td>ydayavg</td>
<td>110</td>
</tr>
<tr>
<td>ydaydiv</td>
<td>71</td>
</tr>
<tr>
<td>ydaymax</td>
<td>110</td>
</tr>
<tr>
<td>ydaymean</td>
<td>110</td>
</tr>
<tr>
<td>ydaymin</td>
<td>110</td>
</tr>
<tr>
<td>ydavfi</td>
<td>71</td>
</tr>
<tr>
<td>ydaypctl</td>
<td>111</td>
</tr>
<tr>
<td>ydastd</td>
<td>110</td>
</tr>
<tr>
<td>ydaysub</td>
<td>71</td>
</tr>
<tr>
<td>ydaysum</td>
<td>110</td>
</tr>
<tr>
<td>ydayvar</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>116</td>
</tr>
<tr>
<td>ydruavg</td>
<td>116</td>
</tr>
<tr>
<td>ydruavg</td>
<td>116</td>
</tr>
<tr>
<td>ydruavg</td>
<td>116</td>
</tr>
<tr>
<td>ydruavg</td>
<td>116</td>
</tr>
<tr>
<td>ydruavg</td>
<td>116</td>
</tr>
<tr>
<td>ydruavg</td>
<td>116</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>sin</td>
<td>66</td>
</tr>
<tr>
<td>sinf</td>
<td>19</td>
</tr>
<tr>
<td>smooth9</td>
<td>157</td>
</tr>
<tr>
<td>sp2gp</td>
<td>143</td>
</tr>
<tr>
<td>sp2gpl</td>
<td>143</td>
</tr>
<tr>
<td>sp2sp</td>
<td>143</td>
</tr>
<tr>
<td>splitcode</td>
<td>28</td>
</tr>
<tr>
<td>splitday</td>
<td>30</td>
</tr>
<tr>
<td>splitgrid</td>
<td>28</td>
</tr>
<tr>
<td>splithour</td>
<td>30</td>
</tr>
<tr>
<td>splitlevel</td>
<td>28</td>
</tr>
<tr>
<td>splitmon</td>
<td>30</td>
</tr>
<tr>
<td>splitname</td>
<td>28</td>
</tr>
<tr>
<td>splitparam</td>
<td>28</td>
</tr>
<tr>
<td>splitsea</td>
<td>30</td>
</tr>
<tr>
<td>splitsel</td>
<td>31</td>
</tr>
<tr>
<td>splitthbnum</td>
<td>28</td>
</tr>
<tr>
<td>splityear</td>
<td>30</td>
</tr>
<tr>
<td>splitzaxis</td>
<td>28</td>
</tr>
<tr>
<td>sgr</td>
<td>66</td>
</tr>
<tr>
<td>sqrt</td>
<td>66</td>
</tr>
<tr>
<td>stdcm</td>
<td>158</td>
</tr>
<tr>
<td>strbre</td>
<td>163</td>
</tr>
<tr>
<td>strgal</td>
<td>163</td>
</tr>
<tr>
<td>strwin</td>
<td>162</td>
</tr>
<tr>
<td>sub</td>
<td>68</td>
</tr>
<tr>
<td>subc</td>
<td>67</td>
</tr>
<tr>
<td>subtrend</td>
<td>124</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tan</td>
<td>66</td>
</tr>
<tr>
<td>timavg</td>
<td>97</td>
</tr>
<tr>
<td>timcor</td>
<td>120</td>
</tr>
<tr>
<td>timcovar</td>
<td>121</td>
</tr>
<tr>
<td>timmax</td>
<td>97</td>
</tr>
<tr>
<td>timmean</td>
<td>97</td>
</tr>
<tr>
<td>timmin</td>
<td>97</td>
</tr>
<tr>
<td>timpctl</td>
<td>98</td>
</tr>
<tr>
<td>timselavg</td>
<td>93</td>
</tr>
<tr>
<td>timselmax</td>
<td>93</td>
</tr>
<tr>
<td>timselmean</td>
<td>93</td>
</tr>
<tr>
<td>timselmin</td>
<td>93</td>
</tr>
<tr>
<td>timselpctl</td>
<td>94</td>
</tr>
<tr>
<td>timselstd</td>
<td>93</td>
</tr>
<tr>
<td>timselsum</td>
<td>93</td>
</tr>
<tr>
<td>timtotal</td>
<td>158</td>
</tr>
<tr>
<td>timstd</td>
<td>97</td>
</tr>
<tr>
<td>timsum</td>
<td>97</td>
</tr>
<tr>
<td>timvar</td>
<td>97</td>
</tr>
<tr>
<td>trend</td>
<td>124</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>145</td>
</tr>
<tr>
<td>v</td>
<td>145</td>
</tr>
<tr>
<td>w</td>
<td>162</td>
</tr>
<tr>
<td>y</td>
<td>71</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
<tr>
<td>ydruavg</td>
<td>110</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>145</td>
</tr>
<tr>
<td>V</td>
<td>23</td>
</tr>
<tr>
<td>W</td>
<td>162</td>
</tr>
<tr>
<td>Y</td>
<td>71</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Operator</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>145</td>
</tr>
<tr>
<td>V</td>
<td>23</td>
</tr>
<tr>
<td>W</td>
<td>162</td>
</tr>
<tr>
<td>Y</td>
<td>71</td>
</tr>
<tr>
<td>Operator</td>
<td>Page</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
</tr>
<tr>
<td>ydrunsum</td>
<td>116</td>
</tr>
<tr>
<td>ydrunvar</td>
<td>116</td>
</tr>
<tr>
<td>yearavg</td>
<td>105</td>
</tr>
<tr>
<td>yearmax</td>
<td>105</td>
</tr>
<tr>
<td>yearmean</td>
<td>105</td>
</tr>
<tr>
<td>yearmin</td>
<td>105</td>
</tr>
<tr>
<td>yearsum</td>
<td>105</td>
</tr>
<tr>
<td>yearvar</td>
<td>105</td>
</tr>
<tr>
<td>yhouradd</td>
<td>72</td>
</tr>
<tr>
<td>yhouravg</td>
<td>109</td>
</tr>
<tr>
<td>yhourdiv</td>
<td>72</td>
</tr>
<tr>
<td>yhourmax</td>
<td>109</td>
</tr>
<tr>
<td>yhourmean</td>
<td>109</td>
</tr>
<tr>
<td>yhourmul</td>
<td>72</td>
</tr>
<tr>
<td>yhoursd</td>
<td>109</td>
</tr>
<tr>
<td>yhoursub</td>
<td>72</td>
</tr>
<tr>
<td>yhoursum</td>
<td>109</td>
</tr>
<tr>
<td>yhourvar</td>
<td>109</td>
</tr>
<tr>
<td>ymonadd</td>
<td>70</td>
</tr>
<tr>
<td>ymonavg</td>
<td>112</td>
</tr>
<tr>
<td>ymondiv</td>
<td>70</td>
</tr>
<tr>
<td>ymonmax</td>
<td>112</td>
</tr>
<tr>
<td>ymonmean</td>
<td>112</td>
</tr>
<tr>
<td>ymonmin</td>
<td>112</td>
</tr>
<tr>
<td>ymonmul</td>
<td>70</td>
</tr>
<tr>
<td>ymonpctl</td>
<td>113</td>
</tr>
<tr>
<td>ymonstd</td>
<td>112</td>
</tr>
<tr>
<td>ymonsub</td>
<td>70</td>
</tr>
<tr>
<td>ymonsum</td>
<td>112</td>
</tr>
<tr>
<td>ymonvar</td>
<td>112</td>
</tr>
<tr>
<td>yseasavg</td>
<td>114</td>
</tr>
<tr>
<td>yseasmax</td>
<td>114</td>
</tr>
<tr>
<td>yseasmean</td>
<td>114</td>
</tr>
<tr>
<td>yseasmin</td>
<td>114</td>
</tr>
<tr>
<td>yseaspctl</td>
<td>115</td>
</tr>
<tr>
<td>yseastrd</td>
<td>114</td>
</tr>
<tr>
<td>yseassum</td>
<td>114</td>
</tr>
<tr>
<td>yseasvar</td>
<td>114</td>
</tr>
<tr>
<td>zaxisdes</td>
<td>23</td>
</tr>
<tr>
<td>zonavg</td>
<td>87</td>
</tr>
<tr>
<td>zonmax</td>
<td>87</td>
</tr>
<tr>
<td>zonmean</td>
<td>87</td>
</tr>
<tr>
<td>zonmin</td>
<td>87</td>
</tr>
<tr>
<td>zonpctl</td>
<td>87</td>
</tr>
<tr>
<td>zonstd</td>
<td>87</td>
</tr>
<tr>
<td>zonsum</td>
<td>87</td>
</tr>
<tr>
<td>zonvar</td>
<td>87</td>
</tr>
</tbody>
</table>