
CDO

Release 2.6.4

Uwe Schulzweida

Sep 04, 2026

CONTENTS

1	Introduction	1
1.1	Installation	1
1.1.1	Unix	1
1.1.2	MacOS	3
1.1.3	Windows	3
1.2	Usage	4
1.2.1	Options	4
1.2.2	Environment Variables	7
1.2.3	Operators	7
1.2.4	Parallelized operators	8
1.2.5	Operator parameter	8
1.2.6	Operator chaining	8
1.2.7	Chaining Benefits	9
1.3	Advanced Usage	9
1.3.1	Wildcards	10
1.3.2	Argument Groups	10
1.3.3	Applying an operator or chain to multiple inputs	11
1.3.4	Apply with [:] notation	11
1.3.5	Apply Keyword (LEGACY)	12
1.4	Memory Requirements	13
1.5	Query	13
1.5.1	Keyword: group	14
1.5.2	Keyword: cell	14
1.6	Horizontal grids	14
1.6.1	Grid area weights	14
1.6.2	Grid description	14
1.6.3	ICON - Grid File Server	19
1.7	Z-axis description	19
1.8	Time axis	20
1.8.1	Absolute time	21
1.8.2	Relative time	21
1.8.3	Conversion of the time	21
1.9	Parameter table	21
1.10	Missing values	21
1.10.1	Mean and average	22
1.11	Percentile	23
1.11.1	Percentile over timesteps	23
1.12	Regions	23
2	Reference manual	25
2.1	Information	26
2.1.1	Info	27
2.1.2	Sinfo	29
2.1.3	XSinfo	31
2.1.4	Diff	33
2.1.5	Ninfo	35

2.1.6	Showinfo	37
2.1.7	Showattribute	39
2.1.8	Filedes	40
2.2	File operation	42
2.2.1	Copy	43
2.2.2	Tee	44
2.2.3	Pack	45
2.2.4	Unpack	46
2.2.5	Setchunkspec	47
2.2.6	Setfilter	48
2.2.7	Bitrounding	49
2.2.8	Replace	51
2.2.9	Duplicate	52
2.2.10	Mergegrid	53
2.2.11	Merge	54
2.2.12	Split	56
2.2.13	Splittime	58
2.2.14	Splitsel	60
2.2.15	Splitdate	61
2.2.16	Distgrid	62
2.2.17	Collgrid	64
2.3	Selection	66
2.3.1	Query	67
2.3.2	Select	69
2.3.3	Selmulti	71
2.3.4	Selvar	73
2.3.5	Seltime	76
2.3.6	Selbox	78
2.3.7	Selregion	80
2.3.8	Selgridcell	81
2.3.9	Samplegrid	82
2.3.10	Selyearidx	83
2.3.11	Seltimeidx	84
2.3.12	Selsurface	85
2.4	Conditional selection	86
2.4.1	Cond	87
2.4.2	Cond2	88
2.4.3	Condc	89
2.4.4	Mapreduce	90
2.5	Comparison	91
2.5.1	Comp	92
2.5.2	Compc	94
2.5.3	Ymoncomp	96
2.5.4	Yseascomp	97
2.6	Modification	98
2.6.1	Setattribute	100
2.6.2	Setpartab	102
2.6.3	Set	104
2.6.4	Settime	106
2.6.5	Change	108
2.6.6	Setgrid	110
2.6.7	Setzaxis	112
2.6.8	Invert	113
2.6.9	Invertlev	113
2.6.10	Shiftxy	114
2.6.11	Maskregion	115
2.6.12	Maskbox	116
2.6.13	Setbox	118

2.6.14	Enlarge	120
2.6.15	Setmiss	121
2.6.16	Vertfillmiss	124
2.6.17	Timfillmiss	124
2.6.18	Setgridcell	126
2.7	Arithmetic	127
2.7.1	Expr	129
2.7.2	Math	133
2.7.3	Arithc	135
2.7.4	Arith	136
2.7.5	Dayarith	138
2.7.6	Monarith	139
2.7.7	Yeararith	140
2.7.8	Yhourarith	141
2.7.9	Ydayarith	142
2.7.10	Ymonarith	143
2.7.11	Ysearith	144
2.7.12	Arithdays	145
2.7.13	Arithlat	146
2.8	Statistic	147
2.8.1	Timcumsum	154
2.8.2	Consecstat	155
2.8.3	Varsstat	156
2.8.4	Ensstat	158
2.8.5	Ensstat2	161
2.8.6	Ensval	162
2.8.7	Fldstat	164
2.8.8	Zonstat	167
2.8.9	Merstat	169
2.8.10	Gridboxstat	171
2.8.11	Remapstat	173
2.8.12	Vertstat	175
2.8.13	Timselstat	177
2.8.14	Timselpctl	179
2.8.15	Runstat	180
2.8.16	Runpctl	182
2.8.17	Timstat	183
2.8.18	Timpctl	185
2.8.19	Hourstat	186
2.8.20	Hourpctl	188
2.8.21	Daystat	189
2.8.22	Daypctl	191
2.8.23	Monstat	192
2.8.24	Monpctl	194
2.8.25	Timyearstat	195
2.8.26	Yearmonstat	196
2.8.27	Yearstat	197
2.8.28	Yearpctl	200
2.8.29	Seasstat	201
2.8.30	Seaspctl	203
2.8.31	Yhourstat	204
2.8.32	Dhourstat	206
2.8.33	Dminutestat	208
2.8.34	Ydaystat	210
2.8.35	Ydaypctl	212
2.8.36	Ymonstat	213
2.8.37	Ymonpctl	215
2.8.38	Yseasstat	216

2.8.39	Yseaspctl	218
2.8.40	Ydrunstat	219
2.8.41	Ydrunpctl	222
2.9	Correlation	224
2.9.1	Fldcor	225
2.9.2	Timcor	226
2.9.3	Fldcovar	227
2.9.4	Timcovar	227
2.10	Regression	228
2.10.1	Regres	229
2.10.2	Detrend	230
2.10.3	Trend	231
2.10.4	Trendarith	232
2.11	EOFs	233
2.11.1	EOF	234
2.11.2	Eofcoeff	236
2.12	Interpolation	238
2.12.1	Remapbil	239
2.12.2	Remapbic	241
2.12.3	Remapknn	243
2.12.4	Remapcon	245
2.12.5	Remaplaf	247
2.12.6	Remap	249
2.12.7	Remapeta	250
2.12.8	Vertintml	252
2.12.9	Vertintap	253
2.12.10	Vertintgh	254
2.12.11	Intlevel	255
2.12.12	Intlevel3d	256
2.12.13	Inttime	257
2.12.14	Intyear	258
2.13	Transformation	259
2.13.1	Spectral	260
2.13.2	Speconv	262
2.13.3	Wind2	263
2.13.4	Wind	264
2.13.5	Fourier	266
2.14	Import/Export	267
2.14.1	Importbinary	268
2.14.2	Importcmsaf	269
2.14.3	Input	271
2.14.4	Output	272
2.14.5	Outputtab	274
2.14.6	Outputgmt	276
2.15	Graphic with Magics	278
2.15.1	Magplot	279
2.15.2	Magvector	282
2.15.3	Maggraph	284
2.16	Climate Model Output Rewriting	286
2.16.1	CMOR	287
2.17	Climate indices	290
2.17.1	Eca_cdd	293
2.17.2	Eca_cfd	294
2.17.3	Eca_csu	295
2.17.4	Eca_cwd	296
2.17.5	Eca_cwdi	297
2.17.6	Eca_cwfi	298
2.17.7	Eca_etr	299

2.17.8	Eca_fd	300
2.17.9	Eca_gsl	301
2.17.10	Eca_hd	302
2.17.11	Eca_hwdi	303
2.17.12	Eca_hwfi	304
2.17.13	Eca_id	305
2.17.14	Eca_r75p	306
2.17.15	Eca_r75ptot	307
2.17.16	Eca_r90p	308
2.17.17	Eca_r90ptot	309
2.17.18	Eca_r95p	310
2.17.19	Eca_r95ptot	311
2.17.20	Eca_r99p	312
2.17.21	Eca_r99ptot	313
2.17.22	Eca_pd	314
2.17.23	Eca_rr1	316
2.17.24	Eca_rx1day	317
2.17.25	Eca_rx5day	318
2.17.26	Eca_sdii	319
2.17.27	Eca_su	320
2.17.28	Eca_tg10p	321
2.17.29	Eca_tg90p	322
2.17.30	Eca_tn10p	323
2.17.31	Eca_tn90p	324
2.17.32	Eca_tr	325
2.17.33	Eca_tx10p	326
2.17.34	Eca_tx90p	327
2.17.35	Eca_etccdi	328
2.18	Miscellaneous	330
2.18.1	Gradsdes	332
2.18.2	Afterburner	334
2.18.3	Filter	336
2.18.4	Gridcell	338
2.18.5	Smooth	339
2.18.6	Deltat	340
2.18.7	Replacevalues	341
2.18.8	Getgridcell	342
2.18.9	Vargen	343
2.18.10	Timsort	345
2.18.11	WindTrans	346
2.18.12	Rotuv	347
2.18.13	Mrotuvb	348
2.18.14	Mastrfu	349
2.18.15	Pressure	350
2.18.16	Derivepar	351
2.18.17	Adisit	353
2.18.18	Rhopot	354
2.18.19	Histogram	355
2.18.20	Sethalo	356
2.18.21	Wct	357
2.18.22	Fdns	358
2.18.23	Strwin	359
2.18.24	Strbre	360
2.18.25	Strgal	361
2.18.26	Hurr	362
2.18.27	CMORlite	363
2.18.28	Verifygrid	365
2.18.29	Healpix	366

	2.18.30	Symmetrize	367
	2.18.31	NCL_wind	368
3		Contributors	369
	3.1	History	369
	3.2	External sources	369
	3.3	Contributors	369
4		Environment Variables	371
5		Parallelized operators	372
6		Standard name table	374
7		Grid description examples	375
	7.1	Example of a curvilinear grid description	375
	7.2	Example description for an unstructured grid	376
		Bibliography	377
		Index	379

1 Introduction

The Climate Data Operators (**CDO**) software is a collection of many operators for standard processing of climate and forecast model data. The operators include simple statistical and arithmetic functions, data selection and subsampling tools, and spatial interpolation.

CDO was developed to have the same set of processing functions for GRIB [[GRIB](#)] and NetCDF [[NetCDF](#)] datasets in one package.

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

There are some limitations for GRIB and NetCDF datasets:

GRIB datasets have 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. Multiple fields in single GRIB2 messages are not supported!

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 main **CDO** features are:

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

Latest PDF documentation can be found [here](#).

1.1 Installation

CDO is supported in different operating systems such as Unix, macOS and Windows. This section describes how to install **CDO** in those platforms. More examples are found on the main website (<https://code.mpimet.mpg.de/projects/cdo/wiki>)

1.1.1 Unix

Prebuilt CDO packages

Prebuilt **CDO** versions are available in online Unix repositories, and you can install them by typing on the Unix terminal

```
apt-get install cdo
```

Note that prebuilt libraries do not offer the most recent version, and their version might vary with the Unix system. It is recommended to build from the source or Conda environment for an updated version or a customised setting.

Building from sources

CDO uses the GNU configure and build system for compilation. The only requirement is a working ISO C++20 and C11 compiler.

First go to the [download page](https://code.mpimet.mpg.de/projects/cdo) (<https://code.mpimet.mpg.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](https://www.unidata.ucar.edu/software/netcdf) library (<https://www.unidata.ucar.edu/software/netcdf>) version 4.3.3 or higher. This library is needed to process NetCDF [[NetCDF](#)] files with **CDO**.
- ECMWF [ecCodes](https://stage.ecmwf.int/en/computing/software) library (<https://stage.ecmwf.int/en/computing/software>) version 2.3.0 or higher. This library is needed to process GRIB2 files with **CDO**.
- HDF5 [szip](https://support.hdfgroup.org/documentation/hdf5/latest/group___s_z_i_p.html) library (https://support.hdfgroup.org/documentation/hdf5/latest/group___s_z_i_p.html) version 2.1 or higher. This library is needed to process szip compressed GRIB [[GRIB](#)] files with **CDO**.
- [HDF5](https://www.hdfgroup.org) library (<https://www.hdfgroup.org>) version 1.6 or higher. This library is needed to import CM-SAF [[CM-SAF](#)] HDF5 files with the **CDO** operator **import_cmsaf**.
- [PROJ](https://proj.org) library (<https://proj.org>) version 5.0 or higher. This library is needed to convert Sinusoidal and Lambert Azimuthal Equal Area coordinates to geographic coordinates, e.g. for remapping.
- [Magics](https://stage.ecmwf.int/en/computing/software) library (<https://stage.ecmwf.int/en/computing/software>) version 2.18 or higher. This library is needed to create contour, vector and graph plots with **CDO**.

Compilation

Compilation is done by performing the following steps:

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

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

2. Run the configure script:

```
./configure
```

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

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

- and with ecCodes:

```
./configure --with-ecCodes=<ecCodes 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.

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.

Conda

Conda is an open-source package manager and environment management system for various languages (Python, R, etc.). Conda is installed via Anaconda or Miniconda. Unlike Anaconda, miniconda is a lightweight conda distribution. They can be downloaded from the main conda Website (<https://conda.io/projects/conda/en/latest/user-guide/install/linux.html>) or on the terminal

```
wget https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-x86_64.sh
bash Anaconda3-2021.11-Linux-x86_64.sh
source ~/.bashrc
```

and

```
wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
sh Miniconda3-latest-Linux-x86_64.sh
```

Upon setting your conda environment, you can install **CDO** using conda

```
conda install cdo
conda install python-cdo
```

1.1.2 MacOS

Among the macOS package managers, **CDO** can be installed from Homebrew and MacPorts. The installation via Homebrew is a straightforward process on the terminal

```
brew install cdo
```

Similarly, MacPorts

```
port install cdo
```

In contrast to Homebrew, MacPorts allows you to enable GRIB2, szip compression and Magics++ graphics in **CDO** installation.

```
port install cdo +grib_api +magicssp +szip
```

In addition, you could also set **CDO** via Conda as Unix. You can follow this [tutorial](https://conda.io/projects/conda/en/latest/user-guide/install/macos.html) to install Anaconda or Miniconda in your computer (<https://conda.io/projects/conda/en/latest/user-guide/install/macos.html>). Then, you can install cdo by

```
conda install -c conda-forge cdo
```

1.1.3 Windows

Currently, **CDO** is not supported in Windows system and the binary is not available in the Windows conda repository. Therefore, **CDO** needs to be set in a virtual environment. Here, it covers the installation of **CDO** using Windows Subsystem for Linux (WSL) and virtual machines.

WSL

WSL emulates Unix in your Windows system. Then, you can install Unix libraries and software such as **CDO** or the Linux Conda distribution in your computer. Also, it allows you to directly share your files between your Windows and the WSL environment. However, more complex functions that require a graphical interface are not allowed.

In Windows 10 or newer, WSL can be readily set in your cmd by typing

```
wsl --install
```

This command will install, by default, Ubuntu 20.04 in WSL2. You could also choose a different system from this list.

```
wsl -l -o
```

Then, you can install your WSL environment as

```
wsl --install -d NAME
```

Virtual machine

Virtual machines can emulate different operating systems in your computer. Virtual machines are guest computers mounted inside your host computer. You can set a Linux distribution in your Windows device in this particular case. The advantages of Virtual machines to WSL are the graphical interface and the fully operational Linux system. You can follow any tutorial on the internet such as this one <https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox#1-overview>

Finally, you can install **CDO** following any method listed in the section *Unix*.

1.2 Usage

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

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

1.2.1 Options

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

-a, --absolute_taxis

Generate an absolute time axis.

--async_read <true|false>

Read input data asynchronously [default: false]. Available for the operators: *diff*, *info*, *trend*, *detrend*, *Timstat*

-b <nbits>

Set the number of bits for the output precision. The valid precisions depend on the file format:

format	nbits
grb1, grb2	P1 - P24
nc1, nc2, nc4, nc4c, nc5	I8/I16/I32/F32/F64
nc4, nc4c, nc5	U8/U16/U32
grb2, srv, ext, ieg	F32/F64

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

-C, --color

Colorized output messages.

-c, --check_data_range

Enables checks for data overflow.

--chunksize <size>

NetCDF4 chunk size (x/y dimension).

--chunkspec <chunkspec>

NetCDF4 specify chunking for dimensions (x,y,z,t).

--cmor

CMOR conform NetCDF output.

--copy_chunkspec

Copy chunk specification.

--double

Uses double precision floats for reading data. This is the default for 64-bit float data.

--eccodes

Use [ecCodes] to decode/encode GRIB1 messages.

-F, --filter <filterspec>

NetCDF4 filter specification. A filter specification consists of the filterId and the filter parameters. **CDO** supports multiple filters connected with '|'. Here is a filter specification for bzip2 (filterId: 307) combined with szip (filterId:4): "307,9|4,32,32". The environment variable HDF5_PLUGIN_PATH must point to the directory with the installed plugins.

-f <format>

Set the output file format. The valid file formats are:

File format	format
GRIB version 1	grb1/grb
GRIB version 2	grb2
NetCDF	nc1
NetCDF version 2 (64-bit offset)	nc2/nc
NetCDF-4 (HDF5)	nc4
NetCDF-4 classic	nc4c
NetCDF version 5 (64-bit data)	nc5
SERVICE	srv
EXTRA	ext
IEG	ieg

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

--force

Forcing a CDO process.

-g <grid>

Define the default grid description by name or from file (see [Grid description](#)). Available grid names are: global_<DXY>, zonal_<DY>, r<NX>x<NY>, lon=<LON>/lat=<LAT>, F<X>, N<X>, O<X>, gme<NI>, hpr<ZOOM>.

-h, --help

Help information for the operators.

--netcdf_hdr_pad, --hdr_pad, --header_pad <nbr>

Pad NetCDF output header with *nbr* bytes.

-k <chunkType>

NetCDF4 chunk type: auto, grid or lines.

-m <missVal>

Set the missing value of non NetCDF files (default: -9e+33).

--nofile <num>

Set maximum number of files that can be opened.

--no_history

Do not append to NetCDF *history* global attribute.

--nsb <1-23>

Number of significant bits used for bit-rounding (Set NetCDF quantize parameter NC_QUANTIZE_BITROUND).

-0, --overwrite

Overwrite existing output file, if checked. Existing output file is checked only for: *Ensstat*, *merge*, *mergetime*

--operators

List of all operators.

-P <nthreads>

Set number of OpenMP threads (Only available if OpenMP support was compiled in).

-p

Short for *--async_read true*

--pedantic

Warnings count as errors.

--percentile <method>

Methods: nrank, nist, rtype8, < NumPy method (linear | lower | higher | nearest | ...) >

--precision <float_digits[,double_digits]>

Precision to use in displaying floating-point data (default: 7,15).

--query <keyValueList>

Pre-selects a subset of the data cube from a NetCDF dataset. Available key words:

Keyword	Description
group	Select group (group=groupName)
name	Variable names (name=var1,var2,...)
cell	Cell index range (cell=first/to/last)
layer	Layer index range (layer=first/to/last)
step	Time step index range (step=first/to/last)
startdate	Start date (format: YYYY-MM-DD[Thh:mm:ss])
enddate	End date (format: YYYY-MM-DD[Thh:mm:ss])

--reduce_dim

Reduce all NetCDF dimensions.

--reduce_dims <dimensions>

Reduce specific NetCDF dimensions (x,y,z,t).

-R, --regular

Convert GRIB1 data from global reduced to regular Gaussian grid (only with cgribex lib).

-r, --relative_taxis

Generate a relative time axis.

--remove_chunkspec

Remove chunk specification.

-S, --diagnostic

Create a diagnostic output stream for the module *Timstat*. This stream contains the number of non missing values for each output period.

-s, --silent

Silent mode.

--shuffle

Specify shuffling of variable data bytes before compression (NetCDF).

--single

Uses single precision floats for reading data. This is the default for 32-bit float data.

--sortname

Alphanumeric sorting of NetCDF parameter names.

-t <partab>

Set the GRIB1 (cgribex) default parameter table name or file (see *Parameter table*). Predefined tables are: echam4 echam5 echam6 mpiom1 ecmwf remo

--timestat_date <srcdate>

Target timestamp (temporal statistics): first, middle, midhigh or last source timestep.

-V, --version

Print the version number.

-v, --verbose

Print extra details for some operators.

-w

Disable warning messages.

--worker <num>

Number of workers to decode/decompress GRIB records.

-z <aec|jpeg|zip[_1-9]|zstd[_1-19]>

Sets the compression type for the output data:

aec	AEC compression of GRIB2 records
jpeg	JPEG compression of GRIB2 records
zip	Deflate compression of NetCDF4 arrays
zstd	Zstandard compression of NetCDF4 arrays

1.2.2 Environment Variables

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

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

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

1.2.3 Operators

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

1.2.4 Parallelized operators

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

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

```
cdo -P 8 remapbil,targetgrid infile outfile
```

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

1.2.5 Operator parameter

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

- **STRING**

String parameters require quotes if the string contains blanks or other characters interpreted by the shell. The following command select variables with the name `pressure` and `tsurf`:

```
cdo selvar,pressure,tsurf infile outfile
```

- **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 infile outfile
```

- **BOOL**

Boolean parameter in the following representation TRUE/FALSE, T/F or 0/1. To disable the weighting by grid cell area in the calculation of a field mean, use:

```
cdo fldmean,weights=FALSE infile outfile
```

- **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 infile outfile
```

The result is the same as:

```
cdo selday,5,6,7,8,9 infile outfile
```

1.2.6 Operator chaining

Operator chaining allows to combine two or more operators on the command line into a single **CDO** call. This allows the creation of complex operations out of more simple ones: reductions over several dimensions, file merges and all kinds of analysis processes. All operators with a fixed number of input streams and one output stream can pass the result directly to another operator. For differentiation between files and operators all operators must be written with a prepended "-" when chaining.

```
cdo -monmean -add -mulc,2.0 infile1 -daymean infile2 outfile (CDO example call)
```

Here `monmean` will have the output of `add` while `add` takes the output of `mulc,2.0` and `daymean`. `infile1` and `infile2` are inputs for their predecessor. When mixing operators with an arbitrary number of input streams extra care needs to be taken. The following examples illustrates why.

1. `cdo info -timavg infile1 infile2`

```
2. cdo info -timavg infile?
3. cdo timavg infile1 tmpfile
   cdo info tmpfile infile2
   rm tmpfile
```

All three examples produce identical results. The time average will be computed only on the first input file.

Note

In section *Argument Groups* we introduce argument groups which will make this a lot easier and less error prone.

Note

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

1.2.7 Chaining Benefits

Combining operators can have several benefits. The most obvious is a performance increase through reducing disk I/O:

```
cdo sub -dayavg infile2 -timavg infile1 outfile
```

instead of:

```
cdo timavg infile1 tmp1
cdo dayavg infile2 tmp2
cdo sub tmp2 tmp1 outfile
rm tmp1 tmp2
```

Especially with large input files the reading and writing of intermediate files can have a big influence on the overall performance.

A second aspect is the execution of operators: Limited by the algorithms potentially all operators of a chain can run in parallel.

1.3 Advanced Usage

In this section we will introduce advanced features of **CDO**. These include operator grouping which allows to write more complex **CDO** calls and the apply keyword which allows to shorten calls that need an operator to be executed on multiple files as well as wildcards which allow to search paths for file signatures. These features have several restrictions and follow rules that depend on the input/output properties. These required properties of operators can be investigated with the following commands which will output a list of operators that have selected properties:

```
cdo --attribs [arbitrary/filesOnly/onlyFirst/noOutput/obase]
```

- *arbitrary* describes all operators where the number of inputs is not defined.
- *filesOnly* are operators that can have other operators as input.
- *onlyFirst* shows which operators can only be at the most left position of the polish notation argument chain.
- *noOutput* are all operators that do not print to any file (e.g info).
- *obase* Here obase describes an operator that does not use the output argument as file but e.g as a file name base (output base). This is almost exclusively used for operators that split input files.

```
cdo -splithour baseName_
could result in: baseName_1 baseName_2 ... baseName_N
```

For checking a single or multiple operator directly the following usage of `-{}-attribs` can be used:

```
cdo --attribs operatorName
```

1.3.1 Wildcards

Wildcards are a standard feature of command line interpreters (shells) on many operating systems. They are placeholder characters used in file paths that are expanded by the interpreter into file lists. For further information the [Advanced Bash Scripting Guide](#) is a valuable source of information. Handling of input is a central issue for **CDO** and in some circumstances it is not enough to use the wildcards from the shell. That's why **CDO** can handle them on its own.

all files	2020-2-01.txt 2020-2-11.txt 2020-2-15.txt 2020-3-01.txt 2020-3-02.txt 2020-3-12.txt 2020-3-13.txt 2020-3-15.txt 2021.grb 2022.grb
wildcard	filelist results
2020-3* and 2020-3-?? .txt	2020-3-01.txt 2020-3-02.txt 2020-3-12.txt 2020-3-13.txt 2020-3-15.txt
2020-3-?1.txt	2020-3-01.txt
*.grb	2021.grb 2020.grb

Use single quotes if the input stream names matched to a single wildcard expression. In this case **CDO** will do the pattern matching and the output can be combined with other operators. Here is an example for this feature:

```
cdo timavg -select,name=temperature 'infile?' outfile
```

In earlier versions of **CDO** this was necessary to have the right files parsed to the right operator. Newer versions support this with the argument grouping feature (see [Argument Groups](#)). We advise the use of the grouping mechanism instead of the single quoted wildcards since this feature could be deprecated in future versions.

Note

Wildcard expansion is not available on operating systems without the `glob()` function!

1.3.2 Argument Groups

In section [Operator chaining](#) we described that it is not possible to chain operators with an arbitrary number of inputs. In this section we want to show how this can be achieved through the use of *operator grouping* with angled brackets `[]`. Using these brackets **CDO** can be assigned the inputs to their corresponding operators during the execution of the command line. The ability to write operator combination in a parentheses-free way is partly given up in favor of allowing operators with arbitrary number of inputs. This allows a much more compact way to handle large number of input files.

The following example shows an example which we will transform from a non-working solution to a working one.

```
cdo -infon -div -fldmean -cat infileA -mulc,2.0 infileB -fldmax infileC
```

This example will throw the following error:

```
cdo (Abort):
-infon -div -fldmean -cat infileA -mulc,2.0 infileB -fldmax infileC
                                     ^ Operator cannot be assigned.

Reason:
```

(continues on next page)

(continued from previous page)

Multiple variable input operators used.
Use subgroups via [] to clarify relations (help: --argument_groups).

The error is raised by the operator *div*. This operator needs two input streams and one output stream, but the *cat* operator has claimed all possible streams on its right hand side as input because it accepts an arbitrary number of inputs. Hence it didn't leave anything for the remaining input or output streams of *div*. For this we can declare a group which will be passed to the operator left of the group.

```
cdo -infon -div -fldmean -cat [ infileA -mulc,2.0 infileB ] -fldmax infileC
```

For full flexibility it is possible to have groups inside groups:

```
cdo -infon -div -fldmean -cat [ infileA infileB -merge [ infileC1 infileC2 ] ] -  
↪fldmax infileD
```

1.3.3 Applying an operator or chain to multiple inputs

When working with medium or large number of similar files there is a common problem of a processing step (often a reduction) which needs to be performed on all of them before a more specific analysis can be applied. Usually this can be done in two ways: One option is to use a merge operator to glue everything together and chain the reduction step after it. The second option is to write a for-loop over all inputs which perform the basic processing on each of the files separately and call a merge operator on the results. Unfortunately both options have side-effects: The first one needs a lot of memory because all files are read in completely and reduced afterwards while the latter one creates a lot of temporary files. Both memory and disk IO can be bottlenecks and should be avoided. In **CDO** there exist two approaches to circumvent most drawbacks. The first is to use the more recent 'apply' feature using the [to_be_applied : applied_to] syntax, the second is an older approach and is only listed and documented for completeness sake. We highly recommend the more recent approach!

1.3.4 Apply with [:] notation

With the [to_be_applied : applied_to] syntax it is possible to prepend multiple inputs or chains with another chain. For example:

```
-mergetime [ -selname,tsurf : *.grb ]
```

would merge all GRIB files in the folder after selecting the variable *tsurf* from them.

In general the *to_be_applied* is applied in parallel to all related input streams (*applied_to*) before all streams are passed to operator next in the chain.

Usage and result of [:] notation

The following is an example with three input files:

```
cdo -mergetime [ -selname,tsurf : infile1 infile2 infile3 ] outfile
```

This would result in **CDO** executing as if the following was used:

```
cdo -mergetime -selname,tsurf infile1 -selname,tsurf infile2 -selname,tsurf_  
↪infile3 outfile
```

This notation is especially useful when combined with wildcards. The previous example can be shortened further.

```
cdo -mergetime [ -selname,tsurf : infile? ] outfile
```

As shown this feature allows to simplify commands with medium amount of files and to move reductions further back. This can also have a positive impact on the performance.

Notation simplifies command and execution

An example where performance can take a hit.

```
cdo -yearmean -selname,tsurf -mergetime [ f1 ... f40 ]
```

An improved but ugly to write example.

```
cdo -yearmean -mergetime [ -selname,tsurf f1 -selname,tsurf f2 ... -selname,  
→tsurf f40 ]
```

Apply saves the day. And creates the call above with much less typing.

```
cdo -yearmean -mergetime [ -selname,tsurf : f1 ... f40 ]
```

Further this notation allows to prepend full cdo chains with other operators:

```
cdo -info -mergetime [ -selname,tsurf : -addc,1 -mul f1 f2 -addc,4 f3 ]
```

Resolving internally to the following command:

```
info -mergetime -selname,tsurf [ -addc,1 -mul f1 f2 -selvar,tsurf -addc,4 f3 ]
```

1.3.5 Apply Keyword (LEGACY)

Originally the *apply* keyword was introduced for that purpose. We want to repeat that this is an older method and should not be used in newly written **CDO** commands! It can be used as an operator, but it needs at least one operator as a parameter, which is applied in parallel to all related input streams in a parallel way before all streams are passed to operator next in the chain.

Usage and result of apply keyword

The following is an example with three input files:

```
cdo -mergetime -apply,-selname,tsurf [ infile1 infile2 infile3 ] outfile
```

would result in:

```
cdo -mergetime -selname,tsurf infile1 -selname,tsurf infile2 -selname,tsurf,  
→infile3 outfile
```

Apply is especially useful when combined with wildcards. The previous example can be shortened further:

```
cdo -mergetime -apply,-selname,tsurf [ infile? ] outfile
```

As shown this feature allows to simplify commands with medium amount of files and to move reductions further back. This can also have a positive impact on the performance.

Apply keyword simplifies command and execution

An example where performance can take a hit:

```
cdo -yearmean -selname,tsurf -mergetime [ f1 ... f40 ]
```

An improved but ugly to write example:

```
cdo -yearmean -mergetime [ -selname,tsurf f1 -selname,tsurf f2 ... -selname,  
→tsurf f40 ]
```

Apply saves the day. And creates the call above with much less typing:

```
cdo -yearmean -mergetime [ -apply,-selname,tsurf [ f1 ... f40 ] ]
```

In the example in figure [simpleApply](#) the resulting call will dramatically save process interaction as well as execution times since the reduction (*selname,tsurf*) is applied on the files first. That means that the *mergetime* operator will

receive the reduced files and the operations for merging the whole data is saved. For other **CDO** calls further improvements can be made by adding more arguments to apply (*multiApply*)

Multi argument apply

A less performant example:

```
cdo -aReduction -anotherReduction -selname,tsurf -mergetime [ f1 ... f40 ]
```

```
cdo -mergetime -apply,"-aReduction -anotherReduction -selname,tsurf" [ f1 .  
→ .. f40 ]
```

Restrictions:

While the apply keyword can be extremely helpful it has several restrictions (for now!).

- Apply inputs can only be files, wildcards and operators that have 0 inputs and 1 output.
- Apply cannot be used as the first **CDO** operator.
- Apply arguments can only be operators with 1 input and 1 output.
- Grouping inside the Apply argument or input is not allowed.

1.4 Memory Requirements

This section roughly describes the memory requirements of **CDO**. **CDO** tries to use as little memory as possible. The smallest unit that is read by all operators is a horizontal field. The required memory depends mainly on the used operators, the data format, the data type and the size of the fields.

The operators have partly very different memory requirements. Many **CDO** modules like *Fldstat* process one horizontal field at a time. Memory-intensive modules such as *Ensstat* and *Timstat* require all fields of a time step to be held in memory. Of course, the memory requirements of each operator add up when they are combined. Some operators are parallelized with OpenMP. In multi-threaded mode (see option *-P*) the memory requirement can increase for these operators. This increase grows with the number of threads used.

The data type determines the number of bytes per value. Single precision floating point data occupies 4 bytes per value. All other data types are read as double precision floats and thus occupy 8 bytes per value. With the **CDO** option *--single* all data is read as single precision floats. This can reduce the memory requirement by a factor of 2.

1.5 Query

CDO supports the NetCDF classic data model with attributes that conform to the CF conventions. The **CDO** I/O interface always reads in a complete horizontal field. If the dataset is chunked and the chunks extend beyond the horizontal field, the chunks are cached.

Using the **query** feature allows you to reduce the view to the dataset before **CDO** processes it. This can improve performance and reduce memory requirements – particularly for high-resolution, chunked and compressed datasets, especially when the data has to be loaded over the network. The size of the required cache is also affected by this.

The following keys are available for pre-selecting a subset of a dataset:

Keyword	Description
group	Select a group (group=groupName)
name	Comma-separated list of variable names (name=var1,var2,...)
cell	Cell index range (cell=first[/to/last])
layer	Layer index range (layer=first[/to/last])
step	Time step index range (step=first[/to/last])
startdate	Start date (format: YYYY-MM-DD[Thh:mm:ss])
enddate	End date (format: YYYY-MM-DD[Thh:mm:ss])

The indices for keys **cell**, **layer** and **step** start at 1. The **cell** key is only implemented for unstructured and HEALPix grids.

The keys can be defined for all input streams using the global **CDO** option `--query`. Here is an example to select the first 100 timesteps of the variable `xxx`:

```
cdo --query step=1/to/100,name=xxx copy infile outfile
```

Use the query operator for one specific input stream:

```
cdo query,step=1/to/100,name=xxx infile outfile
```

1.5.1 Keyword: group

NetCDF-4 added support for hierarchical groups within NetCDF datasets. Groups are not compatible with the NetCDF classic data model. If the dataset contains more than one group, use the **group** key to select just one of them:

```
cdo --query group=groupName xsinfo infile
```

1.5.2 Keyword: cell

The **CDO** I/O interface always reads in a complete horizontal field. If the data is chunked and the number of grid points required is significantly smaller than the complete horizontal field, more data is read in than necessary. Use the **cell** key to select a contiguous set of grid cells, so that only the required data chunks are read in. This can significantly speed up the processing of high resolution compressed datasets. Here is an example of selecting only the grid point with the index 1234:

```
cdo --query cell=1234 info infile
```

Use the **CDO** *gridcellindex* operator to find the cell index based on the longitude and latitude coordinates.

1.6 Horizontal grids

Physical quantities of climate models are typically stored on a horizontal grid. **CDO** supports structured grids like regular lon/lat or curvilinear grids and also unstructured grids.

1.6.1 Grid area weights

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

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

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

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

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

1.6.2 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 (all remapping operators)
- Generating of variables (operator: *const*, *random*)

As described in the following, there are several possibilities to define a horizontal grid.

Predefined grids

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

Global regular grid: `global_<DXY>`

`global_<DXY>` defines a global regular lon/lat grid. The grid increment `<DXY>` can be chosen arbitrarily. The longitudes start at $\langle DXY \rangle / 2 - 180^\circ$ and the latitudes start at $\langle DXY \rangle / 2 - 90^\circ$. Adding `_b` also generates the grid cell bounds.

Regional regular grid: `dcw:<CountryCode>[_<DXY>]`

`dcw:<CountryCode>[_<DXY>]` defines a regional regular lon/lat grid from the country code. The default value of the optional grid increment `<DXY>` is 0.1 degree. The ISO two-letter country codes can be found on https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2. To define a state, append the state code to the country code, e.g. USAK for Alaska. For the coordinates of a country CDO uses the DCW (Digital Chart of the World) dataset from GMT. This dataset must be installed on the system and the environment variable `DIR_DCW` must point to it.

Zonal latitudes: `zonal_<DY>`

`zonal_<DY>` defines a grid with zonal latitudes only. The latitude increment `<DY>` can be chosen arbitrarily. The latitudes start at $\langle DY \rangle / 2 - 90^\circ$. The boundaries of each latitude are also generated. The number of longitudes is 1. A grid description of this type is needed to calculate the zonal mean (*zonmean*) for data on an unstructured grid.

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 chosen arbitrarily. The longitudes start at 0° with an increment of $(360/\langle NX \rangle)^\circ$. The latitudes go from south to north with an increment of $(180/\langle NY \rangle)^\circ$.

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

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

Full regular Gaussian grid: `F<X>`

`F<X>` defines a global regular Gaussian grid. `X` specifies the number of latitudes lines between the Pole and the Equator. The total number of latitudes and longitudes is: $nlat = X * 2$; $nlon = nlat * 2$. The longitudes start at 0° with an increment of $(360/nlon)^\circ$. The gaussian latitudes go from north to south.

Reduced Gaussian grid: `N<X>`

`N<X>` defines a global original ECMWF reduced Gaussian grid. `X` specifies the number of latitudes lines between the Pole and the Equator. Available grid: N16, N24, N32, N48, N64, N80, N96, N128, N160, N200, N256, N320, N400, N512, N576, N640, N800, N1024

Octahedral reduced Gaussian grid: `O<X>`

`O<X>` defines a global octahedral ECMWF reduced Gaussian grid. `X` specifies the number of latitudes lines between the Pole and the Equator.

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.

HEALPix grid: `hp<NSIDE>[_<ORDER>]`

HEALPix is an acronym for Hierarchical Equal Area isoLatitude Pixelization of a sphere. `hp<NSIDE>[_<ORDER>]` defines the parameters of a global HEALPix grid. The `NSIDE` parameter controls the resolution of the pixelization. It is the number of pixels on the side of each of the 12 top-level HEALPix pixels. The total number of grid pixels is $12 * \text{NSIDE} * \text{NSIDE}$. `NSIDE=1` generates the 12 ($H=4$, $K=3$) equal sized top-level HEALPix pixels. Only powers of two for `NSIDE` are allowed. `ORDER` sets the index ordering convention of the pixels, available are `nested` (default) or `ring` ordering. A shortcut for `hp<NSIDE>_nested` is `hpr<ZOOM>`. `ZOOM` is the refinement level and the relation to `NSIDE` is $zoom = \log_2(nside)$.

If the geographical coordinates are required in **CDO**, they are calculated from the HEALPix parameters. For this calculation the `astropy-healpix` C library is used.

Grids from data files

You can use the grid description from another 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 information about your variables and the grids. If there are more than one grid in the datafile the grid description of the first variable will be used. Add the extension `:N` to the name of the datafile to select grid number `N`.

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. Therefore 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_corners = 4 ;
    grid_rank = 2 ;
variables:
    int grid_dims(grid_rank) ;
    double grid_center_lat(grid_size) ;
        grid_center_lat:units = "degrees" ;
        grid_center_lat:bounds = "grid_corner_lat" ;
    double grid_center_lon(grid_size) ;
        grid_center_lon:units = "degrees" ;
        grid_center_lon:bounds = "grid_corner_lon" ;
    int grid_imask(grid_size) ;
        grid_imask:units = "unitless" ;
        grid_imask:coordinates = "grid_center_lon grid_center_lat" ;
    double grid_corner_lat(grid_size, grid_corners) ;
        grid_corner_lat:units = "degrees" ;
    double grid_corner_lon(grid_size, grid_corners) ;
        grid_corner_lon:units = "degrees" ;

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

CDO grids

All supported grids can also be described with the **CDO** grid description. The **CDO** grid description is an ASCII formatted file.

The following keywords can be used to describe a grid:

Key-word	Datatype	Description
gridtype	STRING	Type of the grid (gaussian, lonlat, curvilinear, unstructured).
gridsize	INTEGER	Size of the grid.
xsize	INTEGER	Size in x direction (number of longitudes).
ysize	INTEGER	Size in y direction (number of latitudes).
xvals	FLOAT ARRAY	X values of the grid cell center.
yvals	FLOAT ARRAY	Y values of the grid cell center.
nvertex	INTEGER	Number of the vertices for all grid cells.
xbounds	FLOAT ARRAY	X bounds of each gridbox.
ybounds	FLOAT ARRAY	Y bounds of each gridbox.
xfirst, xinc	FLOAT, FLOAT	Macros to define xvals with a constant increment, xfirst is the x value of the first grid cell center.
yfirst, yinc	FLOAT, FLOAT	Macros to define yvals with a constant increment, yfirst is the y value of the first grid cell center.
xunits	STRING	units of the x axis
yunits	STRING	units of the y axis

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

gridtype	lonlat	gaussian	projection	curvilinear	unstructured
gridsize	xsize*ysize	xsize*ysize	xsize*ysize	xsize*ysize	ncell
xsize	nlon	nlon	nx	nlon	gridsize
ysize	nlat	nlat	ny	nlat	gridsize
xvals	xsize	xsize	xsize	gridsize	gridsize
yvals	ysize	ysize	ysize	gridsize	gridsize
nvertex	2	2	2	4	nv
xbounds	2*xsize	2*xsize	2*xsize	4*gridsize	nv*gridsize
ybounds	2*ysize	2*ysize	2*xsize	4*gridsize	nv*gridsize
xunits	degrees	degrees	m	degrees	degrees
yunits	degrees	degrees	m	degrees	degrees

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

CDO grid description example of a T21 gaussian grid:

```

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

```

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

```

gridtype = lonlat
xsize    = 60

```

(continues on next page)

(continued from previous page)

```
ysize    = 30
xfirst   = -177
xinc     = 6
yfirst   = -87
yinc     = 6
```

The description for a projection is somewhat more complicated. Use the first section to describe the coordinates of the projection with the above keywords. Add the keyword **grid_mapping_name** to describe the mapping between the given coordinates and the true latitude and longitude coordinates. **grid_mapping_name** takes a string value that contains the name of the projection. A list of attributes can be added to define the mapping. The name of the attributes depend on the projection. The valid names of the projection and their attributes follow the NetCDF CF-Convention.

CDO supports the special grid mapping attribute **proj_params**. These parameter will be passed directly to the [\[PROJ\]](#) library to generate the geographic coordinates if needed.

The geographic coordinates of the following projections can be generated without the attribute **proj_params**, if all other attributes are available:

- **rotated_latitude_longitude**
- **lambert_conformal_conic**
- **lambert_azimuthal_equal_area**
- **sinusoidal**
- **polar_stereographic**

It is recommended to set the attribute **proj_params** also for the above projections to make sure all PROJ parameters are set correctly.

Here is an example of a **CDO** grid description using the attribute **proj_params** to define the PROJ parameter of a polar stereographic projection:

```
gridtype = projection
ysize    = 11
xsize    = 11
xunits   = "meter"
yunits   = "meter"
xfirst   = -638000
xinc     = 150
yfirst   = -3349350
yinc     = 150
grid_mapping = crs
grid_mapping_name = polar_stereographic
proj_params = "+proj=stere +lon_0=-45 +lat_ts=70 +lat_0=90 +x_0=0 +y_0=0"
```

The result is the same as using the CF conform Grid Mapping Attributes:

```
gridtype = projection
ysize    = 11
xsize    = 11
xunits   = "meter"
yunits   = "meter"
xfirst   = -638000
xinc     = 150
yfirst   = -3349350
yinc     = 150
grid_mapping = crs
grid_mapping_name = polar_stereographic
```

(continues on next page)

(continued from previous page)

```
straight_vertical_longitude_from_pole = -45.
standard_parallel = 70.
latitude_of_projection_origin = 90.
false_easting = 0.
false_northing = 0.
```

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

```
gridtype = projection
xsize    = 81
ysize    = 91
xunits   = "degrees"
yunits   = "degrees"
xfirst   = -19.5
xinc     = 0.5
yfirst   = -25.0
yinc     = 0.5
grid_mapping_name = rotated_latitude_longitude
grid_north_pole_longitude = -170
grid_north_pole_latitude = 32.5
```

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

1.6.3 ICON - Grid File Server

The geographic coordinates of the ICON model are located on an unstructured grid. This grid is stored in a separate grid file independent of the model data. The grid files are made available to the general public via a file server. Furthermore, these grid files are located at DKRZ under `/pool/data/ICON/grids`.

With the CDO `setgrid,<gridfile>` operator this grid information can be added to the data if needed. Here is an example:

```
cdo sellonlatbox,-20,60,10,70 -setgrid,<path_to_gridfile> icondatafile result
```

ICON model data in NetCDF format contains the global attribute `grid_file_uri`. This attribute contains a link to the appropriate grid file on the ICON grid file server. If the global attribute `grid_file_uri` is present and valid, the grid information can be added automatically. The `setgrid` operator is then no longer required. The environment variable `CDO_DOWNLOAD_PATH` can be used to select a directory for storing the grid file. If this environment variable is set, the grid file will be automatically downloaded from the grid file server to this directory if needed. If the grid file already exists in the current directory, the environment variable does not need to be set.

If the grid files are available locally, like at DKRZ, they do not need to be fetched from the grid file server. Use the environment variable `CDO_ICON_GRIDS` to set the root directory of the ICON grids. Here is an example for the ICON grids at DKRZ:

```
CDO_ICON_GRIDS=/pool/data/ICON
```

1.7 Z-axis description

Sometimes it is necessary to change the description of a z-axis. This can be done with the operator `setzaxis`. `setzaxis` replaces the Z-axis of all variables with the same number of levels. The parameter can be an ASCII file with a description of the vertical axis or the name of another dataset. This page describes the ASCII format.

The following keywords can be used to describe a z-axis:

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

zaxistype must be the first keyword in the file. The keywords **lbounds** and **ubounds** are optional. **vctsize** and **vct** are only necessary to define hybrid model levels.

Available z-axis types:

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

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

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

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

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

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

1.8 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.

1.8.1 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".

1.8.2 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".

1.8.3 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'.

1.9 Parameter table

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

Example of a **CDO** parameter table:

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

1.10 Missing values

Missing values are data points that are missing or invalid. Such data points are treated in a different way than valid data. Most **CDO** operators can handle missing values in a smart way. But if the missing value is within the range of valid data, it can lead to incorrect results. This applies to all arithmetic operations, but especially to logical operations when the missing value is 0 or 1.

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

The **CDO** use of the missing value is shown in the following tables, where one table is printed for each operation. The operations are applied to arbitrary numbers *a*, *b*, the special case 0, and the missing value *miss*.

For example the table named "addition" shows that the sum of an arbitrary number *a* and the missing value is the missing value, and the table named "multiplication" shows that 0 multiplied by missing value results in 0.

Table 1: Maximum

	b	miss
a	$\max(a, b)$	a
miss	b	miss

Table 2: Minimum

	b	miss
a	$\min(a, b)$	a
miss	b	miss

Table 3: Sum

	b	miss
a	$a + b$	a
miss	b	miss

Table 4: Addition

	b	miss
a	$a + b$	miss
miss	miss	miss

Table 5: Subtraction

	b	miss
a	$a - b$	miss
miss	miss	miss

Table 6: Multiplication

	b	miss	0
a	$a * b$	miss	0
miss	miss	miss	0
0	0	0	0

Table 7: Division

	b	miss	0
a	a/b	miss	miss
miss	miss	miss	miss
0	0	miss	miss

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

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

1.10.1 Mean and average

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

1.11 Percentile

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

Percentile method	Description
nrank	Nearest Rank method [default in CDO]
nist	The primary method recommended by NIST
rtype8	R's type=8 method
inverted_cdf	NumPy with percentile method='inverted_cdf' (R type=1)
averaged_inverted_cdf	NumPy with percentile method='averaged_inverted_cdf' (R type=2)
closest_observation	NumPy with percentile method='closest_observation' (R type=3)
interpolated_inverted_cdf	NumPy with percentile method='interpolated_inverted_cdf' (R type=4)
hazen	NumPy with percentile method='hazen' (R type=5)
weibull	NumPy with percentile method='weibull' (R type=6)
linear	NumPy with percentile method='linear' (R type=7) [default in NumPy and R]
median_unbiased	NumPy with percentile method='median_unbiased' (R type=8)
normal_unbiased	NumPy with percentile method='normal_unbiased' (R type=9)
lower	NumPy with percentile method='lower'
higher	NumPy with percentile method='higher'
midpoint	NumPy with percentile method='midpoint'
nearest	NumPy with percentile method='nearest'

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

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

Percentile P	nrank	nist	rtype8	NumPy linear	NumPy lower	NumPy higher	NumPy nearest
30th	20	21.5	23.5	27.5	20	35	35
40th	35	32	33	35	35	35	35
50th	35	37.5	37.5	37.5	35	40	40
75th	50	51.25	50.42	47.5	40	50	50
100th	55	55	55	55	55	55	55

1.11.1 Percentile over timesteps

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

1.12 Regions

The **CDO** operators *maskregion* and *selregion* can be used to mask and select regions. A region can be defined in two ways.

One possibility is to define the regions with an ASCII file. Each region is defined by a polygon. Each line of the polygon contains the longitude and latitude coordinates of a point. A description file for regions can contain several polygons, these must be separated by a line with the character &.

Here is a simple example of a polygon for a box with longitudes from 120E to 90W and latitudes from 20N to 20S:

```
cat > myregion << EOF
120 20
120 -20
270 -20
270 20
EOF

cdo selregion,myregion infile outfile
```

With the second option, predefined regions can be used via their country/state/collection codes or names. A region is specified using `dcw:<code|name>`. Codes|Names can be combined with the plus sign.

Here is an example to select the combined region Spain and Portugal with their codes:

```
cdo selregion,dcw:ES+PT infile outfile
```

or by their names:

```
cdo selregion,dcw:Spain+Portugal infile outfile
```

The ISO two-letter country codes can be found on https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2. To define a state, append the state code to the country code, e.g. USAK for Alaska. For the coordinates of a country **CDO** uses the DCW (Digital Chart of the World) dataset from [GMT](#). This dataset must be installed on the system and the environment variable `DIR_DCW` must point to it.

From DCW versions 2.1.0 onwards you can also use named regions (continents, seas, etc) using their names. DCW offers 37 geographic regions based on [Natural Earth](#) at scale 1:110, 104 seas named (102 based on [IHO1953](#) and 57 lakes, islands and archipelagoes. You can also use 46 collections of countries; 30 are from [UN49](#) and the rest were obtained from [Wikipedia](#). You can see the full list at `dcw-collections.txt`. This allows you to select specific regions, for example all the countries of Africa:

```
cdo selregion,dcw:"UN Africa" infile outfile
```

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 `infile` or `infile1`, `infile2`, etc., and an arbitrary number of input files are named `infiles`. All output files are named `outfile` or `outfile1`, `outfile2`, etc. Further the following definition is introduced:

$i(t)$

Timestep t of `infile`

$i(t, x)$

Element number x of the field at timestep t of `infile`

$o(t)$

Timestep t of `outfile`

$o(t, x)$

Element number x of the field at timestep t of `outfile`

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:

<i>Info</i>	<i>info</i>	Dataset information listed by identifier
	<i>infor</i>	Dataset information listed by name
	<i>cinfo</i>	Compact information listed by name
	<i>map</i>	Dataset information and simple map
<i>Sinfo</i>	<i>sinfo</i>	Short information listed by identifier
	<i>sinfor</i>	Short information listed by name
<i>XSinfo</i>	<i>xsinfo</i>	Extra short information listed by name
	<i>xsinfof</i>	Extra short information listed by identifier
<i>Diff</i>	<i>diff</i>	Compare two datasets listed by identifier
	<i>diffn</i>	Compare two datasets listed by name
<i>Ninfo</i>	<i>npar</i>	Number of parameters
	<i>nlevel</i>	Number of levels
	<i>nyear</i>	Number of years
	<i>nmon</i>	Number of months
	<i>ndate</i>	Number of dates
	<i>ntime</i>	Number of timesteps
	<i>ngridpoints</i>	Number of gridpoints
	<i>ngrids</i>	Number of horizontal grids
<i>Showinfo</i>	<i>showformat</i>	Show file format
	<i>showcode</i>	Show code numbers
	<i>showname</i>	Show variable names
	<i>showstdname</i>	Show standard names
	<i>showlevel</i>	Show levels
	<i>showltype</i>	Show GRIB level types
	<i>showyear</i>	Show years
	<i>showmon</i>	Show months
	<i>showdate</i>	Show date information
	<i>showtime</i>	Show time information
	<i>showtimestamp</i>	Show timestamp
	<i>showchunkspec</i>	Show chunk specification
	<i>showfilter</i>	Show filter specification
<i>Showattribute</i>	<i>showattribute</i>	Show attributes
<i>Filedes</i>	<i>partab</i>	Parameter table
	<i>codetab</i>	Parameter code table
	<i>griddes</i>	Grid description
	<i>zaxisdes</i>	Z-axis description
	<i>vct</i>	Vertical coordinate table

2.1.1 Info

Name

info, infon, cinfo, map - Information and simple statistics

Synopsis

cdo [options] <operator> *infile*s

Description

This module writes information about the structure and contents for each field of all input files to standard output. A field is a horizontal layer of a data variable. 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 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 name

The same as operator info but using the parameter name instead of the identifier to label the parameter.

cinfo

Compact information listed by name

cinfo is a compact version of **infon**. It prints the minimum, mean and maximum value for each variable across all layers and time steps.

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.

Options

-p, *--async_read true* to read input data asynchronously.

Example

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

```
cdo [options] infon infile
```

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

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

Author

Uwe Schulzweida

2.1.2 Sinfo

Name

sinfo, sinfon - Short information

Synopsis

cdo <operator> *infile*s

Description

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

Operators

sinfo

Short information listed by 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
- time c=constant v=varying
- type of statistical processing
- 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 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 infile
```

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

```
-1 : Institut Source T Steptype Levels Num Points Num Dtype : Name
 1 : MPIMET ECHAM5 c instant      1  1    2048  1  F32  : GEOSP
 2 : MPIMET ECHAM5 v instant      4  2    2048  1  F32  : T
 3 : MPIMET ECHAM5 v instant      1  1    2048  1  F32  : TSURF
Grid coordinates :
 1 : gaussian           : points=2048 (64x32) F16
                        longitude : 0 to 354.375 by 5.625 degrees_east circular
                        latitude  : 85.7606 to -85.7606 degrees_north
Vertical coordinates :
 1 : surface            : levels=1
```

(continues on next page)

(continued from previous page)

```
2 : pressure           : levels=4
                          level : 92500 to 20000 Pa
Time coordinate :
                      time : 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
```

Author

Uwe Schulzweida

2.1.3 XSinfo

Name

xsinfo, xsinfop - Extra short information

Synopsis

cdo <operator> *infile*s

Description

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

Operators

xsinfo

Extra short information listed by name

Prints extra 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
- time c=constant v=varying
- type of statistical processing
- number of levels and z-axis number
- horizontal grid size and number
- data type
- memory type (float or double)
- parameter name

Section 2 to 4 gives an extra short overview of all grid, vertical and time coordinates.

xsinfop

Extra short information listed by identifier

The same as operator *xsinfo* but using the identifier instead of the name to label the parameter.

Example

To print extra short information of a dataset use:

```
cdo xsinfo infile
```

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

```
-1 : Institut Source T Steptype Levels Num Points Num Dtype Mtype : Name
 1 : MPIMET ECHAM5 c instant      1  1   2048  1 F32  F32 : GEOSP
 2 : MPIMET ECHAM5 v instant      4  2   2048  1 F32  F32 : T
 3 : MPIMET ECHAM5 v instant      1  1   2048  1 F32  F32 : TSURF
Grid coordinates :
 1 : gaussian           : points=2048 (64x32) F16
                        longitude: 0 to 354.375 by 5.625 degrees_east circular
                        latitude: 85.7606 to -85.7606 degrees_north
Vertical coordinates :
```

(continues on next page)

(continued from previous page)

```
1 : surface      : levels=1
2 : pressure     : levels=4
                  level: 92500 to 20000 Pa
Time coordinate :
                  steps: 12
                  time: 1987-01-31T18:00:00 to 1987-12-31T18:00:00 by 1 month
                  units: days since 1987-01-01T00:00:00
                  calendar: proleptic_gregorian
```

Author

Uwe Schulzweida

2.1.4 Diff

Name

diff, diffn - Compare two datasets field by field

Synopsis

cdo [options] <operator>[.parameters] infile1 infile2

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 dimensions. Try the option names if the number of variables differ. Exit status is 0 if inputs are the same and 1 if they differ.

Operators

diff

Compare two datasets listed by identifier

Provides statistics on differences between two datasets. For each pair of fields the operator prints one line with the following information:

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

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

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

diffn

Compare two datasets listed by name

The same as operator diff. Using the name instead of the identifier to label the parameter.

Parameters

maxcount

[INTEGER] Stop after maxcount different fields.

abslim

[FLOAT] Limit of the maximum absolute difference (default: 0).

rellim

[FLOAT] Limit of the maximum relative difference (default: 1).

names

[STRING] Consideration of the variable names of only one input file (left/right) or the intersection of both (intersect).

Options

`-p`, `--async_read true` to read input data asynchronously.

Example

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

```
cdo diffn infile1 infile2
```

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

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

11 of 12 records differ

Author

Uwe Schulzweida, Karl-Hermann Wieners

2.1.5 Ninfo

Name

npar, nlevel, nyear, nmon, ndate, ntime, ngridpoints, ngrids - Print the number of parameters, levels or times

Synopsis

cdo <operator> *infile*

Description

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

Operators

npar

Number of parameters

Prints the number of parameters (variables).

nlevel

Number of levels

Prints the number of levels for each variable.

nyear

Number of years

Prints the number of different years.

nmon

Number of months

Prints the number of different combinations of years and months.

ndate

Number of dates

Prints the number of different dates.

ntime

Number of timesteps

Prints the number of timesteps.

ngridpoints

Number of gridpoints

Prints the number of gridpoints for each variable.

ngrids

Number of horizontal grids

Prints the number of horizontal grids.

Example

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

```
cdo npar infile
```

To print the number of months in a dataset use:

```
cdo nmon infile
```

Author

Uwe Schulzweida, Ralf Müller

2.1.6 Showinfo

Name

showformat, showcode, showname, showstdname, showlevel, showltype, showyear, showmon, showdate, showtime, showtimestamp, showchunkspec, showfilter - Show variable information

Synopsis

cdo <operator> *infile*

Description

This module prints meta-data information of all input variables. Depending on the chosen operator the name, level, date, time and other information is printed.

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).

showchunkspec

Show chunk specification

Prints NetCDF4 chunk specification of all variables.

showfilter

Show filter specification

Prints NetCDF4 filter specification of all variables.

Example

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

```
cdo showcode infile
```

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

```
129 130 139
```

To print all months in a dataset use:

```
cdo showmon infile
```

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

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

Author

Uwe Schulzweida

2.1.7 Showattribute

Name

showattribute - Show attributes

Synopsis

cdo showattribute[,attributes] *infile*

Description

This operator prints the attributes of the data variables of a dataset.

Each attribute has the following structure: [**var_nm**@][**att_nm**]

var_nm	Variable name (optional). Example: pressure
att_nm	Attribute name (optional). Example: units

The value of **var_nm** is the name of the variable containing the attribute (named **att_nm**) that you want to print. Use wildcards to print the attribute **att_nm** of more than one variable. A value of **var_nm** of '*' will print the attribute **att_nm** of all data variables. If **var_nm** is missing then **att_nm** refers to a global attribute.

The value of **att_nm** is the name of the attribute you want to print. Use wildcards to print more than one attribute. A value of **att_nm** of '*' will print all attributes.

Note

NetCDF attributes that are interpreted in **CDO** can't be displayed. Here is an incomplete list: formula_terms, cell_measures, coordinates, grid_mapping, valid_range, ...

Parameters

attributes

[STRING] Comma-separated list of attributes.

Author

Uwe Schulzweida

2.1.8 Filedes

Name

partab, codetab, griddes, zaxisdes, vct - Dataset description

Synopsis

cdo <operator> *infile*

Description

This module provides operators to print meta information about a dataset. The printed meta-data depends on the chosen operator.

Operators

partab

Parameter table

Prints all available meta information of the variables.

codetab

Parameter code table

Prints a code 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.

Parameters

genbounds

[BOOL] Generates cell bounds for regular LonLat grids.

Example

Assume all variables of the dataset are on a regular Gaussian F16 grid. To print the grid description of this dataset use:

```
cdo griddes infile
```

Result:

```
gridtype : gaussian
gridsize : 2048
xname    : lon
xlongname : longitude
xunits   : degrees_east
yname    : lat
```

(continues on next page)

(continued from previous page)

```
ylongname : latitude
yunits    : degrees_north
xsize     : 64
ysize     : 32
xfirst    : 0
xinc      : 5.625
yvals     : 85.76058 80.26877 74.74454 69.21297 63.67863 58.1429 52.6065
           : 47.06964 41.53246 35.99507 30.4575 24.91992 19.38223 13.84448
           : 8.306702 2.768903 -2.768903 -8.306702 -13.84448 -19.38223
           : -24.91992 -30.4575 -35.99507 -41.53246 -47.06964 -52.6065
           : -58.1429 -63.67863 -69.21297 -74.74454 -80.26877 -85.76058
```

Author

Uwe Schulzweida

2.2 File operation

This section contains modules to perform operations on files.

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

<i>Copy</i>	<i>copy</i>	Copy datasets
	<i>clone</i>	Clone datasets
	<i>cat</i>	Concatenate datasets
<i>Tee</i>	<i>tee</i>	Duplicate a data stream and write it to file
<i>Pack</i>	<i>pack</i>	Pack data
<i>Unpack</i>	<i>unpack</i>	Unpack data
<i>Setchunkspec</i>	<i>setchunkspec</i>	Specify chunking
<i>Setfilter</i>	<i>setfilter</i>	Specify filter
<i>Bitrounding</i>	<i>bitrounding</i>	Bit rounding
<i>Replace</i>	<i>replace</i>	Replace variables
<i>Duplicate</i>	<i>duplicate</i>	Duplicates a dataset
<i>Mergegrid</i>	<i>mergegrid</i>	Merge grid
<i>Merge</i>	<i>merge</i>	Merge datasets with different fields
	<i>mergetime</i>	Merge datasets sorted by date and time
<i>Split</i>	<i>splitcode</i>	Split code numbers
	<i>splitparam</i>	Split parameter identifiers
	<i>splitname</i>	Split variable names
	<i>splitlevel</i>	Split levels
	<i>splitgrid</i>	Split grids
	<i>splitzaxis</i>	Split z-axes
	<i>splittabnum</i>	Split parameter table numbers
	<i>splitensemble</i>	Split ensembles
<i>Splittime</i>	<i>splithour</i>	Split hours
	<i>splitday</i>	Split days
	<i>splitseas</i>	Split seasons
	<i>splityear</i>	Split years
	<i>splityearmon</i>	Split in years and months
	<i>splitmon</i>	Split months
<i>Splitsel</i>	<i>splitsel</i>	Split selected timesteps
<i>Splitdate</i>	<i>splitdate</i>	Splits a file into dates
<i>Distgrid</i>	<i>distgrid</i>	Distribute horizontal grid
<i>Collgrid</i>	<i>collgrid</i>	Collect horizontal grid

2.2.1 Copy

Name

copy, clone, cat - Copy datasets

Synopsis

cdo <operator> *infile* *outfile*

Description

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

Operators

copy

Copy datasets

Copies all input datasets to *outfile*.

clone

Clone datasets

Copies all input datasets to *outfile*. In contrast to the copy operator, clone tries not to change the input data. GRIB records are neither decoded nor decompressed.

cat

Concatenate datasets

Concatenates all input datasets and appends the result to the end of *outfile*. If *outfile* does not exist it will be created.

Example

To change the format of a dataset to NetCDF use:

```
cdo -f nc copy infile outfile.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 infile outfile.nc
```

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

```
cdo copy infile1 infile2 infile3 outfile
```

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

```
cdo cat infile1 infile2 infile3 outfile
```

Author

Uwe Schulzweida

2.2.2 Tee

Name

tee - Duplicate a data stream and write it to file

Synopsis

cdo tee,outfile2 infile outfile1

Description

This operator copies the input dataset to **outfile1** and **outfile2**. The first output stream in **outfile1** can be further processed with other cdo operators. The second output **outfile2** is written to disk. It can be used to store intermediate results to a file.

Parameters

outfile2

[STRING] Destination filename for the copy of the input file

Example

To compute the daily and monthly average of a dataset use:

```
cdo monavg -tee,outfile_dayavg dayavg infile outfile_monavg
```

Author

Uwe Schulzweida

2.2.3 Pack

Name

pack - Pack data

Synopsis

cdo pack[,*parameter*] *infile outfile*

Description

Packing reduces the data volume by reducing the precision of the stored numbers. It is implemented using the NetCDF attributes `add_offset` and `scale_factor`. The operator **pack** calculates the attributes `add_offset` and `scale_factor` for all variables. The default data type for all variables is automatically changed to 16-bit integer. Use the **CDO** option `-b` to change the data type to a different integer precision, if needed. Missing values are automatically transformed to the current data type.

Alternatively, the pack parameters `add_offset` and `scale_factor` can be read from a file for each variable.

Parameters

printparam

[BOOL] Print pack parameters to stdout for each variable

filename

[STRING] Read pack parameters from file for each variable[format: name=<> add_offset=<>
scale_factor=<>]

Author

Uwe Schulzweida

2.2.4 Unpack

Name

unpack - Unpack data

Synopsis

cdo unpack *infile outfile*

Description

Packing reduces the data volume by reducing the precision of the stored numbers. It is implemented using the NetCDF attributes `add_offset` and `scale_factor`. The operator **unpack** unpack all packed variables. The default data type for all variables is automatically changed to 32-bit floats. Use the **CDO** option -b F64 to change the data type to 64-bit floats, if needed.

Author

Uwe Schulzweida

2.2.5 Setchunkspec

Name

setchunkspec - Specify chunking

Synopsis

cdo setchunkspec,*parameter infile outfile*

Description

Specify chunking for selected variables in the output. Chunking is available for NetCDF4 and useful to specify the units of disk access and compression. The filename parameter is used to specify the file which contains the chunk specification for each variable. The chunkspec argument is a comma-separated string with the chunk size for the dimensions x,y,z,t. A chunkspec must name at least one dimension, e.g. t=<chunksize> to set the chunk size of the time dimension to <chunksize>.

Use the **CDO** option --chunkspec instead of setchunkspec if all variables require the same chunks.

Parameters

filename

[STRING] Name of the file containing the chunk specification per variable [format: varname=<chunkspec>]

Author

Uwe Schulzweida

2.2.6 Setfilter

Name

setfilter - Specify filter

Synopsis

cdo setfilter,*parameter infile outfile*

Description

Specify filter for selected variables in the output. Filters are available for NetCDF4 and mainly used to compress/decompress data. NetCDF4 uses the HDF5 plugins for filter support. To find the HDF5 plugins, the environment variable HDF5_PLUGIN_PATH must point to the directory with the installed filter plugins. The program may terminate unexpectedly if filters are used whose plugins are not found.

The filename parameter is used to specify the file which contains the filter specification for each variable. A filter specification consists of the filterId and the filter parameters. **CDO** supports multiple filters connected with '|'. Here is a filter specification for bzip2 (filterId: 307) combined with szip (filterId:4): "307,9|4,32,32".

Use the **CDO** option --filter instead of setfilter if all variables require the same filter. More information about NetCDF4 filters can be found in <https://docs.unidata.ucar.edu/netcdf-c/current/filters.html>.

Parameters

filename

[STRING] Name of the file containing the filter specification per variable [format: varname=<filterspec>]

Author

Uwe Schulzweida

2.2.7 Bitrounding

Name

bitrounding - Bit rounding

Synopsis

cdo bitrounding[*,parameter*] *infile outfile*

Description

This operator calculates for each field the number of necessary mantissa bits to get a certain information level in the data. With this number of significant bits (`numbits`) a rounding of the data is performed. This allows the data to be compressed to a higher level.

The default value of the information level is 0.9999 and can be adjusted with the parameter `inflevel`. That means 99.99% of the information in the mantissa bits is preserved.

Alternatively, the number of significant bits can be set for all variables with the `numbits` parameter. Furthermore, `numbits` can be assigned for each variable via the filename parameter. In this case, `numbits` is still calculated for all variables if they are not present in the file.

The analysis of the bit information is based on the Julia library [BitInformation.jl](#). The procedure to derive the number of significant mantissa bits was adapted from the Python library [xbitinfo](#). Quantize to the number of mantissa bits is done with IEEE rounding using code from NetCDF 4.9.0.

Currently only 32-bit float data is rounded. Data with missing values are not yet supported for the calculation of significant bits.

Parameters

inflevel

[FLOAT] Information level (0 - 1) [default: 0.9999]

addbits

[INTEGER] Add bits to the number of significant bits [default: 0]

minbits

[INTEGER] Minimum value of the number of bits [default: 1]

maxbits

[INTEGER] Maximum value of the number of bits [default: 23]

numsteps

[INTEGER] Set to 1 to run the calculation only in the first time step

numbits

[INTEGER] Set number of significant bits

printbits

[BOOL] Print max. numbits per variable of 1st timestep to stdout [format: name=numbits]

filename

[STRING] Read number of significant bits per variable from file [format: name=numbits]

Example

Apply bit rounding to all 32-bit float fields, preserving 99.9% of the information, followed by compression and storage to NetCDF4:

```
cdo -f nc4 -z zip bitrounding,inflevel=0.999 infile outfile
```

Add the option `-v` to view used number of mantissa bits for each field:

```
cdo -v -f nc4 -z zip bitrounding,inflevel=0.999 infile outfile
```

Author

Uwe Schulzweida

2.2.8 Replace

Name

replace - Replace variables

Synopsis

cdo replace *infile1 infile2 outfile*

Description

This operator replaces variables in *infile1* by variables from *infile2* and write the result to *outfile*. Both input datasets need to have the same number of timesteps. All variable names may only occur once!

Example

Assume the first input dataset *infile1* has three variables with the names *geosp*, *t* and *tslm1* and the second input dataset *infile2* has only the variable *tslm1*. To replace the variable *tslm1* in *infile1* by *tslm1* from *infile2* use:

```
cdo replace infile1 infile2 outfile
```

Author

Uwe Schulzweida

2.2.9 Duplicate

Name

duplicate - Duplicates a dataset

Synopsis

```
cdo duplicate[,parameter] infile outfile
```

Description

This operator duplicates the contents of **infile** and writes the result to **outfile**. The optional parameter sets the number of duplicates, the default is 2.

Parameters

ndup

[INTEGER] Number of duplicates, default is 2.

Author

Uwe Schulzweida

2.2.10 Mergegrid

Name

mergegrid - Merge grid

Synopsis

cdo mergegrid *infile1 infile2 outfile*

Description

Merges grid points of all variables from **infile2** to **infile1** and write the result to **outfile**. Only the non missing values of **infile2** will be used. The horizontal grid of **infile2** should be smaller or equal to the grid of **infile1** and the resolution must be the same. Only rectilinear grids are supported. Both input files need to have the same variables and the same number of timesteps.

Author

Uwe Schulzweida

2.2.11 Merge

Name

merge, mergetime - Merge datasets

Synopsis

cdo [options] merge *infile* *outfile*

cdo [options] mergetime[,*parameters*] *infile* *outfile*

Description

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

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 *outfile* 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 *outfile* and all timesteps are sorted by date and time.

Parameters

skip_same_time

[BOOL] Skips all consecutive timesteps with a double entry of the same timestamp.

names

[STRING] Fill missing variable names with missing values (union) or use the intersection (intersect).

Options

-0, *--overwrite* to overwrite existing output file.

Note

Operators of this module need to open all input files simultaneously. The maximum number of open files depends on the operating system!

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 infile1 infile2 infile3 outfile
```

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 infile1 infile2 infile3 infile4 outfile
```

Author

Uwe Schulzweida

2.2.12 Split

Name

splitcode, splitparam, splitname, splitlevel, splitgrid, splitzaxis, splittabnum, splitensemble - Split a dataset

Synopsis

cdo <operator>[,parameters] *infile* *obase*

Description

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

Operators

splitcode

Split code numbers

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

splitparam

Split parameter identifiers

Splits a dataset into pieces, one for each different parameter identifier. *xxx* will be a string with the parameter identifier.

splitname

Split variable names

Splits a dataset into pieces, one for each variable name. *xxx* will be a string with the variable name.

splitlevel

Split levels

Splits a dataset into pieces, one for each different level. *xxx* will have six digits with the level.

splitgrid

Split grids

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

splitzaxis

Split z-axes

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

splittabnum

Split parameter table numbers

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

splitensemble

Split ensembles

Splits a dataset into pieces, one for each GRIB2 ensemble member. *xxx* will have five digits with the GRIB2 key *perturbationNumber*.

Parameters

swap

[STRING] Swap the position of obase and xxx in the output filename

uuid=<attname>

[STRING] Add a UUID as global attribute <attname> to each output file

Environment

CDO_FILE_SUFFIX sets the filename suffix.

Note

Operators of this module need to open all output files simultaneously. The maximum number of open files depends on the operating system!

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 infile code
```

Result of `dir code*`:

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

Author

Uwe Schulzweida

2.2.13 Splittime

Name

splithour, splitday, splitseas, splityear, splityearmon, splitmon - Split timesteps of a dataset

Synopsis

cdo <operator> *infile* *obase*

Description

This module splits *infile* into timesteps pieces. The output files will be named <obase><xxx><suffix> where :file: `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.
Use the optional format parameter to change the format for the month.

splitseas

Split seasons

Splits a file into pieces, one for each different season. **xxx** will have three characters with the season.

splityearmon

Split in years and months

Splits a file into pieces, one for each different year and month. **xxx** will have six digits with the year and month (YYYYMM).

splityear

Split years

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

Parameters

format

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

Environment

CDO_FILE_SUFFIX sets the filename suffix.

Note

Operators of this module need to open all output files simultaneously. The maximum number of open files depends on the operating system!

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 infile 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
```

Author

Uwe Schulzweida

2.2.14 Splitsel

Name

splitsel - Split selected timesteps

Synopsis

cdo splitsel,*parameters infile obase*

Description

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

Parameters

nsets

[INTEGER] Number of input timesteps for each output file

noffset

[INTEGER] Number of input timesteps skipped before the first timestep range (optional)

nskip

[INTEGER] Number of input timesteps skipped between timestep ranges (optional)

Environment

CDO_FILE_SUFFIX sets the filename suffix.

Author

Etienne Tourigny

2.2.15 Splitdate

Name

splitdate - Splits a file into dates

Synopsis

cdo splitdate *infile* *obase*

Description

This operator splits *infile* into pieces, one for each different date. The output files will be named *<obase><YYYY-MM-DD><suffix>* where *YYYY-MM-DD* is the date and *suffix* is the filename extension derived from the file format.

Environment

CDO_FILE_SUFFIX sets the filename suffix.

Author

Uwe Schulzweida

2.2.16 Distgrid

Name

distgrid - Distribute horizontal grid

Synopsis

cdo distgrid,*parameters infile obase*

Description

This operator distributes a dataset into smaller pieces. Each output file contains a different region of the horizontal source grid. 2D Lon/Lat grids can be split into **nx*****ny** pieces, where a target grid region contains a structured longitude/latitude box of the source grid. Data on an unstructured grid is split into **nx** pieces. The output files will be named **<obase><xxx><suffix>** where suffix is the filename extension derived from the file format. **xxx** will have five digits with the number of the target region.

Parameters

nx

[INTEGER] Number of regions in x direction, or number of pieces for unstructured grids

ny

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

Note

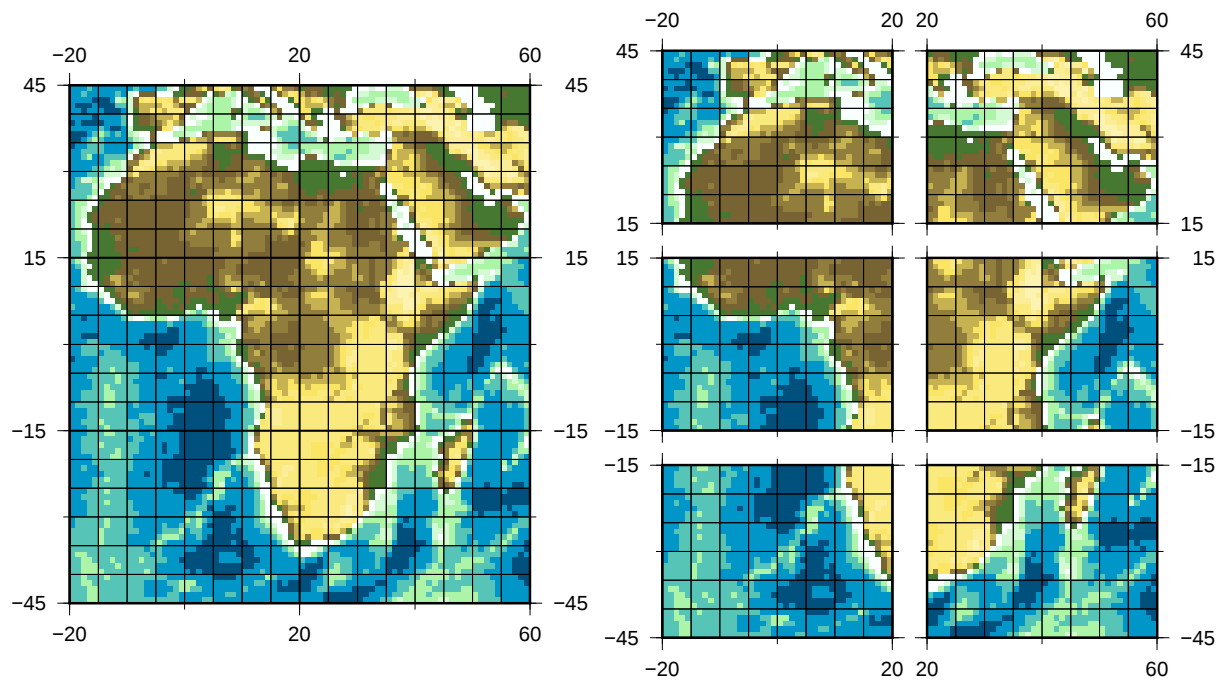
This operator needs to open all output files simultaneously. The maximum number of open files depends on the operating system!

Example

Distribute data on a 2D Lon/Lat grid into 6 smaller files, each output file receives one half of x and a third of y of the source grid:

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

Below is a schematic illustration of this example:



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

Author

Uwe Schulzweida

2.2.17 Collgrid

Name

collgrid - Collect horizontal grid

Synopsis

cdo collgrid,*parameters infile obase*

Description

This operator collects the data of the input files to one output file. All input files need to have the same variables and the same number of timesteps on a different horizontal grid region. If the source regions are on a structured lon/lat grid, all regions together must result in a new structured lat/long grid box. Data on an unstructured grid are concatenated in the order of the input files. For ICON restart data, the array `global_cell_indices` is used for indexing if it is available. The parameter `nx` needs to be specified only for curvilinear grids.

Parameters

nx

[INTEGER] Number of regions in x direction [default: number of input files]

name

[STRING] Comma-separated list of variable names.

levidx

[INTEGER] Comma-separated list or first/last[/inc] range of index of levels.

gridtype

[STRING] For unstructured grids, set to unstructured.

Note

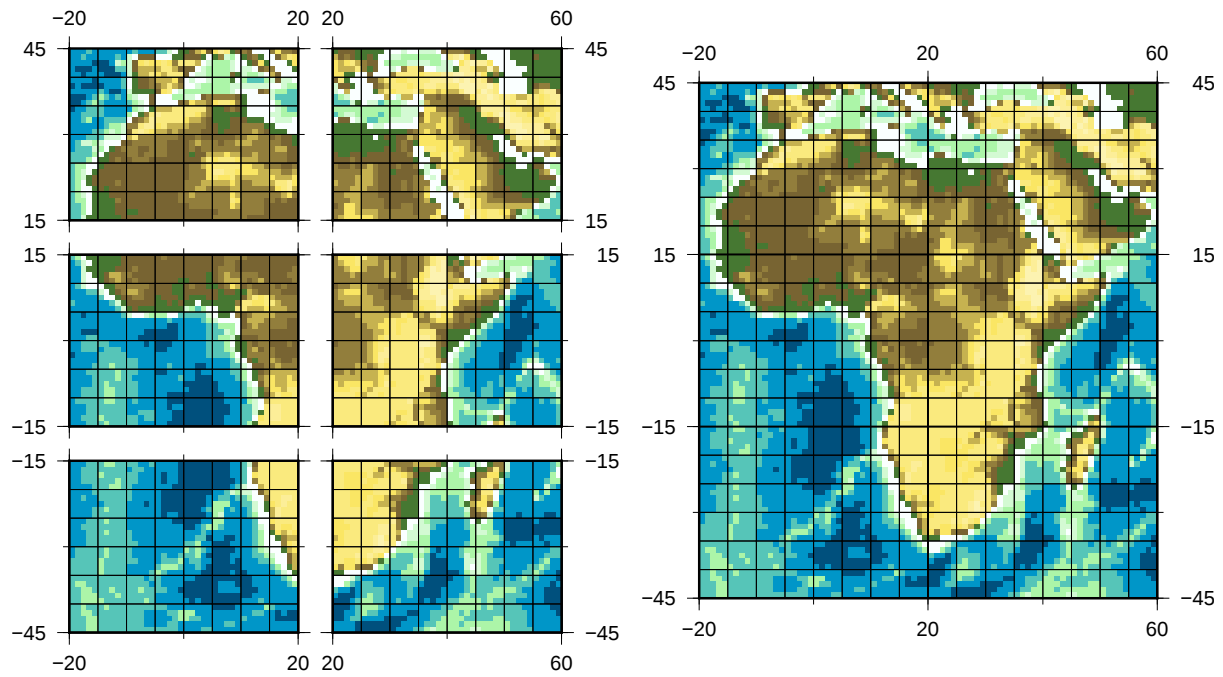
This operator needs to open all input files simultaneously. The maximum number of open files depends on the operating system!

Example

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

```
cdo collgrid infile[1-6] outfile
```

Below is a schematic illustration of this example:



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

Author

Uwe Schulzweida

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:

<i>Query</i>	<i>query</i>	Pre-select a subset of a dataset
<i>Select</i>	<i>select</i>	Select fields
	<i>delete</i>	Delete fields
<i>Selmulti</i>	<i>selmulti</i>	Select multiple fields
	<i>delmulti</i>	Delete multiple fields
	<i>changemulti</i>	Change identification of multiple fields
<i>Selvar</i>	<i>selparam</i>	Select parameters by identifier
	<i>delparam</i>	Delete parameters by identifier
	<i>selcode</i>	Select parameters by code number
	<i>delcode</i>	Delete parameters by code number
	<i>selname</i>	Select parameters by name
	<i>delname</i>	Delete parameters by name
	<i>selstdname</i>	Select parameters by standard name
	<i>sellevel</i>	Select levels
	<i>sellevelidx</i>	Select levels by index
	<i>selgrid</i>	Select grids
	<i>selzaxis</i>	Select z-axes
	<i>selzaxisname</i>	Select z-axes by name
	<i>selltype</i>	Select GRIB level types
	<i>seltabnum</i>	Select parameter table numbers
<i>Seltime</i>	<i>selimestep</i>	Select timesteps
	<i>seltime</i>	Select times
	<i>selhour</i>	Select hours
	<i>selday</i>	Select days
	<i>selmonth</i>	Select months
	<i>selyear</i>	Select years
	<i>selseason</i>	Select seasons
	<i>seldate</i>	Select dates
	<i>selsmon</i>	Select single month
<i>Selbox</i>	<i>sellonlatbox</i>	Select a longitude/latitude box
	<i>selindexbox</i>	Select an index box
<i>Selregion</i>	<i>selregion</i>	Select cells inside regions
	<i>selcircle</i>	Select cells inside a circle
<i>Selgridcell</i>	<i>selgridcell</i>	Select grid cells
	<i>delgridcell</i>	Delete grid cells
<i>Samplegrid</i>	<i>samplegrid</i>	Resample grid cells
<i>Selyearidx</i>	<i>selyearidx</i>	Select year by index
<i>Seltimeidx</i>	<i>seltimeidx</i>	Select timestep by index
<i>Selsurface</i>	<i>bottomvalue</i>	Extract bottom level
	<i>topvalue</i>	Extract top level
	<i>isosurface</i>	Extract isosurface

2.3.1 Query

Name

query - Pre-select a subset of a dataset

Synopsis

cdo query,parameters infile outfile

Description

CDO supports the NetCDF classic data model with attributes that conform to the CF conventions. The **CDO** I/O interface always reads in a complete horizontal field. If the dataset is chunked and the chunks extend beyond the horizontal field, the chunks are cached.

The query operator allows you to reduce the view to the dataset before **CDO** processes it. This can improve performance and reduce memory requirements – particularly for high-resolution, chunked and compressed datasets, especially when the data has to be loaded over the network. The size of the required cache is also affected by this.

The following keys are available for pre-selecting a subset of a dataset:

Keyword	Description
group	Select a group (group=groupName)
name	Comma-separated list of variable names (name=var1,var2,...)
cell	Cell index range (cell=first[/to/last])
layer	Layer index range (layer=first[/to/last])
step	Time step index range (step=first[/to/last])
startdate	Start date (format: YYYY-MM-DD[Thh:mm:ss])
enddate	End date (format: YYYY-MM-DD[Thh:mm:ss])

The indices for keys **cell**, **layer** and **step** start at 1. The **cell** key is only implemented for unstructured and HEALPix grids.

Parameters

group

[STRING] Select a group (group=groupName)

name

[STRING] Comma-separated list of variable names (name=var1,var2,...)

cell

[STRING] Cell index range (cell=first[/to/last])

layer

[STRING] Layer index range (layer=first[/to/last])

step

[STRING] Time step index range (step=first[/to/last])

startdate

[STRING] Start date (format: YYYY-MM-DD[Thh:mm:ss])

enddate

[STRING] End date (format: YYYY-MM-DD[Thh:mm:ss])

Note

The order of the input fields can't be changed by selecting them.

Example

To pre-select the first grid cell of the the first 100 time steps of all variables, use:

```
cdo query,cell=1,step=1/to/100 infile outfile
```

Author

Uwe Schulzweida

2.3.2 Select

Name

select, delete - Select fields

Synopsis

cdo <operator>,*parameters infiles outfile*

Description

This module selects some fields from **infiles** and writes them to **outfile**. **infiles** is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. The fields selected depends on the chosen parameters. Parameter is a comma-separated list of "key=value" pairs. A range of integer values can be specified by first/last[/inc]. Wildcards are supported for string values.

Operators

select

Select fields

Selects all fields with parameters in a user given list.

delete

Delete fields

Deletes all fields with parameters in a user given list.

Parameters

name

[STRING] Comma-separated list of variable names.

param

[STRING] Comma-separated list of parameter identifiers.

code

[INTEGER] Comma-separated list or first/last[/inc] range of code numbers.

level

[FLOAT] Comma-separated list of vertical levels.

levrange

[FLOAT] First and last value of the level range.

levidx

[INTEGER] Comma-separated list or first/last[/inc] range of index of levels.

zaxisname

[STRING] Comma-separated list of zaxis names.

zaxisnum

[INTEGER] Comma-separated list or first/last[/inc] range of zaxis numbers.

ltype

[INTEGER] Comma-separated list or first/last[/inc] range of GRIB level types.

gridname

[STRING] Comma-separated list of grid names.

gridnum

[INTEGER] Comma-separated list or first/last[/inc] range of grid numbers.

steptype

[STRING] Comma-separated list of timestep types (constant|avg|accum|min|max|range|diff|sum)

date

[STRING] Comma-separated list of dates (format: YYYY-MM-DDThh:mm:ss).

startdate

[STRING] Start date (format: YYYY-MM-DDThh:mm:ss).

enddate

[STRING] End date (format: YYYY-MM-DDThh:mm:ss).

minute

[INTEGER] Comma-separated list or first/last[/inc] range of minutes.

hour

[INTEGER] Comma-separated list or first/last[/inc] range of hours.

day

[INTEGER] Comma-separated list or first/last[/inc] range of days.

month

[INTEGER] Comma-separated list or first/last[/inc] range of months.

season

[STRING] Comma-separated list of seasons (substring of DJFMAMJJASOND or ANN).

year

[INTEGER] Comma-separated list or first/last[/inc] range of years.

dom

[STRING] Comma-separated list of the day of month (e.g. 29feb).

timestep

[INTEGER] Comma-separated list or first/last[/inc] range of timesteps. Negative values select timesteps from the end (NetCDF only).

timestep_of_year

[INTEGER] Comma-separated list or first/last[/inc] range of timesteps of year.

timestepmask

[STRING] Read timesteps from a mask file.

Note

The order of the input fields can't be changed by selecting them.

Example

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

```
cdo select,name=T,U,V,level=200,500,850 infile1 infile2 infile3 outfile
```

To remove the February 29th use:

```
cdo delete,dom=29feb infile outfile
```

Author

Uwe Schulzweida

2.3.3 Selmulti

Name

selmulti, delmulti, changemulti - Select multiple fields via GRIB1 parameters

Synopsis

cdo <operator>.parameters infile outfile

Description

This module selects multiple fields from **infile** and writes them to **outfile**. selection-specification is a filename or in-place string with the selection specification. Each selection-specification has the following compact notation format:

```
<type>(parameters; leveltype(s); levels)
```

type : sel for select or del for delete (optional)

parameters : GRIB1 parameter code number

leveltype : GRIB1 level type

levels : value of each level

Examples:

```
(1; 103; 0)
(33,34; 105; 10)
(11,17; 105; 2)
(71,73,74,75,61,62,65,117,67,122,121,11,131,66,84,111,112; 105; 0)
```

The following descriptive notation can also be used for selection specification from a file:

```
SELECT/DELETE, PARAMETER=parameters, LEVTYPE=leveltye(s), LEVEL=levels
```

Examples:

```
SELECT, PARAMETER=1, LEVTYPE=103, LEVEL=0
SELECT, PARAMETER=33/34, LEVTYPE=105, LEVEL=10
SELECT, PARAMETER=11/17, LEVTYPE=105, LEVEL=2
SELECT, PARAMETER=71/73/74/75/61/62/65/117/67/122, LEVTYPE=105, LEVEL=0
DELETE, PARAMETER=128, LEVTYPE=109, LEVEL=*
```

The following will convert Pressure from Pa into hPa; Temp from Kelvin to Celsius:

```
SELECT, PARAMETER=1, LEVTYPE= 103, LEVEL=0, SCALE=0.01
SELECT, PARAMETER=11, LEVTYPE=105, LEVEL=2, OFFSET=273.15
```

If SCALE and/or OFFSET are defined, then the data values are scaled as SCALE*(VALUE-OFFSET).

Operators

selmulti

Select multiple fields

delmulti

Delete multiple fields

changemulti

Change identification of multiple fields

Example

Change ECMWF GRIB code of surface pressure to Hirlam notation:

```
cdo changemulti,'{(134;1;*|1;105;*)}' infile outfile
```

Author

Michal Koutek

2.3.4 Selvar

Name

selparam, delparam, selcode, delcode, selname, delname, selstdname, sellevel, sellevidx, selgrid, selzaxis, selzaxisname, selltype, seltabnum - Select fields

Synopsis

cdo <operator>,*parameters infile outfile*

Description

This module selects some fields from **infile** and writes them to **outfile**. The fields selected depends on the chosen operator and the parameters. A range of integer values can be specified by first/last[/inc].

Operators

selparam

Select parameters by identifier

Selects all fields with parameter identifiers in a user given list (Parameter: identifiers).

delparam

Delete parameters by identifier

Deletes all fields with parameter identifiers in a user given list (Parameter: identifiers).

selcode

Select parameters by code number

Selects all fields with code numbers in a user given list or range (Parameter: codes).

delcode

Delete parameters by code number

Deletes all fields with code numbers in a user given list or range (Parameter: codes).

selname

Select parameters by name

Selects all fields with parameter names in a user given list (Parameter: names).

delname

Delete parameters by name

Deletes all fields with parameter names in a user given list (Parameter: names).

selstdname

Select parameters by standard name

Selects all fields with standard names in a user given list (Parameter: stdnames).

sellevel

Select levels

Selects all fields with levels in a user given list (Parameter: levels).

sellevidx

Select levels by index

Selects all fields with index of levels in a user given list or range (Parameter: levidx).

selgrid

Select grids

Selects all fields with grids in a user given list (Parameter: grids).

selzaxis

Select z-axes

Selects all fields with z-axes in a user given list (Parameter: zaxes).

selzaxisname

Select z-axes by name

Selects all fields with z-axis names in a user given list (Parameter: zaxisnames).

selltype

Select GRIB level types

Selects all fields with GRIB level type in a user given list or range (Parameter: ltypes).

seltabnum

Select parameter table numbers

Selects all fields with parameter table numbers in a user given list or range (Parameter: tabnums).

Parameters**identifier**

[STRING] Comma-separated list of parameter identifiers.

codes

[INTEGER] Comma-separated list or first/last[/inc] range of code numbers.

names

[STRING] Comma-separated list of variable names.

stdnames

[STRING] Comma-separated list of standard names.

levels

[FLOAT] Comma-separated list of vertical levels.

levidx

[INTEGER] Comma-separated list or first/last[/inc] range of index of levels.

ltypes

[INTEGER] Comma-separated list or first/last[/inc] range of GRIB level types.

grids

[STRING] Comma-separated list of grid names or numbers.

zaxes

[STRING] Comma-separated list of z-axis types or numbers.

zaxisnames

[STRING] Comma-separated list of z-axis names.

tabnums

[INTEGER] Comma-separated list or range of parameter table numbers.

Note

The order of the input fields can't be changed by selecting them.

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 infile outfile
```

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

```
cdo delcode,130 infile outfile
```

Author

Uwe Schulzweida

2.3.5 Selttime

Name

seltimestep, selttime, selhour, selday, selmonth, selyear, selseason, seldate, selsmon - Select timesteps

Synopsis

cdo <operator>.parameters *infile outfile*

Description

This module selects user specified timesteps from **infile** and writes them to **outfile**. The timesteps selected depends on the chosen operator and the parameters. A range of integer values can be specified by first/last[/inc].

Operators

seltimestep

Select timesteps

Selects all timesteps with a timestep in a user given list or range (Parameter: timesteps).

selttime

Select times

Selects all timesteps with a time in a user given list or range (Parameter: times).

selhour

Select hours

Selects all timesteps with an hour in a user given list or range (Parameter: hours).

selday

Select days

Selects all timesteps with a day in a user given list or range (Parameter: days).

selmonth

Select months

Selects all timesteps with a month in a user given list or range (Parameter: months).

selyear

Select years

Selects all timesteps with a year in a user given list or range (Parameter: years).

selseason

Select seasons

Selects all timesteps with a month of a season in a user given list (Parameter: seasons).

seldate

Select dates

Selects all timesteps with a date in a user given range (Parameter: startdate [enddate]).

selsmon

Select single month

Selects a month and optional an arbitrary number of timesteps before and after this month (Parameter: month [nts1] [nts2]).

Parameters

timesteps

[INTEGER] Comma-separated list or first/last[/inc] range of timesteps. Negative values select timesteps from the end (NetCDF only).

times

[STRING] Comma-separated list of times (format hh:mm:ss).

hours

[INTEGER] Comma-separated list or first/last[/inc] range of hours.

days

[INTEGER] Comma-separated list or first/last[/inc] range of days.

months

[INTEGER] Comma-separated list or first/last[/inc] range of months.

years

[INTEGER] Comma-separated list or first/last[/inc] range of years.

seasons

[STRING] Comma-separated list of seasons (substring of DJFMAMJJASOND or ANN).

startdate

[STRING] Start date (format: YYYY-MM-DDThh:mm:ss).

enddate

[STRING] End date (format: YYYY-MM-DDThh:mm:ss) [default: startdate].

nts1

[INTEGER] Number of timesteps before the selected month [default: 0].

nts2

[INTEGER] Number of timesteps after the selected month [default: nts1].

Author

Uwe Schulzweida

2.3.6 Selbox

Name

`sellonlatbox`, `selindexbox` - Select a box

Synopsis

cdo `sellonlatbox,lon1,lon2,lat1,lat2 infile outfile`

cdo `selindexbox,idx1,idx2,idy1,idy2 infile outfile`

Description

Selects grid cells inside a lon/lat or index box.

Operators

sellonlatbox

Select a longitude/latitude box

Selects grid cells inside a lon/lat box. The user must specify the longitude and latitude of the edges of the box. Only those grid cells are considered whose grid center lies within the lon/lat box. For rotated lon/lat grids the parameters must be specified in rotated coordinates.

selindexbox

Select an index box

Selects grid cells within an index box. The user must specify the indices of the edges of the box. The index of the left edge can be greater than the one of the right edge. Use negative indexing to start from the end. The input grid must be a regular lon/lat or a 2D curvilinear grid.

Parameters

lon1

[FLOAT] Western longitude in degrees

lon2

[FLOAT] Eastern longitude in degrees

lat1

[FLOAT] Southern or northern latitude in degrees

lat2

[FLOAT] Northern or southern latitude in degrees

idx1

[INTEGER] Index of first longitude (1 - nlon)

idx2

[INTEGER] Index of last longitude (1 - nlon)

idy1

[INTEGER] Index of first latitude (1 - nlat)

idy2

[INTEGER] Index of last latitude (1 - nlat)

Example

To select the region with the longitudes from 30W to 60E and latitudes from 30N to 80N from all input fields use:

```
cdo sellonlatbox,-30,60,30,80 infile outfile
```

If the input dataset has fields on a regular Gaussian F16 grid, the same box can be selected with [selindexbox](#) by:

```
cdo selindexbox,60,11,3,11 infile outfile
```

Author

Uwe Schulzweida

2.3.7 Selregion

Name

selregion, selcircle - Select horizontal regions

Synopsis

cdo selregion,*regions infile outfile*

cdo selcircle,*parameters infile outfile*

Description

Selects all grid cells with the center point inside user defined regions or a circle. The resulting grid is unstructured.

Operators

selregion

Select cells inside regions

Selects all grid cells with the center point inside the regions. Regions can be defined by the user via an ASCII file. Each region consists of the geographic coordinates of 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 &.

Predefined regions of countries can be specified via the country codes. A country is specified with dcw:<CountryCode>. Country codes can be combined with the plus sign.

selcircle

Select cells inside a circle

Selects all grid cells with the center point inside a circle. The circle is described by geographic coordinates of the center and the radius of the circle (Parameter: lon, lat, radius).

Parameters

regions

[STRING] Comma-separated list of ASCII formatted files with different regions

lon

[FLOAT] Longitude of the center of the circle in degrees, default lon=0.0

lat

[FLOAT] Latitude of the center of the circle in degrees, default lat=0.0

radius

[STRING] Radius of the circle, default radius=1deg (units: deg, rad, km, m)

Example

To select all grid cells of a country use the country code with data from the Digital Chart of the World. Here is an example for Spain with the country code ES:

```
cdo selregion,dcw:ES infile outfile
```

Author

Uwe Schulzweida

2.3.8 Selgridcell

Name

selgridcell, delgridcell - Select grid cells

Synopsis

cdo <operator>, *indices infile outfile*

Description

The operator selects grid cells of all fields from **infile**. The user must specify the index of each grid cell. The resulting grid in **outfile** is unstructured.

Operators

selgridcell

Select grid cells

delgridcell

Delete grid cells

Parameters

indices

[INTEGER] Comma-separated list or first/last[/inc] range of indices

Author

Uwe Schulzweida

2.3.9 Samplegrid

Name

samplegrid - Resample grid cells

Synopsis

cdo samplegrid,*factor infile outfile*

Description

This is a special operator for resampling the horizontal grid. No interpolation takes place. Resample factor=2 means every second grid point is removed. Only rectilinear and curvilinear source grids are supported by this operator.

Parameters

factor

[INTEGER] Resample factor, typically 2, which will half the resolution

Author

Michal Koutek

2.3.10 Selyearidx

Name

selyearidx - Select year by index

Synopsis

cdo selyearidx *infile1 infile2 outfile*

Description

Selects field elements from **infile2** according to a year index from **infile1**. The index of the year in **infile1** should be the result of corresponding *yearminidx* or *yearmaxidx* operations, respectively.

Author

Uwe Schulzweida

2.3.11 Seltimeidx

Name

seltimeidx - Select timestep by index

Synopsis

cdo seltimeidx *infile1 infile2 outfile*

Description

Selects field elements from **infile2** according to a timestep index from **infile1**. The index of the timestep in **infile1** should be the result of corresponding *timminidx* or *timmaxidx* operations, respectively.

Author

Uwe Schulzweida

2.3.12 Selsurface

Name

bottomvalue, topvalue, isosurface - Extract surface

Synopsis

cdo <operator> *infile outfile*

cdo isosurface,*isovalue* *infile outfile*

Description

This module computes a surface from all 3D variables. The result is a horizontal 2D field.

Operators

isosurface

Extract isosurface

This operator computes an isosurface. The value of the isosurface is specified by the parameter *isovalue*. The isosurface is calculated by linear interpolation between two layers.

bottomvalue

Extract bottom level

This operator selects the valid values at the bottom level. The NetCDF CF compliant attribute *positive* is used to determine where top and bottom are. If this attribute is missing, low values are bottom and high values are top.

topvalue

Extract top level

This operator selects the valid values at the top level. The NetCDF CF compliant attribute *positive* is used to determine where top and bottom are. If this attribute is missing, low values are bottom and high values are top.

Parameters

isovalue

[FLOAT] Isosurface value

Author

Uwe Schulzweida

2.4 Conditional selection

This section contains modules to conditionally 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:

<i>Cond</i>	<i>ifthen</i>	If then
	<i>ifnotthen</i>	If not then
<i>Cond2</i>	<i>ifthenelse</i>	Conditional selection
<i>Condc</i>	<i>ifthenc</i>	If then constant
	<i>ifnotthenc</i>	If not then constant
<i>Mapreduce</i>	<i>reducegrid</i>	Reduce fields to user-defined mask

2.4.1 Cond

Name

ifthen, ifnotthen - Conditional selection

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module selects field elements from **infile2** with respect to **infile1** and writes them to **outfile**. The fields in **infile1** 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 **infile1** has either to be the same as in **infile2** or the same as in one timestep of **infile2** or only one. The fields in **outfile** inherit the meta data from **infile2**.

Operators

ifthen

If then

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

ifnotthen

If not then

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

Example

To select all field elements of **infile2** if the corresponding field element of **infile1** is greater than 0 use:

```
cdo ifthen infile1 infile2 outfile
```

Author

Uwe Schulzweida

2.4.2 Cond2

Name

ifthenelse - Conditional selection

Synopsis

cdo ifthenelse *infile1 infile2 infile3 outfile*

Description

This operator selects field elements from *infile2* or *infile3* with respect to *infile1* and writes them to *outfile*. The fields in *infile1* 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 *infile1* has either to be the same as in *infile2* or the same as in one timestep of *infile2* or only one. *infile2* and *infile3* need to have the same number of fields. The fields in *outfile* inherit the meta data from *infile2*.

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

Example

To select all field elements of *infile2* if the corresponding field element of *infile1* is greater than 0 and from *infile3* otherwise use:

```
cdo ifthenelse infile1 infile2 infile3 outfile
```

Author

Uwe Schulzweida

2.4.3 Condc

Name

ifthenc, ifnotthenc - Conditional selection

Synopsis

cdo <operator>,c infile outfile

Description

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

Operators

ifthenc

If then constant

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

ifnotthenc

If not then constant

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

Parameters

c

[FLOAT] Constant

Example

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

```
cdo ifthenc,7 infile outfile
```

Author

Uwe Schulzweida

2.4.4 Mapreduce

Name

reducegrid - Reduce fields to user-defined mask

Synopsis

cdo reducegrid,*parameters infile outfile*

Description

This module holds an operator for data reduction based on a user defined mask. The output grid is unstructured and includes coordinate bounds. Bounds can be avoided by using the additional 'nobounds' keyword. With 'nocoords' given, coordinates are completely suppressed.

Operators

reducegrid

Reduce fields to user-defined mask

Reduce input file variables to locations, where mask is non-zero. Horizontal grids of **mask** and **infile** must be identical.

Parameters

mask

[STRING] file which holds the mask field

limitCoordsOutput

[STRING] optional parameter to limit coordinates output: 'nobounds' disables coordinate bounds, 'nocoords' avoids all coordinate information

Example

To limit data fields to land values, a mask has to be created first with:

```
cdo -gtc,0 -topo,ni96 lsm_gme96.grb
```

Here a GME grid is used. Say **temp_gme96.grb** contains a global temperature field. The following command limits the global grid to landpoints:

```
cdo -f nc reducegrid,lsm_gme96.grb temp_gme96.grb tempOnLand_gme96.nc
```

Note that output file type is NetCDF, because unstructured grids cannot be stored in GRIB format.

Author

Ralf Müller

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:

<i>Comp</i>	<i>eq</i>	Equal
	<i>ne</i>	Not equal
	<i>le</i>	Less equal
	<i>lt</i>	Less than
	<i>ge</i>	Greater equal
	<i>gt</i>	Greater than
<i>Compc</i>	<i>eqc</i>	Equal constant
	<i>nec</i>	Not equal constant
	<i>lec</i>	Less equal constant
	<i>ltc</i>	Less than constant
	<i>gec</i>	Greater equal constant
	<i>gtc</i>	Greater than constant
<i>Ymoncomp</i>	<i>ymoneq</i>	Compare time series with Equal
	<i>ymonne</i>	Compare time series with NotEqual
	<i>ymonle</i>	Compare time series with LessEqual
	<i>ymonlt</i>	Compares if time series with LessThan
	<i>ymonge</i>	Compares if time series with GreaterEqual
	<i>ymongt</i>	Compares if time series with GreaterThan
<i>Yseascomp</i>	<i>yseaseq</i>	Compare time series with Equal
	<i>yseasne</i>	Compare time series with NotEqual
	<i>yseasle</i>	Compare time series with LessEqual
	<i>yseaslt</i>	Compares if time series with LessThan
	<i>yseasge</i>	Compares if time series with GreaterEqual
	<i>yseasgt</i>	Compares if time series with GreaterThan

2.5.1 Comp

Name

eq, ne, le, lt, ge, gt - Comparison of two fields

Synopsis

cdo <operator> *infile1* *infile2* *outfile*

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 *infile1* should be the same as in *infile2*. One of the input files can contain only one timestep or one field. The fields in *outfile* inherit the meta data from *infile1* or *infile2*. The type of comparison depends on the chosen operator.

Operators

eq

Equal

$$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) = i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \neq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$$

ne

Not equal

$$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \neq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) = i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$$

le

Less equal

$$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \leq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) > i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$$

lt

Less than

$$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) < i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \geq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$$

ge

Greater equal

$$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \geq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) < i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$$

gt

Greater than

$$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) > i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \leq i_2(t, x) \quad \wedge \quad i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \quad \vee \quad i_2(t, x) = \text{miss} \end{cases}$$

Example

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

```
cdo eq infile1 infile2 outfile
```

Author

Uwe Schulzweida

2.5.2 Compc

Name

eqc, nec, lec, ltc, gec, gtc - Comparison of a field with a constant

Synopsis

cdo <operator>,c infile outfile

Description

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

Operators

eqc

Equal constant

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

nec

Not equal constant

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

lec

Less equal constant

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

ltc

Less than constant

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

gec

Greater equal constant

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

gtc

Greater than constant

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

Parameters

c

[FLOAT] Constant

Example

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

```
cdo gtc,273.15 infile outfile
```

Author

Uwe Schulzweida

2.5.3 Ymoncomp

Name

ymoneq, ymonne, ymonle, ymonlt, ymonge, ymongt - Multi-year monthly comparison

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs comparisons of a time series and one timestep with the same month of year. For each field in *infile1* the corresponding field of the timestep in *infile2* with the same month of year is used. 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. The input files need to have the same structure with the same variables. Usually *infile2* is generated by an operator of the module *Ymonstat*.

Operators

ymoneq

Compare time series with Equal

Compares whether a time series is equal to a multi-year monthly time series.

ymonne

Compare time series with NotEqual

Compares whether a time series is not equal to a multi-year monthly time series.

ymonle

Compare time series with LessEqual

Compares whether a time series is less than or equal to a multi-year monthly time series.

ymonlt

Compares if time series with LessThan

Compares whether a time series is less than a multi-year monthly time series.

ymonge

Compares if time series with GreaterEqual

Compares whether a time series is greater than or equal to a multi-year monthly time series.

ymongt

Compares if time series with GreaterThan

Compares whether a time series is greater than a multi-year monthly time series.

Author

Uwe Schulzweida

2.5.4 Yseascomp

Name

yseaseq, yseasne, yseasle, yseaslt, yseasge, yseasgt - Multi-year seasonal comparison

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs comparisons of a time series and one timestep with the same season of year. For each field in *infile1* the corresponding field of the timestep in *infile2* with the same season of year is used. 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. The input files need to have the same structure with the same variables. Usually *infile2* is generated by an operator of the module @mod{YseasSTAT}.

Operators

yseaseq

Compare time series with Equal

Compares whether a time series is equal to a multi-year seasonal time series.

yseasne

Compare time series with NotEqual

Compares whether a time series is not equal to a multi-year seasonal time series.

yseasle

Compare time series with LessEqual

Compares whether a time series is less than or equal to a multi-year seasonal time series.

yseaslt

Compares if time series with LessThan

Compares whether a time series is less than a multi-year seasonal time series.

yseasge

Compares if time series with GreaterEqual

Compares whether a time series is greater than or equal to a multi-year seasonal time series.

yseasgt

Compares if time series with GreaterThan

Compares whether a time series is greater than a multi-year seasonal time series.

Author

Uwe Schulzweida

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:

<i>Setattribute</i>	<i>setattribute</i>	Set attributes
	<i>delattribute</i>	Delete attributes
<i>Setpartab</i>	<i>setpartabp</i>	Set parameter table
	<i>setpartabn</i>	Set parameter table
<i>Set</i>	<i>setcodetab</i>	Set parameter code table
	<i>setcode</i>	Set code number
	<i>setparam</i>	Set parameter identifier
	<i>setname</i>	Set variable name
	<i>setstdname</i>	Set standard name
	<i>setunit</i>	Set variable unit
	<i>setlevel</i>	Set level
	<i>setltype</i>	Set GRIB level type
	<i>setmaxsteps</i>	Set max timesteps
	<i>setdate</i>	Set date
	<i>settime</i>	Set time of the day
<i>Settime</i>	<i>setday</i>	Set day
	<i>setmon</i>	Set month
	<i>setyear</i>	Set year
	<i>setunits</i>	Set time units
	<i>settaxis</i>	Set time axis
	<i>settbounds</i>	Set time bounds
	<i>setreftime</i>	Set reference time
	<i>setcalendar</i>	Set calendar
	<i>shifttime</i>	Shift timesteps
	<i>chcode</i>	Change code number
	<i>chparam</i>	Change parameter identifier
	<i>chname</i>	Change variable or coordinate name
	<i>chunit</i>	Change variable unit
	<i>chlevel</i>	Change level
	<i>chlevelc</i>	Change level of one code
	<i>chlevelv</i>	Change level of one variable
<i>Setgrid</i>	<i>setgrid</i>	Set grid
	<i>setgridtype</i>	Set grid type
	<i>setgridarea</i>	Set grid cell area
	<i>setgridmask</i>	Set grid mask
	<i>setprojparams</i>	Set proj params
<i>Setzaxis</i>	<i>setzaxis</i>	Set z-axis
	<i>genlevelbounds</i>	Generate level bounds
<i>Invert</i>	<i>invertlat</i>	Invert latitudes
<i>Invertlev</i>	<i>invertlev</i>	Invert levels
<i>Shiftxy</i>	<i>shiftx</i>	Shift x
	<i>shifty</i>	Shift y
<i>Maskregion</i>	<i>maskregion</i>	Mask regions
<i>Maskbox</i>	<i>masklonlatbox</i>	Mask a longitude/latitude box
	<i>maskindexbox</i>	Mask an index box
<i>Setbox</i>	<i>setclonlatbox</i>	Set a longitude/latitude box to constant
	<i>setcindexbox</i>	Set an index box to constant
<i>Enlarge</i>	<i>enlarge</i>	Enlarge fields
<i>Setmiss</i>	<i>setmissval</i>	Set a new missing value
	<i>setctomiss</i>	Set constant to missing value

continues on next page

Table 11 – continued from previous page

	<i>setmisstoc</i>	Set missing value to constant
	<i>setrtomiss</i>	Set range to missing value
	<i>setvrang</i>	Set valid range
	<i>setmisstonn</i>	Set missing value to nearest neighbor
	<i>setmisstodis</i>	Set missing value to distance-weighted average
<i>Vertfillmiss</i>	<i>vertfillmiss</i>	Vertical filling of missing values
<i>Timfillmiss</i>	<i>timfillmiss</i>	Temporal filling of missing values
<i>Setgridcell</i>	<i>setgridcell</i>	Set the value of a grid cell

2.6.1 Setattribute

Name

setattribute, delattribute - Set attributes

Synopsis

cdo <operator>.parameters *infile outfile*

Description

This operator sets or deletes attributes of a dataset and writes the result to **outfile**. The new attributes are only available in **outfile** if the file format supports attributes.

Each attribute has the following structure:

[**var_nm**@]**att_nm**[:s|d|i]=[**att_val** | {[**var_nm**@]**att_nm**}]

var_nm	Variable name (optional). Example: pressure
att_nm	Attribute name. Example: units
att_val	Comma-separated list of attribute values. Example: pascal

The value of **var_nm** is the name of the variable containing the attribute (named **att_nm**) that you want to set. Use wildcards to set the attribute **att_nm** to more than one variable. A value of **var_nm** of '*' will set the attribute **att_nm** to all data variables. If **var_nm** is missing then **att_nm** refers to a global attribute.

The value of **att_nm** is the name of the attribute you want to set. For each attribute a string (att_nm:s), a double (att_nm:d) or an integer (att_nm:i) type can be defined. By default the native type is set.

The value of **att_val** is the contents of the attribute **att_nm**. **att_val** may be a single value or one-dimensional array of elements. The type and the number of elements of an attribute will be detected automatically from the contents of the values. An already existing attribute **att_nm** will be overwritten or it will be removed if **att_val** is omitted. Alternatively, the values of an existing attribute can be copied. This attribute must then be enclosed in curly brackets.

A special meaning has the attribute name **FILE**. If this is the 1st attribute then all attributes are read from a file specified in the value of **att_val**.

Some NetCDF attributes can't be deleted. Here is an incomplete list: `missing_value`, `formula_terms`, `cell_measures`, `coordinates`, `grid_mapping`, `valid_range`, ...

Operators

setattribute

Set attributes

delattribute

Delete attributes

Parameters

attributes

[STRING] Comma-separated list of attributes.

Note

Attributes are evaluated by **CDO** when opening **infile**. Therefore the result of this operator is not available for other operators when this operator is used in chaining operators.

Example

To set the units of the variable pressure to pascal use:

```
cdo setattribute,pressure@units=pascal infile outfile
```

To set the global text attribute "my_att" to "my contents", use:

```
cdo setattribute,my_att="my contents" infile outfile
```

Result of `ncdump -h outfile`:

```
netcdf outfile {  
dimensions: ...  
  
variables: ...  
  
// global attributes:  
           :my_att = "my contents" ;  
}
```

Author

Uwe Schulzweida

2.6.2 Setpartab

Name

setpartabp, setpartabn - Set parameter table

Synopsis

cdo <operator>.parameters *infile outfile*

Description

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

The following parameter table entries are supported:

Entry	Type	Description
name	WORD	Name of the variable
out_name	WORD	New name of the variable
param	WORD	Parameter identifier (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]])
out_param	WORD	New parameter identifier
type	WORD	Data type (real or double)
standard_name	WORD	As defined in the CF standard name table
long_name	STRING	Describing the variable
units	STRING	Specifying the units for the variable
comment	STRING	Information concerning the variable
cell_methods	STRING	Information concerning calculation of means or climatologies
cell_measures	STRING	Indicates the names of the variables containing cell areas and volumes
filterspec	STRING	NetCDF4 filter specification
missing_value	FLOAT	Specifying how missing data will be identified
valid_min	FLOAT	Minimum valid value
valid_max	FLOAT	Maximum valid value
ok_min_mean_abs	FLOAT	Minimum absolute mean
ok_max_mean_abs	FLOAT	Maximum absolute mean
factor	FLOAT	Scale factor
delete	INTEGER	Set to 1 to delete variable
convert	INTEGER	Set to 1 to convert the unit if necessary

Unsupported parameter table entries are stored as variable attributes. The search key for the variable depends on the operator. Use @oper{setpartabn} to search variables by the name. This is typically used for NetCDF datasets. The operator @oper{setpartabp} searches variables by the parameter ID.

Operators

setpartabp

Set parameter table

Search variables by the parameter identifier.

setpartabn

Set parameter table

Search variables by name.

Parameters

table

[STRING] Parameter table file or name

convert

[STRING] Converts the units if necessary

Example

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

```
prompt> cat mypartab
&parameter
name           = t
out_name       = ta
standard_name  = air_temperature
units          = "K"
missing_value  = 1.0e+20
valid_min      = 157.1
valid_max      = 336.3
/
```

To apply this parameter table to a dataset use:

```
cdo setpartabn,mypartab,convert infile outfile
```

This command renames the variable **t** to **ta**. The standard name of this variable is set to **air_temperature** and the unit is set to **[K]** (converts the unit if necessary). The missing value will be set to **1.0e+20**. In addition it will be checked whether the values of the variable are in the range of **157.1** to **336.3**.

Author

Uwe Schulzweida

2.6.3 Set

Name

setcodetab, setcode, setparam, setname, setstdname, setunit, setlevel, setltype, setmaxsteps - Set field info

Synopsis

cdo <operator>.parameters *infile outfile*

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

setcodetab

Set parameter code table

Sets the parameter code table for all variables (Parameter: table).

setcode

Set code number

Sets the code number for all variables to the same given value (Parameter: code).

setparam

Set parameter identifier

Sets the parameter identifier of the first variable (Parameter: param).

setname

Set variable name

Sets the name of the first variable (Parameter: name).

setstdname

Set standard name

Sets the standard name of the first variable (Parameter: stdname).

setunit

Set variable unit

Sets the unit of the first variable (Parameter: unit).

setlevel

Set level

Sets the first level of all variables (Parameter: level).

setltype

Set GRIB level type

Sets the GRIB level type of all variables (Parameter: ltype).

setmaxsteps

Set max timesteps

Sets maximum number of timesteps (Parameter: maxsteps).

Parameters

table

[STRING] Parameter table file or name

code

[INTEGER] Code number

param

[STRING] Parameter identifier (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]])

name

[STRING] Variable name

stdname

[STRING] Standard name

level

[FLOAT] New level

ltype

[INTEGER] GRIB level type

maxsteps

[INTEGER] Maximum number of timesteps

Author

Uwe Schulzweida

2.6.4 Settime

Name

setdate, settime, setday, setmon, setyear, settunits, settaxis, settbounds, setreftime, setcalendar, shifttime - Set time

Synopsis

cdo <operator>,*parameters infile outfile*

Description

This module sets the time axis or part of the time axis. Which part of the time axis is overwritten/created depends on the chosen operator. The number of time steps does not change.

Operators

setdate

Set date

Sets the date in every timestep to the same given value (Parameter: date).

settime

Set time of the day

Sets the time in every timestep to the same given value (Parameter: time).

setday

Set day

Sets the day in every timestep to the same given value (Parameter: day).

setmon

Set month

Sets the month in every timestep to the same given value (Parameter: month).

setyear

Set year

Sets the year in every timestep to the same given value (Parameter: year).

settunits

Set time units

Sets the base units of a relative time axis (Parameter: units).

settaxis

Set time axis

Sets the time axis (Parameter: date time [inc]).

settbounds

Set time bounds

Sets the time bounds (Parameter: frequency).

setreftime

Set reference time

Sets the reference time of a relative time axis (Parameter: date time [units]).

setcalendar

Set calendar

Sets the calendar attribute of a relative time axis (Parameter: calendar).

shifttime

Shift timesteps

Shifts all timesteps by the parameter shiftValue (Parameter: shiftValue).

Parameters**day**

[INTEGER] Value of the new day

month

[INTEGER] Value of the new month

year

[INTEGER] Value of the new year

units

[STRING] Base units of the time axis (seconds|minutes|hours|days|months|years)

date

[STRING] Date (format: YYYY-MM-DD)

time

[STRING] Time (format: hh:mm:ss)

inc

[STRING] Optional increment (seconds|minutes|hours|days|months|years) [default: 1hour]

frequency

[STRING] Frequency of the time series (hour|day|month|year)

calendar

[STRING] Calendar (standard|proleptic_gregorian|360_day|365_day|366_day)

shiftValue

[STRING] Shift value (e.g. -3hour)

Example

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

```
cdo settaxis,1987-01-16,12:00:00,1mon infile outfile
```

Result of `cdo showdate outfile` for a dataset with 12 timesteps:

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

To shift this time axis by -15 days use:

```
cdo shifttime,-15days infile outfile
```

Result of `cdo showdate outfile`:

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

Author

Uwe Schulzweida

2.6.5 Change

Name

chcode, chparam, chname, chunit, chlevel, chlevelc, chlevelv - Change field header

Synopsis

cdo <operator>,*parameters infile outfile*

Description

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

Operators

chcode

Change code number

Changes some user given code numbers to new user given values (Parameter: oldcode newcode [...]).

chparam

Change parameter identifier

Changes some user given parameter identifiers to new user given values (Parameter: oldparam newparam ...).

chname

Change variable or coordinate name

Changes some user given variable or coordinate names to new user given names (Parameter: oldname newname ...).

chunit

Change variable unit

Changes some user given variable units to new user given units (Parameter: oldunit newunit ...).

chlevel

Change level

Changes some user given levels to new user given values (Parameter: oldlev newlev ...).

chlevelc

Change level of one code

Changes one level of a user given code number (Parameter: code oldlev newlev).

chlevelv

Change level of one variable

Changes one level of a user given variable name (Parameter: name oldlev newlev).

Parameters

code

[INTEGER] Code number

oldcode,newcode,...

[INTEGER] Pairs of old and new code numbers

oldparam,newparam,...

[STRING] Pairs of old and new parameter identifiers

name

[STRING] Variable name

oldname,newname,...

[STRING] Pairs of old and new variable names

oldlev

[FLOAT] Old level

newlev

[FLOAT] New level

oldlev,newlev,...

[FLOAT] Pairs of old and new levels

Example

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

```
cdo chcode,98,179,99,211 infile outfile
```

Author

Uwe Schulzweida

2.6.6 Setgrid

Name

setgrid, setgridtype, setgridarea, setgridmask, setprojparams - Set grid information

Synopsis

cdo <operator>.parameters *infile outfile*

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 (Parameter: grid). The input fields need to have the same grid size as the size of the target grid description. The target grid can be an ASCII file containing the grid description, a predefined grid name or the name of a dataset. If the dataset contains more than one grid, the first one is used. Appending :N to the name of the dataset selects the N-th grid.

setgridtype

Set grid type

Sets the grid type of all input fields (Parameter: gridtype). The following grid types are available:

curvilinear	Converts a regular grid to a curvilinear grid
unstructured	Converts a regular or curvilinear grid to an unstructured grid
dereference	Dereference a reference to a grid
regular	Linear interpolation of a reduced Gaussian grid to a regular Gaussian grid
regularnn	Nearest neighbor interpolation of a reduced Gaussian grid to a regular Gaussian grid
lonlat	Converts a regular lonlat grid stored as a curvilinear grid back to a lonlat grid
projection	Removes the geographical coordinates if projection parameter available

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*.

setgridmask

Set grid mask

Sets the grid mask. The parameter *gridmask* is the path to a data file, the first field is used as the grid mask. The input fields need to have the same grid size as the grid mask. The grid mask is used as the target grid mask for remapping, e.g. for *remapbil*.

setprojparams

Set proj params

Sets the *proj_params* attribute of a projection. This attribute is used to compute geographic coordinates of a projection with the proj library (Parameter: projparams).

Parameters

grid

[STRING] Grid description file, predefined grid name or name of a data stream

gridtype

[STRING] Grid type (curvilinear, unstructured, regular, lonlat, projection or dereference)

gridarea

[STRING] Data file, the first field is used as grid cell area

gridmask

[STRING] Data file, the first field is used as grid mask

projparams

[STRING] Proj library parameter (e.g.:+init=EPSG:3413)

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 regular Gaussian F32 grid (8192 gridpoints) use:

```
cdo setgrid,F32 infile outfile
```

Author

Uwe Schulzweida

2.6.7 Setzaxis

Name

setzaxis, genlevelbounds - Set z-axis information

Synopsis

cdo <operator>.parameters *infile outfile*

Description

This module modifies the metadata of the vertical grid.

Operators

setzaxis

Set z-axis

This operator sets the z-axis description of all variables with the same number of level as the new z-axis (Parameter: zaxis).

genlevelbounds

Generate level bounds

Generates the layer bounds of the z-axis (Parameter: zbot ztop).

Parameters

zaxis

[STRING] Z-axis description file or name of the target z-axis

zbot

[FLOAT] Specifying the bottom of the vertical column. Must have the same units as z-axis.

ztop

[FLOAT] Specifying the top of the vertical column. Must have the same units as z-axis.

Author

Uwe Schulzweida

2.6.8 Invert

Name

invertlat - Invert latitudes

Synopsis

cdo invertlat *infile outfile*

Description

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

Example

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

```
cdo invertlat infile outfile
```

Author

Uwe Schulzweida

2.6.9 Invertlev

Name

invertlev - Invert levels

Synopsis

cdo invertlev *infile outfile*

Description

This operator inverts the levels of all 3D variables.

Author

Uwe Schulzweida

2.6.10 Shiftxy

Name

shiftx, shifty - Shift field

Synopsis

cdo <operator>.parameters *infile outfile*

Description

This module contains operators to shift all fields in x or y direction. All fields need to have the same horizontal rectilinear or curvilinear grid.

Operators

shiftx

Shift x

Shifts all fields in x direction.

shifty

Shift y

Shifts all fields in y direction.

Parameters

nshift

[INTEGER] Number of grid cells to shift (default: 1)

cyclic

[STRING] If set, cells are filled up cyclic (default: missing value)

coord

[STRING] If set, coordinates are also shifted

Example

To shift all input fields in the x direction by +1 cells and fill the new cells with missing value, use:

```
cdo shiftx infile outfile
```

To shift all input fields in the x direction by +1 cells and fill the new cells cyclic, use:

```
cdo shiftx,1,cyclic infile outfile
```

Author

Uwe Schulzweida

2.6.11 Maskregion

Name

maskregion - Mask regions

Synopsis

cdo maskregion,*regions infile outfile*

Description

Masks different regions of the input fields. The grid cells inside a region are untouched, the cells 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.

Regions can be defined by the user via an ASCII file. Each region consists of the geographic coordinates of 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 &.

Predefined regions of countries can be specified via the country codes. A country is specified with dcw:<CountryCode>. Country codes can be combined with the plus sign.

Parameters

regions

[STRING] Comma-separated list of ASCII formatted files with different regions

Example

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

```
cdo maskregion,myregion infile outfile
```

For this example the description file of the region myregion should contain one polygon with the following four coordinates:

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

To mask the region of a country use the country code with data from the Digital Chart of the World. Here is an example for Spain with the country code ES:

```
cdo maskregion,dcw:ES infile outfile
```

Author

Uwe Schulzweida, Cedrick Ansorge

2.6.12 Maskbox

Name

masklonlatbox, maskindexbox - Mask a box

Synopsis

cdo masklonlatbox,*lon1,lon2,lat1,lat2 infile outfile*

cdo maskindexbox,*idx1,idx2,idy1,idy2 infile outfile*

Description

Masks grid cells inside a lon/lat or index box. 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 *sel lonlatbox* or *sel indexbox* if only the data inside the box are needed.

Operators

masklonlatbox

Mask a longitude/latitude box

Masks grid cells inside a lon/lat box. The user must specify the longitude and latitude of the edges of the box. Only those grid cells are considered whose grid center lies within the lon/lat box. For rotated lon/lat grids the parameters must be specified in rotated coordinates.

maskindexbox

Mask an index box

Masks grid cells within an index box. The user must specify the indices of the edges of the box. The index of the left edge can be greater than the one of the right edge. Use negative indexing to start from the end. The input grid must be a regular lon/lat or a 2D curvilinear grid.

Parameters

lon1

[FLOAT] Western longitude in degrees

lon2

[FLOAT] Eastern longitude in degrees

lat1

[FLOAT] Southern or northern latitude in degrees

lat2

[FLOAT] Northern or southern latitude in degrees

idx1

[INTEGER] Index of first longitude (1 - nlon)

idx2

[INTEGER] Index of last longitude (1 - nlon)

idy1

[INTEGER] Index of first latitude (1 - nlat)

idy2

[INTEGER] Index of last latitude (1 - nlat)

Example

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

```
cdo masklonlatbox,120,-90,20,-20 infile outfile
```

If the input dataset has fields on a regular Gaussian F16 grid, the same box can be masked with *maskindexbox* by:

```
cdo maskindexbox,23,48,13,20 infile outfile
```

Author

Uwe Schulzweida

2.6.13 Setbox

Name

setclonlatbox, setcindexbox - Set a box to constant

Synopsis

cdo setclonlatbox,*c,lon1,lon2,lat1,lat2 infile outfile*

cdo setcindexbox,*c,idx1,idx2,idy1,idy2 infile outfile*

Description

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

Operators

setclonlatbox

Set a longitude/latitude box to constant

Sets the values of a longitude/latitude box to a constant value. The user has to give the longitudes and latitudes of the edges of the box.

setcindexbox

Set an index box to constant

Sets the values of an index box to a constant value. The user has to give the indices of the edges of the box. The index of the left edge can be greater than the one of the right edge.

Parameters

c

[FLOAT] Constant

lon1

[FLOAT] Western longitude in degrees

lon2

[FLOAT] Eastern longitude in degrees

lat1

[FLOAT] Southern or northern latitude in degrees

lat2

[FLOAT] Northern or southern latitude in degrees

idx1

[INTEGER] Index of first longitude (1 - nlon)

idx2

[INTEGER] Index of last longitude (1 - nlon)

idy1

[INTEGER] Index of first latitude (1 - nlat)

idy2

[INTEGER] Index of last latitude (1 - nlat)

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 infile outfile
```

If the input dataset has fields on a regular Gaussian F16 grid, the same box can be set with *setcindexbox* by:

```
cdo setcindexbox,-1.23,23,48,13,20 infile outfile
```

Author

Etienne Tourigny

2.6.14 Enlarge

Name

enlarge - Enlarge fields

Synopsis

cdo enlarge,*grid infile outfile*

Description

Enlarge all fields of *infile* to a user given horizontal 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.

Parameters

grid

[STRING] Target grid description file or name

Example

Assume 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 has to be enlarged to the grid size of the first dataset:

```
cdo enlarge,infile1 infile2 tmpfile
cdo add infile1 tmpfile outfile
```

Or shorter using operator piping:

```
cdo add infile1 -enlarge,infile1 infile2 outfile
```

Author

Uwe Schulzweida

2.6.15 Setmiss

Name

setmissval, setctomiss, setmisstoc, setrtomiss, setvrange, setmisstonn, setmisstodis - Set missing value

Synopsis

cdo <operator>.parameters infile outfile

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

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

setvrange

Set valid range

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

setmisstonn

Set missing value to nearest neighbor

Set all missing values to the nearest non missing value.

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

setmisstodis

Set missing value to distance-weighted average

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

Parameters

neighbors

[INTEGER] Number of nearest neighbors

newmiss

[FLOAT] New missing value

c

[FLOAT] Constant

rmin

[FLOAT] Lower bound

rmax

[FLOAT] Upper bound

Example

setrtomiss

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

```
cdo setrtomiss,0,273.15 infile outfile
```

Result of cdo info infile:

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

Result of cdo info outfile:

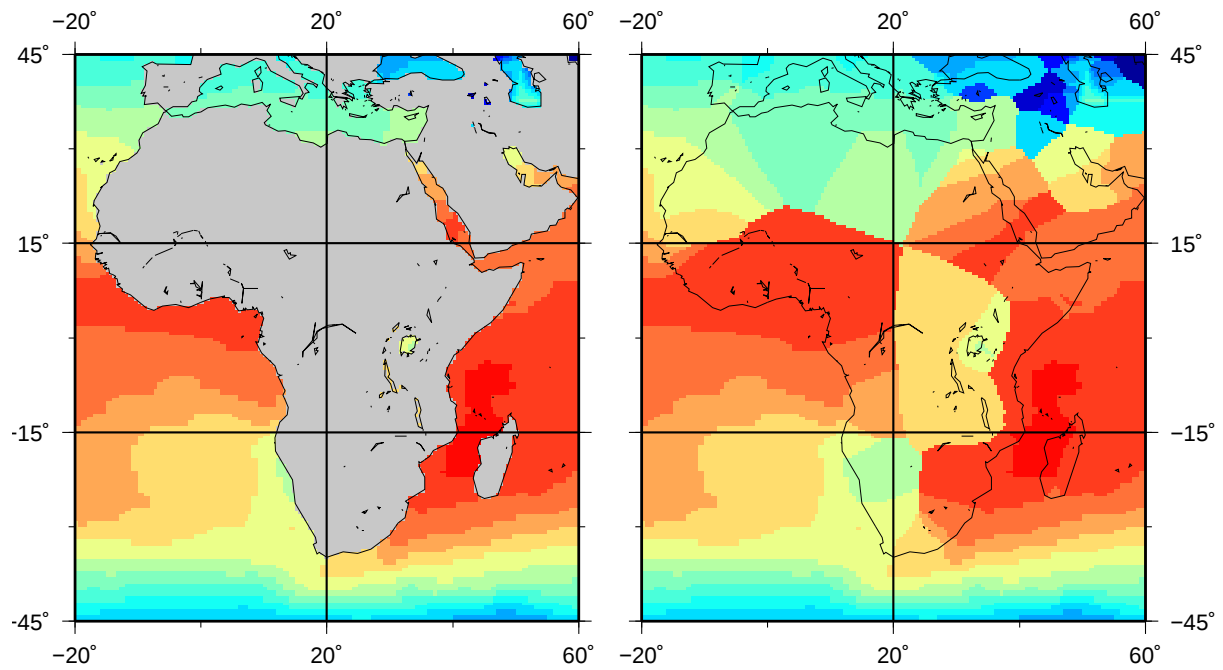
-1 :	Date	Time	Code	Level	Size	Miss :	Minimum	Mean	Maximum
1 :	1987-12-31	12:00:00	139	0	2048	871 :	273.16	287.08	303.71

setmisstonn

Set all missing values to the nearest non missing value:

```
cdo setmisstonn infile outfile
```

Below is a schematic illustration of this example:



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

Author

Uwe Schulzweida

2.6.16 Vertfillmiss

Name

vertfillmiss - Vertical filling of missing values

Synopsis

cdo vertfillmiss,*parameters infile outfile*

Description

This operator fills in vertical missing values. The *method* parameter can be used to select the filling method. The default *method=nearest* fills missing values with the nearest neighbor value. Other options are *forward* and *backward* to fill missing values by forward or backward propagation of values. Use the *limit* parameter to set the maximum number of consecutive missing values to fill and *max_gaps* to set the maximum number of gaps to fill.

Parameters

method

[STRING] Fill method [nearest|linear|forward|backward] (default: nearest)

limit

[INTEGER] The maximum number of consecutive missing values to fill (default: all)

max_gaps

[INTEGER] The maximum number of gaps to fill (default: all)

Author

Uwe Schulzweida

2.6.17 Timfillmiss

Name

timfillmiss - Temporal filling of missing values

Synopsis

cdo timfillmiss,*parameters infile outfile*

Description

This operator fills in temporally missing values. The *method* parameter can be used to select the filling method. The default *method=nearest* fills missing values with the nearest neighbor value. Other options are *forward* and *backward* to fill missing values by forward or backward propagation of values. Use the *limit* parameter to set the maximum number of consecutive missing values to fill and *max_gaps* to set the maximum number of gaps to fill.

Parameters

method

[STRING] Fill method [nearest|linear|forward|backward] (default: nearest)

limit

[INTEGER] The maximum number of consecutive missing values to fill (default: all)

max_gaps

[INTEGER] The maximum number of gaps to fill (default: all)

Author

Uwe Schulzweida

2.6.18 Setgridcell

Name

setgridcell - Set the value of a grid cell

Synopsis

cdo setgridcell,*parameters infile outfile*

Description

This operator sets the value of the selected grid cells. The grid cells can be selected by a comma-separated list of grid cell indices or a mask. The mask is read from a data file, which may contain only one field. If no grid cells are selected, all values are set.

Parameters

value

[FLOAT] Value of the grid cell

cell

[INTEGER] Comma-separated list of grid cell indices

mask

[STRING] Name of the data file which contains the mask

Author

Uwe Schulzweida

2.7 Arithmetic

This section contains modules to arithmetically process datasets.

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

<i>Expr</i>	<i>expr</i>	Evaluate expressions
	<i>exprf</i>	Evaluate expressions script
	<i>aexpr</i>	Evaluate expressions and append results
	<i>aexprf</i>	Evaluate expression script and append results
<i>Math</i>	<i>abs</i>	Absolute value
	<i>int</i>	Integer value
	<i>nint</i>	Nearest integer value
	<i>pow</i>	Power
	<i>sqr</i>	Square
	<i>sqrt</i>	Square root
	<i>exp</i>	Exponential
	<i>ln</i>	Natural logarithm
	<i>log10</i>	Base 10 logarithm
	<i>sin</i>	Sine
	<i>cos</i>	Cosine
	<i>tan</i>	Tangent
	<i>asin</i>	Arc sine
	<i>acos</i>	Arc cosine
	<i>atan</i>	Arc tangent
	<i>reci</i>	Reciprocal value
	<i>not</i>	Logical NOT
<i>Arithc</i>	<i>addc</i>	Add a constant
	<i>subc</i>	Subtract a constant
	<i>mulc</i>	Multiply with a constant
	<i>divc</i>	Divide by a constant
	<i>minc</i>	Minimum of a field and a constant
	<i>maxc</i>	Maximum of a field and a constant
<i>Arith</i>	<i>add</i>	Add two fields
	<i>sub</i>	Subtract two fields
	<i>mul</i>	Multiply two fields
	<i>div</i>	Divide two fields
	<i>min</i>	Minimum of two fields
	<i>max</i>	Maximum of two fields
	<i>atan2</i>	Arc tangent of two fields
	<i>setmiss</i>	Set missing values
<i>Dayarith</i>	<i>dayadd</i>	Add daily time series
	<i>daysub</i>	Subtract daily time series
	<i>daymul</i>	Multiply daily time series
	<i>daydiv</i>	Divide daily time series
<i>Monarith</i>	<i>monadd</i>	Add monthly time series
	<i>monsub</i>	Subtract monthly time series
	<i>monmul</i>	Multiply monthly time series
	<i>mondiv</i>	Divide monthly time series
<i>Yeararith</i>	<i>yearadd</i>	Add yearly time series
	<i>yearsub</i>	Subtract yearly time series
	<i>yearmul</i>	Multiply yearly time series
	<i>yeardiv</i>	Divide yearly time series
<i>Yhourarith</i>	<i>yhouradd</i>	Add multi-year hourly time series
	<i>yhoursub</i>	Subtract multi-year hourly time series

continues on next page

Table 12 – continued from previous page

	<i>yhourmul</i>	Multiply multi-year hourly time series
	<i>yhourdiv</i>	Divide multi-year hourly time series
<i>Ydayarith</i>	<i>ydayadd</i>	Add multi-year daily time series
	<i>ydaysub</i>	Subtract multi-year daily time series
	<i>ydaymul</i>	Multiply multi-year daily time series
	<i>ydaydiv</i>	Divide multi-year daily time series
<i>Ymonarith</i>	<i>ymonadd</i>	Add multi-year monthly time series
	<i>ymonsub</i>	Subtract multi-year monthly time series
	<i>ymonmul</i>	Multiply multi-year monthly time series
	<i>ymondiv</i>	Divide multi-year monthly time series
<i>Ysearith</i>	<i>yseasadd</i>	Add multi-year seasonal time series
	<i>yseassub</i>	Subtract multi-year seasonal time series
	<i>yseasmul</i>	Multiply multi-year seasonal time series
	<i>yseasdiv</i>	Divide multi-year seasonal time series
<i>Arithdays</i>	<i>muldpm</i>	Multiply with days per month
	<i>divdpm</i>	Divide by days per month
	<i>muldpy</i>	Multiply with days per year
	<i>divdpy</i>	Divide by days per year
<i>Arithlat</i>	<i>mulcoslat</i>	Multiply with the cosine of the latitude
	<i>divcoslat</i>	Divide by cosine of the latitude

2.7.1 Expr

Name

expr, exprf, aexpr, aexprf - Evaluate expressions

Synopsis

cdo <operator>.parameters infile outfile

Description

This module arithmetically processes every timestep of the input dataset. Each individual assignment statement have to end with a semi-colon. The special key `_ALL_` is used as a template. A statement with a template is replaced for all variable names. Unlike regular variables, temporary variables are never written to the output stream. To define a temporary variable simply prefix the variable name with an underscore (e.g. `_varname`) when the variable is declared.

The following operators are supported:

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

The following functions are supported:

Math intrinsics:

abs(x)	Absolute value of x
floor(x)	Round to largest integral value not greater than x
ceil(x)	Round to smallest integral value not less than x
float(x)	32-bit float 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
ln(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
sinh(x)	Hyperbolic sine of x, where x is specified in radians
cosh(x)	Hyperbolic cosine of x, where x is specified in radians
tanh(x)	Hyperbolic tangent of x, where x is specified in radians
asinh(x)	Inverse hyperbolic sine of x, where x is specified in radians
acosh(x)	Inverse hyperbolic cosine of x, where x is specified in radians
atanh(x)	Inverse hyperbolic tangent of x, where x is specified in radians
rad(x)	Convert x from degrees to radians
deg(x)	Convert x from radians to degrees
rand(x)	Replace x by pseudo-random numbers in the range of 0 to 1
isMissval(x)	Returns 1 where x is missing

mod(x,y)	Floating-point remainder of x/ y
min(x,y)	Minimum value of x and y
max(x,y)	Maximum value of x and y
pow(x,y)	Power function
hypot(x,y)	Euclidean distance function, $\sqrt{x^2 + y^2}$
atan2(x,y)	Arc tangent function of y/x, using signs to determine quadrants
trimrel(x,kb)	Trim relative precision to kb keep-bits. Max relative error to a value $ \text{result} - x / x < 2^{*(-1-kb)}$ for any finite x. trimrel is a non-decreasing function of x. Loosely follows A5 of https://gmd.copernicus.org/articles/14/377/2021/
trimabs(x,err)	Trim absolute precision introducing given max. absolute error $ \text{result} - x < \text{err}$, trimabs is a non-decreasing function of x. Loosely follows A6 of https://gmd.copernicus.org/articles/14/377/2021/

Coordinates:

<code>clon(x)</code>	Longitude coordinate of x (available only if x has geographical coordinates)
<code>clat(x)</code>	Latitude coordinate of x (available only if x has geographical coordinates)
<code>gridarea(x)</code>	Grid cell area of x (available only if x has geographical coordinates)
<code>gridindex(x)</code>	Grid cell indices of x
<code>clev(x)</code>	Level coordinate of x (0, if x is a 2D surface variable)
<code>clevidx(x)</code>	Level index of x (0, if x is a 2D surface variable)
<code>cthickness(x)</code>	Layer thickness, upper minus lower level bound of x (1, if level bounds are missing)
<code>ctimestep()</code>	Timestep number (1 to N)
<code>cdate()</code>	Verification date as YYYYMMDD
<code>ctime()</code>	Verification time as HHMMSS.millisecond
<code>cdeltat()</code>	Difference between current and last timestep in seconds
<code>cday()</code>	Day as DD
<code>cmonth()</code>	Month as MM
<code>cyear()</code>	Year as YYYY
<code>csecond()</code>	Second as SS.millisecond
<code>cminute()</code>	Minute as MM
<code>chour()</code>	Hour as HH

Constants:

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

Statistics over a field:

`fldmin(x)`, `fldmax(x)`, `fldrange(x)`, `fldsum(x)`, `fldmean(x)`, `fldavg(x)`, `fldstd(x)`, `fldstdl(x)`, `fldvar(x)`, `fldvarl(x)`, `fldskew(x)`, `fldkurt(x)`, `fldmedian(x)`

Zonal statistics for regular 2D grids:

`zonmin(x)`, `zonmax(x)`, `zonrange(x)`, `zonsum(x)`, `zonmean(x)`, `zonavg(x)`, `zonstd(x)`, `zonstdl(x)`, `zonvar(x)`, `zonvarl(x)`, `zonskew(x)`, `zonkurt(x)`, `zonmedian(x)`

Vertical statistics:

`vertmin(x)`, `vertmax(x)`, `vertrange(x)`, `vertsum(x)`, `vertmean(x)`, `vertavg(x)`, `vertstd(x)`, `vertstdl(x)`, `vertvar(x)`, `vertvarl(x)`

Miscellaneous:

<code>sellevel(x,k)</code>	Select level k of variable x
<code>sellevidx(x,k)</code>	Select level index k of variable x
<code>sellevelrange(x,k1,k2)</code>	Select all levels of variable x in the range k1 to k2
<code>sellevidxrange(x,k1,k2)</code>	Select all level indices of variable x in the range k1 to k2
<code>remove(x)</code>	Remove variable x from output stream

Operators

expr

Evaluate expressions

The processing instructions are read from the *instr* parameter .

exprf

Evaluate expressions script

Contrary to *expr* the processing instructions are read from a file (Parameter: filename).

aexpr

Evaluate expressions and append results

Same as *expr*, but keep input variables and append results

aexprf

Evaluate expression script and append results

Same as *exprf*, but keep input variables and append results

Parameters**instr**

[STRING] Processing instructions (need to be 'quoted' in most cases)

filename

[STRING] File with processing instructions

Note

If the input stream contains duplicate entries of the same variable name then the last one is used.

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;' infile outfile
```

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

```
cdo exprf,myexpr infile outfile
```

The file myexpr contains:

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

Author

Uwe Schulzweida, Karl-Hermann Wieners

2.7.2 Math

Name

abs, int, nint, pow, sqr, sqrt, exp, ln, log10, sin, cos, tan, asin, acos, atan, reci, not - Mathematical functions

Synopsis

cdo <operator> infile outfile

Description

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

Operators

abs

Absolute value

$$o(t, x) = \text{abs}(i(t, x))$$

int

Integer value

$$o(t, x) = \text{int}(i(t, x))$$

nint

Nearest integer value

$$o(t, x) = \text{nint}(i(t, x))$$

pow

Power

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

sqr

Square

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

sqrt

Square root

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

exp

Exponential

$$o(t, x) = e^{i(t, x)}$$

ln

Natural logarithm

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

log10

Base 10 logarithm

$$o(t, x) = \log_{10}(i(t, x))$$

sin

Sine

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

cos

Cosine

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

tan

Tangent

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

asin

Arc sine

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

acos

Arc cosine

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

atan

Arc tangent

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

reci

Reciprocal value

$$o(t, x) = 1/i(t, x)$$

not

Logical NOT

$$o(t, x) = 1, \text{ if } x \text{ equal } 0; \text{ else } 0$$

Example

To calculate the square root for all field elements use:

```
cdo sqrt infile outfile
```

Author

Uwe Schulzweida

2.7.3 Arithc

Name

addc, subc, mulc, divc, minc, maxc - Arithmetic with a constant

Synopsis

cdo <operator> *infile outfile*

Description

This module performs simple arithmetic with all field elements of a dataset and a constant. The fields in *outfile* inherit the meta data from *infile*.

Operators

addc

Add a constant

$$o(t, x) = i(t, x) + c$$

subc

Subtract a constant

$$o(t, x) = i(t, x) - c$$

mulc

Multiply with a constant

$$o(t, x) = i(t, x) * c$$

divc

Divide by a constant

$$o(t, x) = i(t, x) / c$$

minc

Minimum of a field and a constant

$$o(t, x) = \min(i(t, x), c)$$

maxc

Maximum of a field and a constant

$$o(t, x) = \max(i(t, x), c)$$

Parameters

c

[FLOAT] Constant

Example

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

```
cdo addc,-273.15 infile outfile
```

Author

Uwe Schulzweida

2.7.4 Arith

Name

add, sub, mul, div, min, max, atan2, setmiss - Arithmetic on two datasets

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs simple arithmetic of two datasets. The number of fields in **infile1** should be the same as in **infile2**. The fields in **outfile** inherit the meta data from **infile1**. All operators in this module simply process one field after the other from the two input files. Neither the order of the variables nor the date is checked. One of the input files can contain only one timestep or one variable.

Operators

add

Add two fields

$$o(t, x) = i_1(t, x) + i_2(t, x)$$

sub

Subtract two fields

$$o(t, x) = i_1(t, x) - i_2(t, x)$$

mul

Multiply two fields

$$o(t, x) = i_1(t, x) * i_2(t, x)$$

div

Divide two fields

$$o(t, x) = i_1(t, x) / i_2(t, x)$$

min

Minimum of two fields

$$o(t, x) = \min(i_1(t, x), i_2(t, x))$$

max

Maximum of two fields

$$o(t, x) = \max(i_1(t, x), i_2(t, x))$$

atan2

Arc tangent of two fields

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

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

setmiss

Set missing values

Sets missing values of **infile1** to values from **infile2**.

Example

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

```
cdo add infile1 infile2 outfile
```

Author

Uwe Schulzweida

2.7.5 Dayarith

Name

dayadd, daysub, daymul, daydiv - Daily arithmetic

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs simple arithmetic of a time series and one timestep with the same day, month and year. For each field in *infile1* the corresponding field of the timestep in *infile2* with the same day, month and year is used. The input files need to have the same structure with the same variables. Usually *infile2* is generated by an operator of the module *Daystat*.

Operators

dayadd

Add daily time series

Adds a time series and a daily time series.

daysub

Subtract daily time series

Subtracts a time series and a daily time series.

daymul

Multiply daily time series

Multiplies a time series and a daily time series.

daydiv

Divide daily time series

Divides a time series and a daily time series.

Example

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

```
cdo daysub infile -dayavg infile outfile
```

Author

Uwe Schulzweida

2.7.6 Monarith

Name

monadd, monsub, monmul, mondiv - Monthly arithmetic

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs simple arithmetic of a time series and one timestep with the same month and year. For each field in *infile1* the corresponding field of the timestep in *infile2* with the same month and year is used. The input files need to have the same structure with the same variables. Usually *infile2* is generated by an operator of the module *Monstat*.

Operators

monadd

Add monthly time series

Adds a time series and a monthly time series.

monsub

Subtract monthly time series

Subtracts a time series and a monthly time series.

monmul

Multiply monthly time series

Multiplies a time series and a monthly time series.

mondiv

Divide monthly time series

Divides a time series and a monthly time series.

Example

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

```
cdo monsub infile -monavg infile outfile
```

Author

Uwe Schulzweida

2.7.7 Yeararith

Name

yearadd, yearsub, yearmul, yeardiv - Yearly arithmetic

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs simple arithmetic of a time series and one timestep with the same year. For each field in *infile1* the corresponding field of the timestep in *infile2* with the same year is used. The header information in *infile1* have to be the same as in *infile2*. Usually *infile2* is generated by an operator of the module *Yearstat*.

Operators

yearadd

Add yearly time series

Adds a time series and a yearly time series.

yearsab

Subtract yearly time series

Subtracts a time series and a yearly time series.

yearmul

Multiply yearly time series

Multiplies a time series and a yearly time series.

yeardiv

Divide yearly time series

Divides a time series and a yearly time series.

Example

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

```
cdo yearsub infile -yearavg infile outfile
```

Author

Uwe Schulzweida

2.7.8 Yhourarith

Name

yhouradd, yhoursub, yhourmul, yhourdiv - Multi-year hourly arithmetic

Synopsis

cdo <operator> *infile1* *infile2* *outfile*

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 *infile1* the corresponding field of the timestep in *infile2* with the same hour and day of year is used. The input files need to have the same structure with the same variables. Usually *infile2* is generated by an operator of the module *Yhourstat*.

Operators

yhouradd

Add multi-year hourly time series

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

yhoursub

Subtract multi-year hourly time series

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

yhourmul

Multiply multi-year hourly time series

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

yhourdiv

Divide multi-year hourly time series

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

Example

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

```
cdo yhoursub infile -yhouravg infile outfile
```

Author

Uwe Schulzweida

2.7.9 Ydayarith

Name

ydayadd, ydaysub, ydaymul, ydaydiv - Multi-year daily arithmetic

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs simple arithmetic of a time series and one timestep with the same day of year. For each field in **infile1** the corresponding field of the timestep in **infile2** with the same day of year is used. The input files need to have the same structure with the same variables. Usually **infile2** is generated by an operator of the module *Ydaystat*.

Operators

ydayadd

Add multi-year daily time series

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

ydaysub

Subtract multi-year daily time series

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

ydaymul

Multiply multi-year daily time series

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

ydaydiv

Divide multi-year daily time series

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

Example

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

```
cdo ydaysub infile -ydayavg infile outfile
```

Author

Uwe Schulzweida

2.7.10 Ymonarith

Name

ymonadd, ymonsub, ymonmul, ymonddiv - Multi-year monthly arithmetic

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs simple arithmetic of a time series and one timestep with the same month of year. For each field in *infile1* the corresponding field of the timestep in *infile2* with the same month of year is used. The input files need to have the same structure with the same variables. Usually *infile2* 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 with a multi-year monthly time series.

ymondiv

Divide multi-year monthly time series

Divides a time series by a multi-year monthly time series.

Example

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

```
cdo ymonsub infile -ymonavg infile outfile
```

Author

Uwe Schulzweida

2.7.11 Yseasarith

Name

yseasadd, yseassub, yseasmul, yseasdiv - Multi-year seasonal arithmetic

Synopsis

cdo <operator> *infile1 infile2 outfile*

Description

This module performs simple arithmetic of a time series and one timestep with the same season. For each field in *infile1* the corresponding field of the timestep in *infile2* with the same season is used. The input files need to have the same structure with the same variables. Usually *infile2* is generated by an operator of the module *Yseasstat*.

Operators

yseasadd

Add multi-year seasonal time series

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

yseassub

Subtract multi-year seasonal time series

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

yseasmul

Multiply multi-year seasonal time series

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

yseasdiv

Divide multi-year seasonal time series

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

Example

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

```
cdo yseassub infile -yseasavg infile outfile
```

Author

Uwe Schulzweida

2.7.12 Arithdays

Name

muldpm, divdpm, muldpy, divdpy - Arithmetic with days

Synopsis

cdo <operator> *infile outfile*

Description

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

Operators

muldpm

Multiply with days per month

$$o(t, x) = i(t, x) * days_per_month$$

divdpm

Divide by days per month

$$o(t, x) = i(t, x) / days_per_month$$

muldpy

Multiply with days per year

$$o(t, x) = i(t, x) * days_per_year$$

divdpy

Divide by days per year

$$o(t, x) = i(t, x) / days_per_year$$

Author

Uwe Schulzweida

2.7.13 Arithlat

Name

mulcoslat, divcoslat - Arithmetic with latitude

Synopsis

cdo <operator> *infile outfile*

Description

This module multiplies or divides each field element with the cosine of the latitude.

Operators

mulcoslat

Multiply with the cosine of the latitude

$$o(t, x) = i(t, x) * \cos(\text{latitude}(x))$$

divcoslat

Divide by cosine of the latitude

$$o(t, x) = i(t, x) / \cos(\text{latitude}(x))$$

Author

Uwe Schulzweida

2.8 Statistic

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

CDO is using the verification time to identify the time range for temporal statistics. The time bounds are never used!

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

$$\begin{aligned}
 \text{sum} &= \sum_{i=1}^n x_i \\
 \text{mean resp. avg} &= n^{-1} \sum_{i=1}^n x_i \\
 \bar{x} & \\
 \text{mean resp. avg} &= \left(\sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i x_i \\
 \text{weighted by} & \\
 \{w_i, i = 1, \dots, n\} & \\
 \text{Variance} &= n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2 \\
 \text{var} & \\
 \text{var1} &= (n-1)^{-1} \sum_{i=1}^n (x_i - \bar{x})^2 \\
 \text{var weighted by} &= \left(\sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left(x_i - \left(\sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right)^2 \\
 \{w_i, i = 1, \dots, n\} & \\
 \text{Standard deviation} &= \sqrt{n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2} \\
 \text{std} & \\
 s & \\
 \text{std1} &= \sqrt{(n-1)^{-1} \sum_{i=1}^n (x_i - \bar{x})^2} \\
 \text{std weighted by} &= \sqrt{\left(\sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left(x_i - \left(\sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right)^2} \\
 \{w_i, i = 1, \dots, n\} & \\
 \text{median} &= \begin{cases} x_{\frac{n+1}{2}} & \text{if } n \text{ is odd} \\ \frac{1}{2} (x_{\frac{n}{2}} + x_{\frac{n}{2}+1}) & \text{if } n \text{ is even} \end{cases} \\
 \text{Skewness} &= \frac{\sum_{i=1}^n (x_i - \bar{x})^3 / n}{s^3} \\
 \text{skew} & \\
 \text{Kurtosis} &= \frac{\sum_{i=1}^n (x_i - \bar{x})^4 / n}{s^4} \\
 \text{kurt} & \\
 \text{CumulativeRanked} & \\
 \text{ProbabilityScore} &= \int_{-\infty}^{\infty} [H(x_1) - cdf(\{x_2 \dots x_n\})|_r]^2 dr \\
 \text{crps} &
 \end{aligned}$$

with $cdf(X)|_r$ being the cumulative distribution function of $\{x_i, i = 2 \dots n\}$ at r and $H(x)$ the Heaviside function jumping at x .

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

<i>Timcumsum</i>	<i>timcumsum</i>	Cumulative sum over all timesteps
<i>Consecstat</i>	<i>consecsum</i>	Consecutive Sum
	<i>consects</i>	Consecutive Timesteps
<i>Varsstat</i>	<i>varsmin</i>	Variables minimum
	<i>varsmax</i>	Variables maximum
	<i>varsrange</i>	Variables range
	<i>varssum</i>	Variables sum
	<i>varsmean</i>	Variables mean
	<i>varsavg</i>	Variables average
	<i>varsstd</i>	Variables standard deviation
	<i>varsstd1</i>	Variables standard deviation (n-1)
	<i>varsvar</i>	Variables variance
	<i>varsvar1</i>	Variables variance (n-1)
	<i>varsskew</i>	Variables skewness
	<i>varskurt</i>	Variables kurtosis
	<i>varsmedian</i>	Variables median
	<i>varspctl</i>	Variables percentile
<i>Ensstat</i>	<i>ensmin</i>	Ensemble minimum
	<i>ensmax</i>	Ensemble maximum
	<i>ensrange</i>	Ensemble range
	<i>enssum</i>	Ensemble sum
	<i>ensmean</i>	Ensemble mean
	<i>ensavg</i>	Ensemble average
	<i>ensstd</i>	Ensemble standard deviation
	<i>ensstd1</i>	Ensemble standard deviation (n-1)
	<i>ensvar</i>	Ensemble variance
	<i>ensvar1</i>	Ensemble variance (n-1)
	<i>ensskew</i>	Ensemble skewness
	<i>enskurt</i>	Ensemble kurtosis
	<i>ensmedian</i>	Ensemble median
	<i>enspctl</i>	Ensemble percentile
<i>Ensstat2</i>	<i>ensrkhistspace</i>	Ranked Histogram averaged over space
	<i>ensrkhisttime</i>	Ranked Histogram averaged over time
	<i>ensroc</i>	Ensemble Receiver Operating characteristics
<i>Ensval</i>	<i>enscrps</i>	Ensemble CRPS and decomposition
	<i>ensbrs</i>	Ensemble Brier score
<i>Fldstat</i>	<i>fldmin</i>	Field minimum
	<i>fldmax</i>	Field maximum
	<i>fldrang</i>	Field range
	<i>fldsum</i>	Field sum
	<i>fldint</i>	Field integral
	<i>fldmean</i>	Field mean
	<i>fldavg</i>	Field average
	<i>fldstd</i>	Field standard deviation
	<i>fldstd1</i>	Field standard deviation (n-1)
	<i>fldvar</i>	Field variance
	<i>fldvar1</i>	Field variance (n-1)
	<i>fldskew</i>	Field skewness
	<i>fldkurt</i>	Field kurtosis
	<i>fldmedian</i>	Field median
	<i>fldcount</i>	Field count
	<i>fldpctl</i>	Field percentile

continues on next page

Table 13 – continued from previous page

<i>Zonstat</i>	<i>zonmin</i>	Zonal minimum
	<i>zonmax</i>	Zonal maximum
	<i>zonrange</i>	Zonal range
	<i>zonsum</i>	Zonal sum
	<i>zonint</i>	Zonal integral
	<i>zonmean</i>	Zonal mean
	<i>zonavg</i>	Zonal average
	<i>zonstd</i>	Zonal standard deviation
	<i>zonstd1</i>	Zonal standard deviation (n-1)
	<i>zonvar</i>	Zonal variance
	<i>zonvar1</i>	Zonal variance (n-1)
	<i>zonskew</i>	Zonal skewness
	<i>zonkurt</i>	Zonal kurtosis
	<i>zonmedian</i>	Zonal median
	<i>zonpctl</i>	Zonal percentile
<i>Merstat</i>	<i>mermin</i>	Meridional minimum
	<i>mermax</i>	Meridional maximum
	<i>merrange</i>	Meridional range
	<i>mersum</i>	Meridional sum
	<i>mermean</i>	Meridional mean
	<i>meravg</i>	Meridional average
	<i>merstd</i>	Meridional standard deviation
	<i>merstd1</i>	Meridional standard deviation (n-1)
	<i>mervar</i>	Meridional variance
	<i>mervar1</i>	Meridional variance (n-1)
	<i>merskew</i>	Meridional skewness
	<i>merkurt</i>	Meridional kurtosis
	<i>mermedian</i>	Meridional median
	<i>merpctl</i>	Meridional percentile
<i>Gridboxstat</i>	<i>gridboxmin</i>	Gridbox minimum
	<i>gridboxmax</i>	Gridbox maximum
	<i>gridboxrange</i>	Gridbox range
	<i>gridboxsum</i>	Gridbox sum
	<i>gridboxmean</i>	Gridbox mean
	<i>gridboxavg</i>	Gridbox average
	<i>gridboxstd</i>	Gridbox standard deviation
	<i>gridboxstd1</i>	Gridbox standard deviation (n-1)
	<i>gridboxvar</i>	Gridbox variance
	<i>gridboxvar1</i>	Gridbox variance (n-1)
	<i>gridboxskew</i>	Gridbox skewness
	<i>gridboxkurt</i>	Gridbox kurtosis
	<i>gridboxmedian</i>	Gridbox median
<i>Remapstat</i>	<i>remapmin</i>	Remap minimum
	<i>remapmax</i>	Remap maximum
	<i>remaprange</i>	Remap range
	<i>remapsum</i>	Remap sum
	<i>remapmean</i>	Remap mean
	<i>remapavg</i>	Remap average
	<i>remapstd</i>	Remap standard deviation
	<i>remapstd1</i>	Remap standard deviation (n-1)
	<i>remapvar</i>	Remap variance
	<i>remapvar1</i>	Remap variance (n-1)
	<i>remapskew</i>	Remap skewness
	<i>remapkurt</i>	Remap kurtosis
	<i>remapmedian</i>	Remap median

continues on next page

Table 13 – continued from previous page

<i>Vertstat</i>	<i>vertmin</i>	Vertical minimum
	<i>vertmax</i>	Vertical maximum
	<i>vertrange</i>	Vertical range
	<i>vertsum</i>	Vertical sum
	<i>vertmean</i>	Vertical mean
	<i>vertavg</i>	Vertical average
	<i>vertstd</i>	Vertical standard deviation
	<i>vertstd1</i>	Vertical standard deviation (n-1)
	<i>vertvar</i>	Vertical variance
	<i>vertvar1</i>	Vertical variance (n-1)
<i>Timstat</i>	<i>timselmin</i>	Time selection minimum
	<i>timselmax</i>	Time selection maximum
	<i>timselrange</i>	Time selection range
	<i>timselsum</i>	Time selection sum
	<i>timselmean</i>	Time selection mean
	<i>timselavg</i>	Time selection average
	<i>timselstd</i>	Time selection standard deviation
	<i>timselstd1</i>	Time selection standard deviation (n-1)
	<i>timselvar</i>	Time selection variance
	<i>timselvar1</i>	Time selection variance (n-1)
<i>Timselctl</i>	<i>timselctl</i>	Time range percentile
<i>Runstat</i>	<i>runmin</i>	Running minimum
	<i>runmax</i>	Running maximum
	<i>runrange</i>	Running range
	<i>runsum</i>	Running sum
	<i>runmean</i>	Running mean
	<i>runavg</i>	Running average
	<i>runstd</i>	Running standard deviation
	<i>runstd1</i>	Running standard deviation (n-1)
	<i>runvar</i>	Running variance
	<i>runvar1</i>	Running variance (n-1)
<i>Runctl</i>	<i>runctl</i>	Running percentile
<i>Timstat</i>	<i>timmin</i>	Time minimum
	<i>timmax</i>	Time maximum
	<i>timminidx</i>	Index of time minimum
	<i>timmaxidx</i>	Index of time maximum
	<i>timrange</i>	Time range
	<i>timsum</i>	Time sum
	<i>timmean</i>	Time mean
	<i>timavg</i>	Time average
	<i>timstd</i>	Time standard deviation
	<i>timstd1</i>	Time standard deviation (n-1)
	<i>timvar</i>	Time variance
	<i>timvar1</i>	Time variance (n-1)
<i>Timpctl</i>	<i>timpctl</i>	Temporal percentile
<i>Hourstat</i>	<i>hourmin</i>	Hourly minimum
	<i>hourmax</i>	Hourly maximum
	<i>hourrange</i>	Hourly range
	<i>hoursum</i>	Hourly sum
	<i>hourmean</i>	Hourly mean
	<i>houravg</i>	Hourly average
	<i>hourstd</i>	Hourly standard deviation
	<i>hourstd1</i>	Hourly standard deviation (n-1)
	<i>hourvar</i>	Hourly variance
	<i>hourvar1</i>	Hourly variance (n-1)

continues on next page

Table 13 – continued from previous page

<i>Hourpctl</i>	<i>hourpctl</i>	Hourly percentile
<i>Daystat</i>	<i>daymin</i>	Daily minimum
	<i>daymax</i>	Daily maximum
	<i>dayrange</i>	Daily range
	<i>daysum</i>	Daily sum
	<i>daymean</i>	Daily mean
	<i>dayavg</i>	Daily average
	<i>daystd</i>	Daily standard deviation
	<i>daystd1</i>	Daily standard deviation (n-1)
	<i>dayvar</i>	Daily variance
	<i>dayvar1</i>	Daily variance (n-1)
<i>Daypctl</i>	<i>daypctl</i>	Daily percentile
<i>Monstat</i>	<i>monmin</i>	Monthly minimum
	<i>monmax</i>	Monthly maximum
	<i>monrange</i>	Monthly range
	<i>monsum</i>	Monthly sum
	<i>monmean</i>	Monthly mean
	<i>monavg</i>	Monthly average
	<i>monstd</i>	Monthly standard deviation
	<i>monstd1</i>	Monthly standard deviation (n-1)
	<i>monvar</i>	Monthly variance
	<i>monvar1</i>	Monthly variance (n-1)
<i>Monpctl</i>	<i>monpctl</i>	Monthly percentile
<i>Timyearstat</i>	<i>timyearmean</i>	Weighted temporal mean from yearly data
<i>Yearmonstat</i>	<i>yearmonmean</i>	Weighted yearly mean from monthly data
<i>Yearstat</i>	<i>yearmin</i>	Yearly minimum
	<i>yearmax</i>	Yearly maximum
	<i>yearminidx</i>	Index of yearly minimum
	<i>yearmaxidx</i>	Index of yearly maximum
	<i>yearrange</i>	Yearly range
	<i>yearsum</i>	Yearly sum
	<i>yearmean</i>	Yearly mean
	<i>yearavg</i>	Yearly average
	<i>yearstd</i>	Yearly standard deviation
	<i>yearstd1</i>	Yearly standard deviation (n-1)
	<i>yearvar</i>	Yearly variance
	<i>yearvar1</i>	Yearly variance (n-1)
<i>Yearpctl</i>	<i>yearpctl</i>	Yearly percentile
<i>Seasstat</i>	<i>seasmin</i>	Seasonal minimum
	<i>seasmax</i>	Seasonal maximum
	<i>seasrange</i>	Seasonal range
	<i>seassum</i>	Seasonal sum
	<i>seasmean</i>	Seasonal mean
	<i>seasavg</i>	Seasonal average
	<i>seasstd</i>	Seasonal standard deviation
	<i>seasstd1</i>	Seasonal standard deviation (n-1)
	<i>seasvar</i>	Seasonal variance
	<i>seasvar1</i>	Seasonal variance (n-1)
<i>Seaspctl</i>	<i>seaspctl</i>	Seasonal percentile
<i>Yhourstat</i>	<i>yhourmin</i>	Multi-year hourly minimum
	<i>yhourmax</i>	Multi-year hourly maximum
	<i>yhourrange</i>	Multi-year hourly range
	<i>yhoursum</i>	Multi-year hourly sum
	<i>yhourmean</i>	Multi-year hourly mean

continues on next page

Table 13 – continued from previous page

	<i>yhouravg</i>	Multi-year hourly average
	<i>yhourstd</i>	Multi-year hourly standard deviation
	<i>yhourstd1</i>	Multi-year hourly standard deviation (n-1)
	<i>yhourvar</i>	Multi-year hourly variance
	<i>yhourvar1</i>	Multi-year hourly variance (n-1)
<i>Dhourstat</i>	<i>dhourmin</i>	Multi-day hourly minimum
	<i>dhourmax</i>	Multi-day hourly maximum
	<i>dhourrange</i>	Multi-day hourly range
	<i>dhoursun</i>	Multi-day hourly sum
	<i>dhourmean</i>	Multi-day hourly mean
	<i>dhouravg</i>	Multi-day hourly average
	<i>dhourstd</i>	Multi-day hourly standard deviation
	<i>dhourstd1</i>	Multi-day hourly standard deviation (n-1)
	<i>dhourvar</i>	Multi-day hourly variance
	<i>dhourvar1</i>	Multi-day hourly variance (n-1)
<i>Dminutestat</i>	<i>dminutemin</i>	Multi-day by the minute minimum
	<i>dminutemax</i>	Multi-day by the minute maximum
	<i>dminuterange</i>	Multi-day by the minute range
	<i>dminutesun</i>	Multi-day by the minute sum
	<i>dminutemean</i>	Multi-day by the minute mean
	<i>dminuteavg</i>	Multi-day by the minute average
	<i>dminutestd</i>	Multi-day by the minute standard deviation
	<i>dminutestd1</i>	Multi-day by the minute standard deviation (n-1)
	<i>dminutevar</i>	Multi-day by the minute variance
	<i>dminutevar1</i>	Multi-day by the minute variance (n-1)
<i>Ydaystat</i>	<i>ydaymin</i>	Multi-year daily minimum
	<i>ydaymax</i>	Multi-year daily maximum
	<i>ydayrange</i>	Multi-year daily range
	<i>ydaysum</i>	Multi-year daily sum
	<i>ydaymean</i>	Multi-year daily mean
	<i>ydayavg</i>	Multi-year daily average
	<i>ydaystd</i>	Multi-year daily standard deviation
	<i>ydaystd1</i>	Multi-year daily standard deviation (n-1)
	<i>ydayvar</i>	Multi-year daily variance
	<i>ydayvar1</i>	Multi-year daily variance (n-1)
<i>Ydaypctl</i>	<i>ydaypctl</i>	Multi-year daily percentile
<i>Ymonstat</i>	<i>ymonmin</i>	Multi-year monthly minimum
	<i>ymonmax</i>	Multi-year monthly maximum
	<i>ymonrange</i>	Multi-year monthly range
	<i>ymonsum</i>	Multi-year monthly sum
	<i>ymonmean</i>	Multi-year monthly mean
	<i>ymonavg</i>	Multi-year monthly average
	<i>ymonstd</i>	Multi-year monthly standard deviation
	<i>ymonstd1</i>	Multi-year monthly standard deviation (n-1)
	<i>ymonvar</i>	Multi-year monthly variance
	<i>ymonvar1</i>	Multi-year monthly variance (n-1)

continues on next page

Table 13 – continued from previous page

<i>Ymonpctl</i>	<i>ymonpctl</i>	Multi-year monthly percentile
<i>Yseasstat</i>	<i>yseasmin</i>	Multi-year seasonal minimum
	<i>yseasmax</i>	Multi-year seasonal maximum
	<i>yseasrange</i>	Multi-year seasonal range
	<i>yseassum</i>	Multi-year seasonal sum
	<i>yseasmean</i>	Multi-year seasonal mean
	<i>yseasavg</i>	Multi-year seasonal average
	<i>yseasstd</i>	Multi-year seasonal standard deviation
	<i>yseasstd1</i>	Multi-year seasonal standard deviation (n-1)
	<i>yseasvar</i>	Multi-year seasonal variance
	<i>yseasvar1</i>	Multi-year seasonal variance (n-1)
<i>Yseaspctl</i>	<i>yseaspctl</i>	Multi-year seasonal percentile
<i>Ydrunstat</i>	<i>ydrunmin</i>	Multi-year daily running minimum
	<i>ydrunmax</i>	Multi-year daily running maximum
	<i>ydrunsum</i>	Multi-year daily running sum
	<i>ydrunmean</i>	Multi-year daily running mean
	<i>ydrunavg</i>	Multi-year daily running average
	<i>ydrunstd</i>	Multi-year daily running standard deviation
	<i>ydrunstd1</i>	Multi-year daily running standard deviation (n-1)
	<i>ydrunvar</i>	Multi-year daily running variance
	<i>ydrunvar1</i>	Multi-year daily running variance (n-1)
<i>Ydrunpctl</i>	<i>ydrunpctl</i>	Multi-year daily running percentile

2.8.1 Timcumsum

Name

timcumsum - Cumulative sum over all timesteps

Synopsis

cdo timcumsum *infile outfile*

Description

The timcumsum operator calculates the cumulative sum over all timesteps. Missing values are treated as numeric zero when summing.

$$o(t, x) = \text{sum}\{i(t', x), 0 \leq t' \leq t\}$$

Author

Uwe Schulzweida

2.8.2 Consecstat

Name

consecsum, consects - Consecutive timestep periods

Synopsis

cdo <operator> *infile outfile*

Description

This module computes periods over all timesteps in *infile* 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 in-place masking the input field:

```
cdo consects -gtc,20.0 infile1 outfile
```

Author

Ralf Müller

2.8.3 Varsstat

Name

varsmín, varsmáx, varsrang, varssum, varsmean, varsavg, varsstd, varsstd1, varsvar, varsvar1, varsskew, varskurt, varsmedian, varspctl - Statistical values over all variables

Synopsis

cdo <operator> *infile outfile*

cdo varspctl,*pn* *infile outfile*

Description

This module computes statistical values over all variables for each timestep. Depending on the chosen operator the minimum, maximum, range, sum, average, standard deviation, variance, skewness, kurtosis, median or a certain percentile is written to *outfile*. All input variables need to have the same gridsize and the same number of levels.

Operators

varsmín

Variables minimum

For every timestep the minimum over all variables is computed.

varsmáx

Variables maximum

For every timestep the maximum over all variables is computed.

varsrang

Variables range

For every timestep the range over all variables is computed.

varssum

Variables sum

For every timestep the sum over all variables is computed.

varsmean

Variables mean

For every timestep the mean over all variables is computed.

varsavg

Variables average

For every timestep the average over all variables is computed.

varsstd

Variables standard deviation

For every timestep the standard deviation over all variables is computed. Normalize by *n*.

varsstd1

Variables standard deviation (*n*-1)

For every timestep the standard deviation over all variables is computed. Normalize by (*n*-1).

varsvar

Variables variance

For every timestep the variance over all variables is computed. Normalize by *n*.

varsvar1

Variables variance (n-1)

For every timestep the variance over all variables is computed. Normalize by (n-1).

varsskew

Variables skewness

For every timestep the skewness over all variables is computed.

varskurt

Variables kurtosis

For every timestep the kurtosis over all variables is computed.

varsmedian

Variables median

For every timestep the median over all variables is computed.

varsptl

Variables percentile

For every timestep the percentile over all variables is computed.

Parameters**pn**

[FLOAT] Percentile number in {0, ..., 100}

Author

Uwe Schulzweida

2.8.4 Ensstat

Name

ensmin, ensmax, ensrange, enssum, ensmean, ensavg, ensstd, ensstd1, ensvar, ensvar1, ensskew, enskurt, ensmedian, ensctl - Ensemble statistics

Synopsis

cdo [options] <operator> *infile* *outfile*

cdo [options] *ensctl,pn* *infile* *outfile*

Description

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

Operators

ensmin

Ensemble minimum

$$o(t, x) = \mathbf{min}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$$

ensmax

Ensemble maximum

$$o(t, x) = \mathbf{max}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$$

ensrange

Ensemble range

$$o(t, x) = \mathbf{range}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$$

enssum

Ensemble sum

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

ensmean

Ensemble mean

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

ensavg

Ensemble average

$$o(t, x) = \mathbf{avg}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$$

ensstd

Ensemble standard deviation

Normalize by n.

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

ensstd1

Ensemble standard deviation (n-1)

Normalize by (n-1).

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

ensvar

Ensemble variance

Normalize by n.

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

ensvar1

Ensemble variance (n-1)

Normalize by (n-1).

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

ensskew

Ensemble skewness

$$o(t, x) = \mathbf{skew}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$$

enskurt

Ensemble kurtosis

$$o(t, x) = \mathbf{kurt}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$$

ensmedian

Ensemble median

$$o(t, x) = \mathbf{median}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$$

enspctl

Ensemble percentile

$$o(t, x) = \mathbf{pth\ percentile}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$$

Parameters**pn**

[FLOAT] Percentile number in {0, ..., 100}

Options*-0*, *--overwrite* to overwrite existing output file.**Note**

Operators of this module need to open all input files simultaneously. The maximum number of open files depends on the operating system!

Example

To compute the ensemble mean over 6 input files use:

```
cdo ensmean infile1 infile2 infile3 infile4 infile5 infile6 outfile
```

Or shorter with filename substitution:

```
cdo ensmean infile[1-6] outfile
```

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

```
cdo enspctl,50 infile1 infile2 infile3 infile4 infile5 infile6 outfile
```

Author

Uwe Schulzweida

2.8.5 Ensstat2

Name

ensrkhistspace, ensrkhisttime, ensroc - Statistical values over an ensemble

Synopsis

cdo <operator> *obsfile ensfiles outfile*

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 **outfile**. The date and grid information of a timestep in **outfile** 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. Therefore the vertical axis is utilized as the Histogram axis, which prohibits the use of files containing more than one level. The histogram axis has **ensfiles+1** bins with level 0 containing for each grid point the number of observations being smaller as all ensembles and level **ensfiles+1** indicating the number of observations being larger than all ensembles.

ensrkhisttime 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

ensrkhisttime

Ranked Histogram averaged over time

ensrkhistspace

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 outfile
```

Or shorter with filename substitution:

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

Author

Uwe Schulzweida

2.8.6 Ensval

Name

enscrps, ensbrs - Ensemble validation tools

Synopsis

cdo *enscrps rfile infiles outfilebase*

cdo *ensbrs,x rfile infiles outfilebase*

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 *infiles* 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 *outfile* is the date of the first input file. The output files are named as *<outfilebase>.<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 *infiles* with respect to the reference given in *rfile* and the threshold *x* is calculated. In the four output files *<type>* has the following meaning: *brs* for the Brier score wrt threshold *x*; *brsreli* for the Brier score reliability wrt threshold *x*; *brsreso* for the Brier score resolution wrt threshold *x*; *brsunct* for the Brier score uncertainty wrt threshold *x*. In analogy to the CRPS the following relation holds $BRS(x) = RELI(x) - RESO(x) + UNCT(x)$.

The implementation of the decomposition of the CRPS and Brier Score follows

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

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

Operators

enscrps

Ensemble CRPS and decomposition

ensbrs

Ensemble Brier score

Parameters

x

[FLOAT] threshold

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
```

Author

Cedrick Ansorge

2.8.7 fldstat

Name

fldmin, fldmax, fldrange, fldsum, fldint, fldmean, fldavg, fldstd, fldstd1, fldvar, fldvar1, fldskew, fldkurt, fldmedian, fldcount, fldpctl - Statistical values over a field

Synopsis

cdo <operator>[,parameter] *infile outfile*

Description

This module computes statistical values of all input fields. A field is a horizontal layer of a data variable. Depending on the chosen operator, the minimum, maximum, range, sum, integral, average, standard deviation, variance, skewness, kurtosis, median or a certain percentile of the field is written to *outfile*.

Operators

fldmin

Field minimum

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{min}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

fldmax

Field maximum

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{max}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

fldrange

Field range

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{range}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

fldsum

Field sum

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{sum}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

fldint

Field integral

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{sum}\{i(t, x') * cellarea(x'), x_1 \leq x' \leq x_n\}$$

fldmean

Field mean

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{mean}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

weighted by area weights obtained by the input field.

fldavg

Field average

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{avg}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

weighted by area weights obtained by the input field.

fldvar

Field variance

Normalize by n. For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{var}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

weighted by area weights obtained by the input field.

fldvar1

Field variance (n-1)

Normalize by (n-1). For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{var1}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

weighted by area weights obtained by the input field.

fldstd

Field standard deviation

Normalize by n. For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{std}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

weighted by area weights obtained by the input field.

fldstd1

Field standard deviation (n-1)

Normalize by (n-1). For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{std1}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

weighted by area weights obtained by the input field.

fldskew

Field skewness

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{skew}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

fldkurt

Field kurtosis

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{kurt}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

fldmedian

Field median

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{median}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

fldcount

Field count

Number of non-missing values of the field.

fldpctl

Field percentile

For every gridpoint $x_1, \dots, x_n$ of the same field it is:

$$o(t, 1) = \mathbf{pth\ percentile}\{i(t, x'), x_1 \leq x' \leq x_n\}$$

Parameters

verbose

[BOOL] print lon/lat coordinates of min/max values

weights

[BOOL] weights=FALSE disables weighting by grid cell area [default: weights=TRUE]

pn

[FLOAT] Percentile number in {0, ..., 100}

Example

To compute the field mean of all input fields use:

```
cdo fldmean infile outfile
```

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

```
cdo fldpctl,pn=90 infile outfile
```

Author

Uwe Schulzweida

2.8.8 Zonstat

Name

zonmin, zonmax, zonrange, zonsum, zonint, zonmean, zonavg, zonstd, zonstd1, zonvar, zonvar1, zonskew, zonkurt, zonmedian, zonpctl - Zonal statistics

Synopsis

cdo <operator>[,parameters] *infile outfile*

cdo zonpctl,pn *infile outfile*

Description

This module computes zonal statistical values of the input fields. Depending on the chosen operator, the zonal minimum, maximum, range, sum, average, standard deviation, variance, skewness, kurtosis, median or a certain percentile of the field is written to *outfile*. Operators of this module require all variables on the same regular lon/lat grid. The operators zonmean and zonint also support unstructured grids if the latitude intervals are defined using the optional parameter zonaldes.

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.

zonrange

Zonal range

For every latitude the range over all longitudes is computed.

zonsum

Zonal sum

For every latitude the sum over all longitudes is computed. Use the optional parameter zonaldes for data on an unstructured grid.

zonint

Zonal integral

For every latitude the integral over all longitudes is computed. Use the optional parameter zonaldes for data on an unstructured grid.

zonmean

Zonal mean

For every latitude the mean over all longitudes is computed. Use the optional parameter zonaldes for data on an unstructured grid.

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. Normalize by n.

zonvar1

Zonal variance (n-1)

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

zonstd

Zonal standard deviation

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

zonstd1

Zonal standard deviation (n-1)

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

zonskew

Zonal skewness

For every latitude the skewness over all longitudes is computed.

zonkurt

Zonal kurtosis

For every latitude the kurtosis over all longitudes is computed.

zonmedian

Zonal median

For every latitude the median over all longitudes is computed.

zonpctl

Zonal percentile

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

Parameters**pn**

[FLOAT] Percentile number in {0, ..., 100}

zonaldes

[STRING] Description of the zonal latitude bins needed for data on an unstructured grid. A predefined zonal description is zonal_<DY>. DY is the increment of the latitudes in degrees.

Example

To compute the zonal mean of all input fields use:

```
cdo zonmean infile outfile
```

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

```
cdo zonpctl,50 infile outfile
```

Author

Uwe Schulzweida

2.8.9 Merstat

Name

mermin, mermax, merrange, mersum, mermean, meravg, merstd, merstd1, mervar, mervar1, merskew, merkurt, mermedian, merpctl - Meridional statistics

Synopsis

cdo <operator> *infile outfile*

cdo merpctl,*pn* *infile outfile*

Description

This module computes meridional statistical values of the input fields. Depending on the chosen operator, the meridional minimum, maximum, range, sum, average, standard deviation, variance, skewness, kurtosis, median or a certain percentile of the field is written to *outfile*. Operators of this module require all variables on the same regular lon/lat grid.

Operators

mermin

Meridional minimum

For every longitude the minimum over all latitudes is computed.

mermax

Meridional maximum

For every longitude the maximum over all latitudes is computed.

merrange

Meridional range

For every longitude the range 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. Normalize by n.

mervar1

Meridional variance (n-1)

For every longitude the variance over all latitudes is computed. Normalize by (n-1).

merstd

Meridional standard deviation

For every longitude the standard deviation over all latitudes is computed. Normalize by n.

merstd1

Meridional standard deviation (n-1)

For every longitude the standard deviation over all latitudes is computed. Normalize by (n-1).

merskew

Meridional skewness

For every longitude the skewness over all latitudes is computed.

merkurt

Meridional kurtosis

For every longitude the kurtosis over all latitudes is computed.

mermedian

Meridional median

For every longitude the median over all latitudes is computed.

merpctl

Meridional percentile

For every longitude the pth percentile over all latitudes is computed.

Parameters**pn**

[FLOAT] Percentile number in {0, ..., 100}

Example

To compute the meridional mean of all input fields use:

```
cdo mermean infile outfile
```

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

```
cdo merpctl,50 infile outfile
```

Author

Uwe Schulzweida

2.8.10 Gridboxstat

Name

gridboxmin, gridboxmax, gridboxrange, gridboxsum, gridboxmean, gridboxavg, gridboxstd, gridboxstd1, gridboxvar, gridboxvar1, gridboxskew, gridboxkurt, gridboxmedian - Statistical values over grid boxes

Synopsis

cdo <operator>.parameters *infile outfile*

Description

This module computes statistical values over surrounding grid boxes. Depending on the chosen operator, the minimum, maximum, range, sum, average, standard deviation, variance, skewness, kurtosis or median of the neighboring grid boxes is written to *outfile*. All gridbox operators only work on quadrilateral curvilinear grids.

Operators

gridboxmin

Gridbox minimum

Minimum value of the selected grid boxes.

gridboxmax

Gridbox maximum

Maximum value of the selected grid boxes.

gridboxrange

Gridbox range

Range (max-min value) of the selected grid boxes.

gridboxsum

Gridbox sum

Sum of the selected grid boxes.

gridboxmean

Gridbox mean

Mean of the selected grid boxes.

gridboxavg

Gridbox average

Average of the selected grid boxes.

gridboxvar

Gridbox variance

Variance of the selected grid boxes. Normalize by n.

gridboxvar1

Gridbox variance (n-1)

Variance of the selected grid boxes. Normalize by (n-1).

gridboxstd

Gridbox standard deviation

Standard deviation of the selected grid boxes. Normalize by n.

gridboxstd1

Gridbox standard deviation (n-1)

Standard deviation of the selected grid boxes. Normalize by (n-1).

gridboxskew

Gridbox skewness

Skewness of the selected grid boxes.

gridboxkurt

Gridbox kurtosis

Kurtosis of the selected grid boxes.

gridboxmedian

Gridbox median

Median of the selected grid boxes.

Parameters

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 infile outfile
```

Author

Uwe Schulzweida

2.8.11 Remapstat

Name

remapmin, remapmax, remaprange, remapsum, remapmean, remapavg, remapstd, remapstd1, remapvar, remapvar1, remapskew, remapkurt, remapmedian - Remaps source points to target cells

Synopsis

cdo <operator>,*parameters infile outfile*

Description

This module maps source points to target cells by calculating a statistical value from the source points. Each target cell contains the statistical value from all source points within that target cell. If there are no source points within a target cell, it gets a missing value. Depending on the chosen operator the minimum, maximum, range, sum, average, variance, standard deviation, skewness, kurtosis or median of source points is computed.

Operators

remapmin

Remap minimum

Minimum value of the source points.

remapmax

Remap maximum

Maximum value of the source points.

remaprange

Remap range

Range (max-min value) of the source points.

remapsum

Remap sum

Sum of the source points.

remapmean

Remap mean

Mean of the source points.

remapavg

Remap average

Average of the source points.

remapvar

Remap variance

Variance of the source points. Normalize by n.

remapvar1

Remap variance (n-1)

Variance of the source points. Normalize by (n-1).

remapstd

Remap standard deviation

Standard deviation of the source points. Normalize by n.

remapstd1

Remap standard deviation (n-1)

Standard deviation of the source points. Normalize by (n-1).

remapskew

Remap skewness

Skewness of the source points.

remapkurt

Remap kurtosis

Kurtosis of the source points.

remapmedian

Remap median

Median of the source points.

Parameters**grid**

[STRING] Target grid description file or name

Example

To compute the mean over source points within the target cells, use:

```
cdo remapmean,<targetgrid> infile outfile
```

If some of the target cells contain missing values, use the Operator *setmisstonn* to fill these missing values with the nearest neighbor cell:

```
cdo setmisstonn -remapmean,<targetgrid> infile outfile
```

Author

Uwe Schulzweida

2.8.12 Vertstat

Name

vertmin, vertmax, vertrange, vertsum, vertmean, vertavg, vertstd, vertstd1, vertvar, vertvar1 - Vertical statistics

Synopsis

cdo <operator> *infile outfile*

Description

This module computes statistical values over all levels of the input variables. According to chosen operator the vertical minimum, maximum, range, sum, average, variance or standard deviation is written to *outfile*. Weighting based on layer thickness can be disabled with the parameter `weights=FALSE`.

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.

vertrange

Vertical range

For every gridpoint the range 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 weighted mean over all levels is computed.

vertavg

Vertical average

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

vertvar

Vertical variance

For every gridpoint the weighted variance over all levels is computed. Normalize by *n*.

vertvar1

Vertical variance (*n*-1)

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

vertstd

Vertical standard deviation

For every gridpoint the weighted standard deviation over all levels is computed. Normalize by *n*.

vertstd1

Vertical standard deviation (*n*-1)

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

Parameters

weights

[BOOL] weights=FALSE disables weighting with layer thickness [default: weights=TRUE]

Example

To compute the vertical sum of all input variables use:

```
cdo vertsum infile outfile
```

Author

Uwe Schulzweida

2.8.13 Timselstat

Name

timselmin, timselmax, timselrange, timselsum, timselmean, timselavg, timselstd, timselstd1, timselvar, timselvar1
- Time range statistics

Synopsis

cdo <operator>,*nsets*[,*noffset*[,*nskip*]] *infile outfile*

Description

This module computes statistical values for a selected number of timesteps. According to the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of the selected timesteps is written to *outfile*. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

Operators

timselmin

Time selection minimum

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

$$o(t, x) = \min\{i(t', x), t_1 \leq t' \leq t_n\}$$

timselmax

Time selection maximum

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

$$o(t, x) = \max\{i(t', x), t_1 \leq t' \leq t_n\}$$

timselrange

Time selection range

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

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

timselsum

Time selection sum

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

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

timselmean

Time selection mean

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

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

timselavg

Time selection average

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

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

timselvar

Time selection variance

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

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

timselvar1

Time selection variance (n-1)

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

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

timselstd

Time selection standard deviation

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

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

timselstd1

Time selection standard deviation (n-1)

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

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

Parameters**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 months have to be skipped:

```
cdo timselmean,3,2 infile outfile
```

Author

Uwe Schulzweida

2.8.14 Timselpctl

Name

timselpctl - Time range percentile

Synopsis

cdo timselpctl,*pn,nsets[,noffset[,nskip]] infile1 infile2 infile3 outfile*

Description

This operator computes percentile values over a selected number of timesteps in *infile1*. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, 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 *infile2* and *infile3* should be the result of corresponding *timselmin* and *timselmax* operations, respectively. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile1*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

nsets

[INTEGER] Number of input timesteps for each output timestep

noffset

[INTEGER] Number of input timesteps skipped before the first timestep range (optional)

nskip

[INTEGER] Number of input timesteps skipped between timestep ranges (optional)

Environment

CDO_PCTL_NBINS sets the number of histogram bins (default: *CDO_PCTL_NBINS*=101).

Author

Uwe Schulzweida

2.8.15 Runstat

Name

runmin, runmax, runrange, runsum, runmean, runavg, runstd, runstd1, runvar, runvar1 - Running statistics

Synopsis

cdo <operator>,nts infile outfile

Description

This module computes running statistical values over a selected number of timesteps. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of a selected number of consecutive timesteps read from **infile** is written to **outfile**. The time of **outfile** is determined by the time in the middle of all contributing timesteps of **infile**. This can be changed with the **CDO** option **--timestat_date** <first|middle|last>.

Operators

runmin

Running minimum

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

runmax

Running maximum

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

runrange

Running range

$$o(t + (nts - 1)/2, x) = \text{range}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$$

runsum

Running sum

$$o(t + (nts - 1)/2, x) = \text{sum}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$$

runmean

Running mean

$$o(t + (nts - 1)/2, x) = \text{mean}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$$

runavg

Running average

$$o(t + (nts - 1)/2, x) = \text{avg}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$$

runvar

Running variance

Normalize by n.

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

runvar1

Running variance (n-1)

Normalize by (n-1).

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

runstd

Running standard deviation

Normalize by n.

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

runstd1

Running standard deviation (n-1)

Normalize by (n-1).

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

Parameters**nts**

[INTEGER] Number of timesteps

Example

To compute the running mean over 9 timesteps use:

```
cdo runmean,9 infile outfile
```

Author

Uwe Schulzweida

2.8.16 Runpctl

Name

runpctl - Running percentile

Synopsis

cdo runpctl,*pn,nts* *infile outfile*

Description

This module computes running percentiles over a selected number of timesteps in **infile**. The time of **outfile** is determined by the time in the middle of all contributing timesteps of **infile**. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

nts

[INTEGER] Number of timesteps

Example

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

```
cdo runpctl,50,9 infile outfile
```

Author

Uwe Schulzweida

2.8.17 Timstat

Name

timmin, timmax, timminidx, timmaxidx, timrange, timsum, timmean, timavg, timstd, timstd1, timvar, timvar1 - Statistical values over all timesteps

Synopsis

cdo [options] <operator> *infile outfile*

Description

This module computes statistical values over all timesteps in *infile*. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of all timesteps read from *infile* is written to *outfile*. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile*. This can be changed with the **CDO** option `--timstat_date <first|middle|last>`.

Operators

timmin

Time minimum

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

timmax

Time maximum

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

timminidx

Index of time minimum

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

timmaxidx

Index of time maximum

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

timrange

Time range

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

timsum

Time sum

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

timmean

Time mean

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

timavg

Time average

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

timvar

Time variance

Normalize by n.

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

timvar1

Time variance (n-1)

Normalize by (n-1).

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

timstd

Time standard deviation

Normalize by n.

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

timstd1

Time standard deviation (n-1)

Normalize by (n-1).

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

Options

-S, *--diagnostic* to create a diagnostic output stream with the number of non missing values for each output period.

-p, *--async_read true* to read input data asynchronously.

Example

To compute the mean over all input timesteps use:

```
cdo timmean infile outfile
```

Author

Uwe Schulzweida

2.8.18 Timpctl

Name

timpctl - Temporal percentile

Synopsis

cdo timpctl,*pn* *infile1* *infile2* *infile3* *outfile*

Description

This operator computes percentiles over all timesteps in *infile1*. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable `CDO_PCTL_NBINS`. The files *infile2* and *infile3* should be the result of corresponding *timmin* and *timmax* operations, respectively. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile1*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

`CDO_PCTL_NBINS` sets the number of histogram bins (default: `CDO_PCTL_NBINS=101`).

Example

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

```
cdo timmin infile minfile
cdo timmax infile maxfile
cdo timpctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo timpctl,90 infile -timmin infile -timmax infile outfile
```

Author

Uwe Schulzweida

2.8.19 Hourstat

Name

hourmin, hourmax, hourrange, hoursum, hourmean, houravg, hourstd, hourstd1, hourvar, hourvar1 - Hourly statistics

Synopsis

cdo [options] <operator> *infile outfile*

Description

This module computes statistical values over timesteps of the same hour. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same hour is written to *outfile*. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

Operators

hourmin

Hourly minimum

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

$$o(t, x) = \min\{i(t', x), t_1 \leq t' \leq t_n\}$$

hourmax

Hourly maximum

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

$$o(t, x) = \max\{i(t', x), t_1 \leq t' \leq t_n\}$$

hourrange

Hourly range

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

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

hoursum

Hourly sum

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

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

hourmean

Hourly mean

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

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

houravg

Hourly average

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

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

hourvar

Hourly variance

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

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

hourvar1

Hourly variance (n-1)

Normalize by (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is:

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

hourstd

Hourly standard deviation

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

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

hourstd1

Hourly standard deviation (n-1)

Normalize by (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same hour it is:

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

Options

`-S`, `--diagnostic` to create a diagnostic output stream with the number of non missing values for each output period.

`-p`, `--async_read true` to read input data asynchronously.

Example

To compute the hourly mean of a time series use:

```
cdo hourmean infile outfile
```

Author

Uwe Schulzweida

2.8.20 Hourpctl

Name

hourpctl - Hourly percentile

Synopsis

cdo hourpctl,*pn* *infile1* *infile2* *infile3* *outfile*

Description

This operator computes percentiles over all timesteps of the same hour in *infile1*. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable `CDO_PCTL_NBINS`. The files *infile2* and *infile3* should be the result of corresponding *hourmin* and *hourmax* operations, respectively. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile1*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

`CDO_PCTL_NBINS` sets the number of histogram bins (default: `CDO_PCTL_NBINS=101`).

Example

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

```
cdo hourmin infile minfile
cdo hourmax infile maxfile
cdo hourpctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo hourpctl,90 infile -hourmin infile -hourmax infile outfile
```

Author

Uwe Schulzweida

2.8.21 Daystat

Name

daymin, daymax, dayrange, daysum, daymean, dayavg, daystd, daystd1, dayvar, dayvar1 - Daily statistics

Synopsis

cdo [options] <operator> *infile outfile*

Description

This module computes statistical values over timesteps of the same day. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same day is written to *outfile*. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`. Set the parameter `complete_only=TRUE` to process the last day only when it is complete.

Operators

daymin

Daily minimum

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

$$o(t, x) = \min\{i(t', x), t_1 \leq t' \leq t_n\}$$

daymax

Daily maximum

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

$$o(t, x) = \max\{i(t', x), t_1 \leq t' \leq t_n\}$$

dayrange

Daily range

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

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

daysum

Daily sum

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

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

daymean

Daily mean

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

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

dayavg

Daily average

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

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

dayvar

Daily variance

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

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

dayvar1

Daily variance (n-1)

Normalize by (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is:

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

daystd

Daily standard deviation

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

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

daystd1

Daily standard deviation (n-1)

Normalize by (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same day it is:

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

Parameters**complete_only**

[BOOL] Process the last day only when it is complete

Options

-S, *--diagnostic* to create a diagnostic output stream with the number of non missing values for each output period.

-p, *--async_read true* to read input data asynchronously.

Example

To compute the daily mean of a time series use:

```
cdo daymean infile outfile
```

Author

Uwe Schulzweida

2.8.22 Daypctl

Name

daypctl - Daily percentile

Synopsis

cdo daypctl,*pn infile1 infile2 infile3 outfile*

Description

This operator computes percentiles over all timesteps of the same day in *infile1*. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable *CDO_PCTL_NBINS*. The files *infile2* and *infile3* should be the result of corresponding *daymin* and *daymax* operations, respectively. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile1*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

CDO_PCTL_NBINS sets the number of histogram bins (default: *CDO_PCTL_NBINS*=101).

Example

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

```
cdo daymin infile minfile
cdo daymax infile maxfile
cdo daypctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo daypctl,90 infile -daymin infile -daymax infile outfile
```

Author

Uwe Schulzweida

2.8.23 Monstat

Name

monmin, monmax, monrange, monsum, monmean, monavg, monstd, monstd1, monvar, monvar1 - Monthly statistics

Synopsis

cdo [options] <operator>[.parameters] *infile outfile*

Description

This module computes statistical values over timesteps of the same month. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same month is written to *outfile*. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

Operators

monmin

Monthly minimum

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

$$o(t, x) = \min\{i(t', x), t_1 \leq t' \leq t_n\}$$

monmax

Monthly maximum

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

$$o(t, x) = \max\{i(t', x), t_1 \leq t' \leq t_n\}$$

monrange

Monthly range

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

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

monsum

Monthly sum

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

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

monmean

Monthly mean

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

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

monavg

Monthly average

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

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

monvar

Monthly variance

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

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

monvar1

Monthly variance (n-1)

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

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

monstd

Monthly standard deviation

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

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

monstd1

Monthly standard deviation (n-1)

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

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

Parameters**complete_only**

[BOOL] Process the last month only if it is complete

Options

-S, *--diagnostic* to create a diagnostic output stream with the number of non missing values for each output period.

-p, *--async_read true* to read input data asynchronously.

Example

To compute the monthly mean of a time series use:

```
cdo monmean infile outfile
```

Author

Uwe Schulzweida

2.8.24 Monpctl

Name

monpctl - Monthly percentile

Synopsis

cdo monpctl,*pn infile1 infile2 infile3 outfile*

Description

This operator computes percentiles over all timesteps of the same month in *infile1*. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable `CDO_PCTL_NBINS`. The files *infile2* and *infile3* should be the result of corresponding *monmin* and *monmax* operations, respectively. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile1*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

`CDO_PCTL_NBINS` sets the number of histogram bins (default: `CDO_PCTL_NBINS=101`).

Example

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

```
cdo monmin infile minfile
cdo monmax infile maxfile
cdo monpctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo monpctl,90 infile -monmin infile -monmax infile outfile
```

Author

Uwe Schulzweida

2.8.25 Timyearstat

Name

timyearmean - Weighted temporal mean from yearly data

Synopsis

cdo timyearmean *infile outfile*

Description

This operator computes the weighted temporal mean of a yearly time series. Each year is weighted with the number of days per year. The time of **outfile** is determined by the time in the middle of all contributing timesteps of **infile**. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

Example

To compute the weighted temporal mean of a yearly time series use:

```
cdo timyearmean infile outfile
```

Author

Uwe Schulzweida

2.8.26 Yearmonstat

Name

yearmonmean - Weighted yearly mean from monthly data

Synopsis

cdo yearmonmean *infile outfile*

Description

This operator computes the weighted yearly mean of a monthly time series. Each month is weighted with the number of days per month. The time of **outfile** is determined by the time in the middle of all contributing timesteps of **infile**. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

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

Example

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

```
cdo yearmonmean infile outfile
```

Author

Uwe Schulzweida

2.8.27 Yearstat

Name

yearmin, yearmax, yearminidx, yearmaxidx, yearrange, yearsum, yearmean, yearavg, yearstd, yearstd1, yearvar, yearvar1 - Yearly statistics

Synopsis

cdo [options] <operator> *infile outfile*

Description

This module computes statistical values over timesteps of the same year. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same year is written to *outfile*. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`. Set the parameter `complete_only=TRUE` to process the last year only when it is complete.

Operators

yearmin

Yearly minimum

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

$$o(t, x) = \min\{i(t', x), t_1 \leq t' \leq t_n\}$$

yearmax

Yearly maximum

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

$$o(t, x) = \max\{i(t', x), t_1 \leq t' \leq t_n\}$$

yearminidx

Index of yearly minimum

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

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

yearmaxidx

Index of yearly maximum

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

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

yearrange

Yearly range

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

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

yearsum

Yearly sum

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

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

yearmean

Yearly mean

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

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

yearavg

Yearly average

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

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

yearvar

Yearly variance

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

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

yearvar1

Yearly variance (n-1)

Normalize by (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is:

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

yearstd

Yearly standard deviation

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

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

yearstd1

Yearly standard deviation (n-1)

Normalize by (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same year it is:

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

Parameters**complete_only**

[BOOL] Process the last year only when it is complete

Options

-S, **--diagnostic** to create a diagnostic output stream with the number of non missing values for each output period.

-p, **--async_read true** to read input data asynchronously.

Note

The operators yearmean and yearavg compute only arithmetical means!

Example

To compute the yearly mean of a time series use:

```
cdo yearmean infile outfile
```

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

```
cdo yearmonmean infile outfile
```

Author

Uwe Schulzweida

2.8.28 Yearpctl

Name

yearpctl - Yearly percentile

Synopsis

cdo yearpctl,*pn* *infile1* *infile2* *infile3* *outfile*

Description

This operator computes percentiles over all timesteps of the same year in *infile1*. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable `CDO_PCTL_NBINS`. The files *infile2* and *infile3* should be the result of corresponding *yearmin* and *yearmax* operations, respectively. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile1*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

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

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

`CDO_PCTL_NBINS` sets the number of histogram bins (default: `CDO_PCTL_NBINS=101`).

Example

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

```
cdo yearmin infile minfile
cdo yearmax infile maxfile
cdo yearpctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo yearpctl,90 infile -yearmin infile -yearmax infile outfile
```

Author

Uwe Schulzweida

2.8.29 Seasstat

Name

seasmin, seasmax, seasrange, seassum, seasmean, seasavg, seasstd, seasstd1, seasvar, seasvar1 - Seasonal statistics

Synopsis

cdo <operator> *infile outfile*

Description

This module computes statistical values over timesteps of the same meteorological season. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same season is written to *outfile*. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`. 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 $t_1, \dots, t_n$ of timesteps of the same season it is:

$$o(t, x) = \min\{i(t', x), t_1 \leq t' \leq t_n\}$$

seasmax

Seasonal maximum

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

$$o(t, x) = \max\{i(t', x), t_1 \leq t' \leq t_n\}$$

seasrange

Seasonal range

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

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

seassum

Seasonal sum

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

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

seasmean

Seasonal mean

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

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

seasavg

Seasonal average

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

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

seasvar

Seasonal variance

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

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

seasvar1

Seasonal variance (n-1)

Normalize by (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is:

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

seasstd

Seasonal standard deviation

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

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

seasstd1

Seasonal standard deviation (n-1)

Normalize by (n-1). For every adjacent sequence $t_1, \dots, t_n$ of timesteps of the same season it is:

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

Example

To compute the seasonal mean of a time series use:

```
cdo seasmean infile outfile
```

Author

Uwe Schulzweida

2.8.30 Seaspctl

Name

seaspctl - Seasonal percentile

Synopsis

cdo seaspctl,*pn* *infile1* *infile2* *infile3* *outfile*

Description

This operator computes percentiles over all timesteps in *infile1* of the same season. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable `CDO_PCTL_NBINS`. The files *infile2* and *infile3* should be the result of corresponding *seasmin* and *seasmax* operations, respectively. The time of *outfile* is determined by the time in the middle of all contributing timesteps of *infile1*. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`. Be careful about the first and the last output timestep, they may be incorrect values if the seasons have incomplete timesteps.

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

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

`CDO_PCTL_NBINS` sets the number of histogram bins (default: `CDO_PCTL_NBINS=101`).

Example

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

```
cdo seasmin infile minfile
cdo seasmax infile maxfile
cdo seaspctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo seaspctl,90 infile -seasmin infile -seasmax infile outfile
```

Author

Uwe Schulzweida

2.8.31 Yhourstat

Name

yhourmin, yhourmax, yhourrange, yhoursum, yhourmean, yhouravg, yhourstd, yhourstd1, yhourvar, yhourvar1 - Multi-year hourly statistics

Synopsis

cdo <operator> *infile outfile*

Description

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

Operators

yhourmin

Multi-year hourly minimum

$$\begin{aligned}o(0001, x) &= \mathbf{min}\{i(t, x), \text{day}(i(t)) = 0001\} \\ &\vdots \\ o(8784, x) &= \mathbf{min}\{i(t, x), \text{day}(i(t)) = 8784\}\end{aligned}$$

yhourmax

Multi-year hourly maximum

$$\begin{aligned}o(0001, x) &= \mathbf{max}\{i(t, x), \text{day}(i(t)) = 0001\} \\ &\vdots \\ o(8784, x) &= \mathbf{max}\{i(t, x), \text{day}(i(t)) = 8784\}\end{aligned}$$

yhourrange

Multi-year hourly range

$$\begin{aligned}o(0001, x) &= \mathbf{range}\{i(t, x), \text{day}(i(t)) = 0001\} \\ &\vdots \\ o(8784, x) &= \mathbf{range}\{i(t, x), \text{day}(i(t)) = 8784\}\end{aligned}$$

yhoursum

Multi-year hourly sum

$$\begin{aligned}o(0001, x) &= \mathbf{sum}\{i(t, x), \text{day}(i(t)) = 0001\} \\ &\vdots \\ o(8784, x) &= \mathbf{sum}\{i(t, x), \text{day}(i(t)) = 8784\}\end{aligned}$$

yhourmean

Multi-year hourly mean

$$\begin{aligned}o(0001, x) &= \mathbf{mean}\{i(t, x), \text{day}(i(t)) = 0001\} \\ &\vdots \\ o(8784, x) &= \mathbf{mean}\{i(t, x), \text{day}(i(t)) = 8784\}\end{aligned}$$

yhouravg

Multi-year hourly average

$$\begin{aligned}
 o(0001, x) &= \mathbf{avg}\{i(t, x), \text{day}(i(t)) = 0001\} \\
 &\vdots \\
 o(8784, x) &= \mathbf{avg}\{i(t, x), \text{day}(i(t)) = 8784\}
 \end{aligned}$$

yhourvar

Multi-year hourly variance

Normalize by n.

$$\begin{aligned}
 o(0001, x) &= \mathbf{var}\{i(t, x), \text{day}(i(t)) = 0001\} \\
 &\vdots \\
 o(8784, x) &= \mathbf{var}\{i(t, x), \text{day}(i(t)) = 8784\}
 \end{aligned}$$

yhourvar1

Multi-year hourly variance (n-1)

Normalize by (n-1).

$$\begin{aligned}
 o(0001, x) &= \mathbf{var1}\{i(t, x), \text{day}(i(t)) = 0001\} \\
 &\vdots \\
 o(8784, x) &= \mathbf{var1}\{i(t, x), \text{day}(i(t)) = 8784\}
 \end{aligned}$$

yhourstd

Multi-year hourly standard deviation

Normalize by n.

$$\begin{aligned}
 o(0001, x) &= \mathbf{std}\{i(t, x), \text{day}(i(t)) = 0001\} \\
 &\vdots \\
 o(8784, x) &= \mathbf{std}\{i(t, x), \text{day}(i(t)) = 8784\}
 \end{aligned}$$

yhourstd1

Multi-year hourly standard deviation (n-1)

Normalize by (n-1).

$$\begin{aligned}
 o(0001, x) &= \mathbf{std1}\{i(t, x), \text{day}(i(t)) = 0001\} \\
 &\vdots \\
 o(8784, x) &= \mathbf{std1}\{i(t, x), \text{day}(i(t)) = 8784\}
 \end{aligned}$$

Example

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

```
cdo yhourmean infile outfile
```

Author

Uwe Schulzweida

2.8.32 Dhourstat

Name

dhourmin, dhourmax, dhourrange, dhoursum, dhourmean, dhouravg, dhourstd, dhourstd1, dhourvar, dhourvar1 - Multi-day hourly statistics

Synopsis

cdo <operator> *infile outfile*

Description

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

Operators

dhourmin

Multi-day hourly minimum

$$\begin{aligned} o(01, x) &= \min\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(24, x) &= \min\{i(t, x), \text{day}(i(t)) = 24\} \end{aligned}$$

dhourmax

Multi-day hourly maximum

$$\begin{aligned} o(01, x) &= \max\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(24, x) &= \max\{i(t, x), \text{day}(i(t)) = 24\} \end{aligned}$$

dhourrange

Multi-day hourly range

$$\begin{aligned} o(01, x) &= \text{range}\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(24, x) &= \text{range}\{i(t, x), \text{day}(i(t)) = 24\} \end{aligned}$$

dhoursum

Multi-day hourly sum

$$\begin{aligned} o(01, x) &= \text{sum}\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(24, x) &= \text{sum}\{i(t, x), \text{day}(i(t)) = 24\} \end{aligned}$$

dhourmean

Multi-day hourly mean

$$\begin{aligned} o(01, x) &= \text{mean}\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(24, x) &= \text{mean}\{i(t, x), \text{day}(i(t)) = 24\} \end{aligned}$$

dhouravg

Multi-day hourly average

$$\begin{aligned}
 o(01, x) &= \mathbf{avg}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\
 &\vdots \\
 o(24, x) &= \mathbf{avg}\{i(t, x), \mathbf{day}(i(t)) = 24\}
 \end{aligned}$$

dhourvar

Multi-day hourly variance

Normalize by n.

$$\begin{aligned}
 o(01, x) &= \mathbf{var}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\
 &\vdots \\
 o(24, x) &= \mathbf{var}\{i(t, x), \mathbf{day}(i(t)) = 24\}
 \end{aligned}$$

dhourvar1

Multi-day hourly variance (n-1)

Normalize by (n-1).

$$\begin{aligned}
 o(01, x) &= \mathbf{var1}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\
 &\vdots \\
 o(24, x) &= \mathbf{var1}\{i(t, x), \mathbf{day}(i(t)) = 24\}
 \end{aligned}$$

dhourstd

Multi-day hourly standard deviation

Normalize by n.

$$\begin{aligned}
 o(01, x) &= \mathbf{std}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\
 &\vdots \\
 o(24, x) &= \mathbf{std}\{i(t, x), \mathbf{day}(i(t)) = 24\}
 \end{aligned}$$

dhourstd1

Multi-day hourly standard deviation (n-1)

Normalize by (n-1).

$$\begin{aligned}
 o(01, x) &= \mathbf{std1}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\
 &\vdots \\
 o(24, x) &= \mathbf{std1}\{i(t, x), \mathbf{day}(i(t)) = 24\}
 \end{aligned}$$

Author

Uwe Schulzweida

2.8.33 Dminutestat

Name

dminutemin, dminutemax, dminuterange, dminutesum, dminutemean, dminuteavg, dminutestd, dminutestd1, dminutevar, dminutevar1 - Multi-day by the minute statistics

Synopsis

cdo <operator> *infile outfile*

Description

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

Operators

dminutemin

Multi-day by the minute minimum

$$\begin{aligned}o(01, x) &= \min\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \min\{i(t, x), \text{day}(i(t)) = 1440\}\end{aligned}$$

dminutemax

Multi-day by the minute maximum

$$\begin{aligned}o(01, x) &= \max\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \max\{i(t, x), \text{day}(i(t)) = 1440\}\end{aligned}$$

dminuterange

Multi-day by the minute range

$$\begin{aligned}o(01, x) &= \text{range}\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \text{range}\{i(t, x), \text{day}(i(t)) = 1440\}\end{aligned}$$

dminutesum

Multi-day by the minute sum

$$\begin{aligned}o(01, x) &= \text{sum}\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \text{sum}\{i(t, x), \text{day}(i(t)) = 1440\}\end{aligned}$$

dminutemean

Multi-day by the minute mean

$$\begin{aligned}o(01, x) &= \text{mean}\{i(t, x), \text{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \text{mean}\{i(t, x), \text{day}(i(t)) = 1440\}\end{aligned}$$

dminuteavg

Multi-day by the minute average

$$\begin{aligned} o(01, x) &= \mathbf{avg}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \mathbf{avg}\{i(t, x), \mathbf{day}(i(t)) = 1440\} \end{aligned}$$

dminutevar

Multi-day by the minute variance

Normalize by n.

$$\begin{aligned} o(01, x) &= \mathbf{var}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \mathbf{var}\{i(t, x), \mathbf{day}(i(t)) = 1440\} \end{aligned}$$

dminutevar1

Multi-day by the minute variance (n-1)

Normalize by (n-1).

$$\begin{aligned} o(01, x) &= \mathbf{var1}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \mathbf{var1}\{i(t, x), \mathbf{day}(i(t)) = 1440\} \end{aligned}$$

dminutestd

Multi-day by the minute standard deviation

Normalize by n.

$$\begin{aligned} o(01, x) &= \mathbf{std}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \mathbf{std}\{i(t, x), \mathbf{day}(i(t)) = 1440\} \end{aligned}$$

dminutestd1

Multi-day by the minute standard deviation (n-1)

Normalize by (n-1).

$$\begin{aligned} o(01, x) &= \mathbf{std1}\{i(t, x), \mathbf{day}(i(t)) = 01\} \\ &\vdots \\ o(1440, x) &= \mathbf{std1}\{i(t, x), \mathbf{day}(i(t)) = 1440\} \end{aligned}$$

Author

Uwe Schulzweida

2.8.34 Ydaystat

Name

ydaymin, ydaymax, ydayrange, ydaysum, ydaymean, ydayavg, ydaystd, ydaystd1, ydayvar, ydayvar1 - Multi-year daily statistics

Synopsis

cdo <operator> *infile outfile*

Description

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

Operators

ydaymin

Multi-year daily minimum

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

ydaymax

Multi-year daily maximum

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

ydayrange

Multi-year daily range

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

ydaysum

Multi-year daily sum

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

ydaymean

Multi-year daily mean

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

ydayavg

Multi-year daily average

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

ydayvar

Multi-year daily variance

Normalize by n.

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

ydayvar1

Multi-year daily variance (n-1)

Normalize by (n-1).

$$\begin{aligned}
 o(001, x) &= \mathbf{var1}\{i(t, x), \mathbf{day}(i(t)) = 001\} \\
 &\vdots \\
 o(366, x) &= \mathbf{var1}\{i(t, x), \mathbf{day}(i(t)) = 366\}
 \end{aligned}$$

ydaystd

Multi-year daily standard deviation

Normalize by n.

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

ydaystd1

Multi-year daily standard deviation (n-1)

Normalize by (n-1).

$$\begin{aligned}
 o(001, x) &= \mathbf{std1}\{i(t, x), \mathbf{day}(i(t)) = 001\} \\
 &\vdots \\
 o(366, x) &= \mathbf{std1}\{i(t, x), \mathbf{day}(i(t)) = 366\}
 \end{aligned}$$

Example

To compute the daily mean over all input years use:

```
cdo ydaymean infile outfile
```

Author

Uwe Schulzweida

2.8.35 Ydayctl

Name

ydayctl - Multi-year daily percentile

Synopsis

cdo ydayctl,*pn infile1 infile2 infile3 outfile*

Description

This operator writes a certain percentile of each day of year in **infile1** to **outfile**. The algorithm uses histograms with minimum and maximum bounds given in **infile2** and **infile3**, 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 **infile2** and **infile3** should be the result of corresponding *ydaymin* and *ydaymax* operations, respectively. The date information in an output field is the date of the last contributing input field.

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

`CDO_PCTL_NBINS` sets the number of histogram bins (default: `CDO_PCTL_NBINS=101`).

Example

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

```
cdo ydaymin infile minfile
cdo ydaymax infile maxfile
cdo ydayctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo ydayctl,90 infile -ydaymin infile -ydaymax infile outfile
```

Author

Uwe Schulzweida

2.8.36 Ymonstat

Name

ymin, ymonmax, ymonrange, ymonsum, ymonmean, ymonavg, ymonstd, ymonstd1, ymonvar, ymonvar1 - Multi-year monthly statistics

Synopsis

cdo <operator> *infile outfile*

Description

This module computes statistical values of each month of year. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of each month of year in *infile* is written to *outfile*. The date information in an output field is the date of the last contributing input field. This can be changed with the **CDO** option `--timestat_date <first|middle|last>`.

Operators

ymin

Multi-year monthly minimum

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

ymonmax

Multi-year monthly maximum

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

ymonrange

Multi-year monthly range

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

ymonsum

Multi-year monthly sum

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

ymonmean

Multi-year monthly mean

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

ymonavg

Multi-year monthly average

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

ymonvar

Multi-year monthly variance

Normalize by n.

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

ymonvar1

Multi-year monthly variance (n-1)

Normalize by (n-1).

$$\begin{aligned}o(01, x) &= \mathbf{var1}\{i(t, x), \text{month}(i(t)) = 01\} \\ &\vdots \\ o(12, x) &= \mathbf{var1}\{i(t, x), \text{month}(i(t)) = 12\}\end{aligned}$$

ymonstd

Multi-year monthly standard deviation

Normalize by n.

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

ymonstd1

Multi-year monthly standard deviation (n-1)

Normalize by (n-1).

$$\begin{aligned}o(01, x) &= \mathbf{std1}\{i(t, x), \text{month}(i(t)) = 01\} \\ &\vdots \\ o(12, x) &= \mathbf{std1}\{i(t, x), \text{month}(i(t)) = 12\}\end{aligned}$$

Example

To compute the monthly mean over all input years use:

`cdo ymonmean infile outfile`

Author

Uwe Schulzweida

2.8.37 Ymonpctl

Name

ymonpctl - Multi-year monthly percentile

Synopsis

cdo ymonpctl,*pn* *infile1* *infile2* *infile3* *outfile*

Description

This operator writes a certain percentile of each month of year in *infile1* to *outfile*. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, 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 *infile2* and *infile3* should be the result of corresponding *ymonmin* and *ymonmax* operations, respectively. The date information in an output field is the date of the last contributing input field.

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

`CDO_PCTL_NBINS` sets the number of histogram bins (default: `CDO_PCTL_NBINS=101`).

Example

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

```
cdo ymonmin infile minfile
cdo ymonmax infile maxfile
cdo ymonpctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo ymonpctl,90 infile -ymonmin infile -ymonmax infile outfile
```

Author

Uwe Schulzweida

2.8.38 Yseasstat

Name

yseasmin, yseasmax, yseasrange, yseassum, yseasmean, yseasavg, yseasstd, yseasstd1, yseasvar, yseasvar1 - Multi-year seasonal statistics

Synopsis

cdo <operator> *infile outfile*

Description

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

Operators

yseasmin

Multi-year seasonal minimum

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

yseasmax

Multi-year seasonal maximum

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

yseasrange

Multi-year seasonal range

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

yseassum

Multi-year seasonal sum

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

yseasmean

Multi-year seasonal mean

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

yseasavg

Multi-year seasonal average

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

yseasvar

Multi-year seasonal variance

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

yseasvar1

Multi-year seasonal variance (n-1)

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

yseasstd

Multi-year seasonal standard deviation

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

yseasstd1

Multi-year seasonal standard deviation (n-1)

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

Example

To compute the seasonal mean over all input years use:

```
cdo yseasmean infile outfile
```

Author

Uwe Schulzweida

2.8.39 Yseaspctl

Name

yseaspctl - Multi-year seasonal percentile

Synopsis

cdo yseaspctl,*pn* *infile1* *infile2* *infile3* *outfile*

Description

This operator writes a certain percentile of each season in *infile1* to *outfile*. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, 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 *infile2* and *infile3* should be the result of corresponding *yseasmin* and *yseasmax* operations, respectively. The date information in an output field is the date of the last contributing input field.

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

Environment

`CDO_PCTL_NBINS` sets the number of histogram bins (default: `CDO_PCTL_NBINS=101`).

Example

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

```
cdo yseasmin infile minfile
cdo yseasmax infile maxfile
cdo yseaspctl,90 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo yseaspctl,90 infile -yseasmin infile -yseasmax infile outfile
```

Author

Uwe Schulzweida

2.8.40 Ydrunstat

Name

ydrunmin, ydrunmax, ydrunsum, ydrunmean, ydrunavg, ydrunstd, ydrunstd1, ydrunvar, ydrunvar1 - Multi-year daily running statistics

Synopsis

cdo <operator>,nts[,rm=c] *infile outfile*

Description

This module writes running statistical values for each day of year in *infile* to *outfile*. Depending on the chosen operator, the minimum, maximum, sum, average, variance or standard deviation of all timesteps in running windows of which the middle timestep corresponds to a certain day of year is computed. The date information in an output field is the date of the timestep in the middle of the last contributing running window. Note that the operator has 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

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

ydrunmax

Multi-year daily running maximum

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

ydrunsum

Multi-year daily running sum

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

ydrunmean

Multi-year daily running mean

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

ydrunavg

Multi-year daily running average

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

ydrunvar

Multi-year daily running variance

Normalize by n.

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

ydrunvar1

Multi-year daily running variance (n-1)

Normalize by (n-1).

$$\begin{aligned}
 o(001, x) &= \text{var1}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\} \\
 &\vdots \\
 o(366, x) &= \text{var1}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}
 \end{aligned}$$

ydrunstd

Multi-year daily running standard deviation

Normalize by n.

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

ydrunstd1

Multi-year daily running standard deviation (n-1)

Normalize by (n-1).

$$\begin{aligned}
 o(001, x) &= \text{std1}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\} \\
 &\vdots \\
 o(366, x) &= \text{std1}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}
 \end{aligned}$$

Parameters**nts**

[INTEGER] Number of timesteps

rm=c

[STRING] Read method circular

Example

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

```
cdo ydrunmean,5 infile outfile
```

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

```
cdo ydaymean -runmean,5 infile outfile
```

Author

Ralf Quast, Uwe Schulzweida, Fabian Wachsman

2.8.41 Ydrunpctl

Name

ydrunpctl - Multi-year daily running percentile

Synopsis

cdo ydrunpctl,*pn,nts*[,*rm=c*][,*pm=r8*] *infile1 infile2 infile3 outfile*

Description

This operator writes running percentile values for each day of year in *infile1* to *outfile*. A certain percentile is computed for all timesteps in running windows of which the middle timestep corresponds to a certain day of year. The algorithm uses histograms with minimum and maximum bounds given in *infile2* and *infile3*, 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 *infile2* and *infile3* should be the result of corresponding *ydrunmin* and *ydrunmax* operations, respectively. The date information in an output field is the date of the timestep in the middle of the last contributing running window. Note that the operator has to be applied to a continuous time series of daily measurements in order to yield physically meaningful results. Also note that the output time series begins $(nts-1)/2$ timesteps after the first timestep of the input time series and ends $(nts-1)/2$ timesteps before the last. For input data which are complete but not continuous, such as time series of daily measurements for the same month or season within different years, the operator only yields physically meaningful results if the input time series does include the $(nts-1)/2$ days before and after each period of interest.

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

Parameters

pn

[FLOAT] Percentile number in {0, ..., 100}

nts

[INTEGER] Number of timesteps

rm=c

[STRING] Read method circular

pm=r8

[STRING] Percentile method rtype8

Environment

CDO_PCTL_NBINS sets the number of histogram bins (default: *CDO_PCTL_NBINS*=101).

Example

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

```
cdo ydrunmin,5 infile minfile
cdo ydrunmax,5 infile maxfile
cdo ydrunpctl,90,5 infile minfile maxfile outfile
```

Or shorter using operator piping:

```
cdo ydrunpctl,90,5 infile -ydrunmin infile -ydrunmax infile outfile
```

Author

Ralf Quast, Uwe Schulzweida, Fabian Wachsmann

2.9 Correlation

This section contains modules for correlation and covariance in grid space and over time.

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

$$\begin{array}{l} \text{Covariance} \\ \mathbf{covar} \end{array} \quad n^{-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$
$$\mathbf{covar} \text{ weighted by } \{w_i, i = 1, \dots, n\} \quad \left(\sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left(x_i - \left(\sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right) \left(y_i - \left(\sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j y_j \right)$$

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

<i>Fldcor</i>	<i>fldcor</i>	Correlation in grid space
<i>Timcor</i>	<i>timcor</i>	Correlation over time
<i>Fldcovar</i>	<i>fldcovar</i>	Covariance in grid space
<i>Timcovar</i>	<i>timcovar</i>	Covariance over time

2.9.1 Fldcor

Name

fldcor - Correlation in grid space

Synopsis

cdo fldcor *infile1 infile2 outfile*

Description

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

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

it is

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

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

Author

Uwe Schulzweida

2.9.2 Timcor

Name

timcor - Correlation over time

Synopsis

cdo timcor *infile1 infile2 outfile*

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. If there is only one input field, the p-value (probability value) is also written out. With

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

it is

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

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

Author

Uwe Schulzweida

2.9.3 fldcovar

Name

fldcovar - Covariance in grid space

Synopsis

cdo fldcovar *infile1 infile2 outfile*

Description

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

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

it is

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

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

Author

Uwe Schulzweida

2.9.4 Timcovar

Name

timcovar - Covariance over time

Synopsis

cdo timcovar *infile1 infile2 outfile*

Description

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

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

it is

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

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

Author

Uwe Schulzweida

2.10 Regression

This section contains modules for linear regression of time series.

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

<i>Regres</i>	<i>regres</i>	Regression
<i>Detrend</i>	<i>detrend</i>	Detrend time series
<i>Trend</i>	<i>trend</i>	Trend of time series
<i>Trendarith</i>	<i>addtrend</i>	Add trend
	<i>subtrend</i>	Subtract trend

2.10.1 Regres

Name

regres - Regression

Synopsis

cdo regres[,*equal*] *infile outfile*

Description

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

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

It is assumed that all timesteps are equidistant, if this is not the case set the parameter `equal=false`.

Parameters

equal

[BOOL] Set to false for unequally distributed timesteps (default: true)

Author

Uwe Schulzweida

2.10.2 Detrend

Name

detrend - Detrend time series

Synopsis

cdo [options] detrend[,equal] *infile outfile*

Description

Every time series in *infile* is linearly detrended. For every field element x only those timesteps t belong to the sample S , which have $i(t, x) \neq \text{miss}$. It is assumed that all timesteps are equidistant, if this is not the case set the parameter `equal=false`. With

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

and

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

it is

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

Parameters

equal

[BOOL] Set to false for unequally distributed timesteps (default: true)

Options

`-p`, `--async_read true` to read input data asynchronously.

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 *infile* and to store the detrended data in *outfile* use:

```
cdo detrend infile outfile
```

Author

Uwe Schulzweida

2.10.3 Trend

Name

trend - Trend of time series

Synopsis

cdo [options] trend[,equal] *infile outfile1 outfile2*

Description

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

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

and

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

Thus the estimation for a is stored in *outfile1* and that for b is stored in *outfile2*. To subtract the trend from the data see operator *subtrend*. It is assumed that all timesteps are equidistant, if this is not the case set the parameter *equal=false*.

Parameters

equal

[BOOL] Set to false for unequally distributed timesteps (default: true)

Options

-p, *--async_read true* to read input data asynchronously.

Author

Uwe Schulzweida

2.10.4 Trendarith

Name

addtrend, subtrend - Add or subtract a trend

Synopsis

cdo <operator>[,equal] infile1 infile2 infile3 outfile

Description

This module is for adding or subtracting a trend computed by the operator *trend*.

Operators

addtrend

Add 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.

subtrend

Subtract 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.

Parameters

equal

[BOOL] Set to false for unequally distributed timesteps (default: true)

Example

The typical call for detrending the data in **infile** and storing the detrended data in **outfile** is:

```
cdo trend infile afile bfile
cdo subtrend infile afile bfile outfile
```

The result is identical to a call of the operator *detrend*:

```
cdo detrend infile outfile
```

Author

Uwe Schulzweida

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 in Meteorology and Oceanography [[Preisendorfer](#)]

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

Statistical Analysis in Climate Research [[vonStorch](#)]

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**

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

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

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

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

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

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

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

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

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

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

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

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

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

<i>EOF</i>	<i>eof</i>	Calculate EOFs in spatial or time space
	<i>eoftime</i>	Calculate EOFs in time space
	<i>eofspatial</i>	Calculate EOFs in spatial space
	<i>eof3d</i>	Calculate 3-Dimensional EOFs in time space
<i>Eofcoeff</i>	<i>eofcoeff</i>	Principal coefficients of EOFs

2.11.1 EOF

Name

eof, eoftime, eofspatial, eof3d - Empirical Orthogonal Functions

Synopsis

cdo <operator>,neof infile outfile1 outfile2

Description

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

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

If operator *eof* is chosen, the EOFs are computed in either time or spatial space, whichever is the fastest. If the user already knows, which computation is faster, the module can be forced to perform a computation in time- or gridspace by using the operators *eoftime* or *eofspatial*, respectively. This can enhance performance, especially for very long time series, where the number of timesteps is larger than the number of grid-points. Data in **infile** are assumed to be anomalies. If they are not, the behavior of this module is **not well defined**. After execution **outfile1** will contain all eigen-values and **outfile2** the eigenvectors e_j . All EOFs and eigen-values are computed. However, only the first *neof* EOFs are written to **outfile2**. Nonetheless, **outfile1** contains all eigen-values.

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

Operators

- eof**
Calculate EOFs in spatial or time space
- eof3d**
Calculate 3-Dimensional EOFs in time space
- eoftime**
Calculate EOFs in time space
- eofspatial**
Calculate EOFs in spatial space

Parameters

- neof**
[INTEGER] Number of eigen functions

Environment

CDO_SVD_MODE

Is used to choose the algorithm for eigenvalue calculation. Options are 'jacobi' for a one-sided parallel jacobi-algorithm (only executed in parallel if -P flag is set) and 'danielson_lanczos' for a non-parallel d/l algorithm. The default setting is 'jacobi'.

CDO_WEIGHT_MODE

It is used to set the weight mode. The default is 'off'. Set it to 'on' for a weighted version.

MAX_JACOBI_ITER

Is the maximum integer number of annihilation sweeps that is executed if the jacobi-algorithm is used to compute the eigen values. The default value is 12.

FNORM_PRECISION

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

Example

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

```
cdo eof,40 infile outfile1 outfile2
```

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

```
cdo sub infile1 -timmean infile1 anom_file  
cdo eof,40 anom_file outfile1 outfile2
```

Author

Cedrick Anorge

2.11.2 Eofcoeff

Name

eofcoeff - Principal coefficients of EOFs

Synopsis

cdo eofcoeff *infile1 infile2 obase*

Description

This module calculates the time series of the principal coefficients for given EOF (empirical orthogonal functions) and data. Time steps in *infile1* are assumed to be the EOFs, time steps in *infile2* are assumed to be the time series.

Note that this operator calculates a non weighted dot product of the fields in *infile1* and *infile2*. For consistency set the environment variable `CDO_WEIGHT_MODE=off` when using *eof* or *eof3d*.

Given a set of EOFs e_j and a time series of data $z(t)$ with p entries for each timestep from which e_j have been calculated, this operator calculates the time series of the projections of data onto each EOF

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

There will be a separate file o_j for the principal coefficients of each EOF.

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

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

There will be a separate file containing a time series of principal coefficients with time information from *infile2* for each EOF in *infile1*. Output files will be numbered as `<obase><neof><suffix>` where `neof+1` is the number of the EOF (timestep) in *infile1* and `suffix` is the filename extension derived from the file format.

Environment

`CDO_FILE_SUFFIX` sets the filename suffix.

Example

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

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

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

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

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

Author

Cedrick Ansoerge

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:

<i>Remapbil</i>	<i>remapbil</i>	Bilinear interpolation
	<i>genbil</i>	Generate bilinear interpolation weights
<i>Remapbic</i>	<i>remapbic</i>	Bicubic interpolation
	<i>genbic</i>	Generate bicubic interpolation weights
<i>Remapknn</i>	<i>remapknn</i>	k-nearest neighbor remapping
	<i>remapnn</i>	Nearest neighbor remapping
	<i>remapdis</i>	Distance weighted average remapping
	<i>genknn</i>	Generate k-nearest neighbor remap weights
	<i>gennn</i>	Generate nearest neighbor remap weights
	<i>gendis</i>	Generate distance weighted average remap weights
<i>Remapcon</i>	<i>remapcon</i>	First order conservative remapping
	<i>gencon</i>	Generate 1st order conservative remap weights
<i>Remaplaf</i>	<i>remaplaf</i>	Largest area fraction remapping
	<i>genlaf</i>	Generate largest area fraction remap weights
<i>Remap</i>	<i>remap</i>	Grid remapping
<i>Remapeta</i>	<i>remapeta</i>	Remap vertical hybrid levels
<i>Vertintml</i>	<i>ml2pl</i>	Model to pressure level interpolation
<i>Vertintap</i>	<i>ap2pl</i>	Vertical pressure interpolation
<i>Vertintgh</i>	<i>gh2hl</i>	Geometric height interpolation
<i>Intlevel</i>	<i>intlevel</i>	Linear level interpolation
<i>Intlevel3d</i>	<i>intlevel3d</i>	Linear level interpolation from/to 3D vertical coordinates
<i>Inttime</i>	<i>inttime</i>	Interpolation between timesteps
	<i>intntime</i>	Interpolation between timesteps
<i>Intyear</i>	<i>intyear</i>	Interpolation between two years

2.12.1 Remapbil

Name

remapbil, genbil - Bilinear interpolation

Synopsis

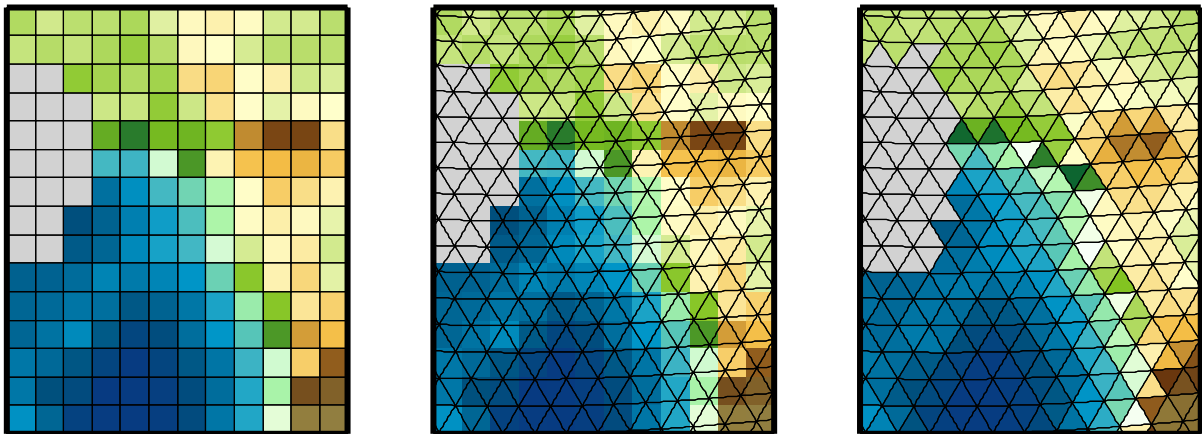
cdo remapbil,*targetgrid* *infile* *outfile*

cdo genbil,*targetgrid*[,*map3d*] *infile* *outfile*

Description

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

Below is a schematic illustration of the bilinear remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapbil

Bilinear interpolation

Performs a bilinear interpolation on all input fields.

genbil

Generate bilinear interpolation weights

Generates bilinear interpolation weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator *remap* to apply this remapping weights to a data file with the same source grid. Set the parameter **map3d=true** to generate all mapfiles of the first 3D field with varying masks. In this case the mapfiles will be named <outfile><xxx>.nc. xxx will have five digits with the number of the mapfile.

Parameters

targetgrid

[STRING] Target grid description file or name

map3d

[BOOL] Generate all mapfiles of the first 3D field

Environment

REMAP_EXTRAPOLATE is used to switch the extrapolation feature 'on' or 'off'. By default the extrapolation is enabled for cyclic grids.

Example

Say *infile* contains fields on a quadrilateral curvilinear grid. To remap all fields bilinear to a regular Gaussian F32 grid, type:

```
cdo remapbil,F32 infile outfile
```

Author

Uwe Schulzweida

2.12.2 Remapbic

Name

remapbic, genbic - Bicubic interpolation

Synopsis

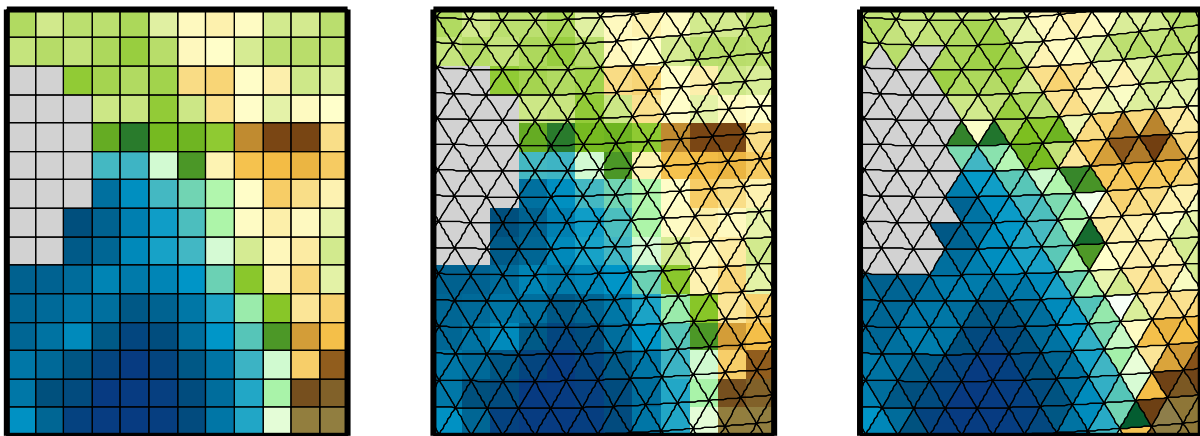
cdo remapbic,targetgrid infile outfile

cdo genbic,targetgrid[,map3d] infile outfile

Description

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

Below is a schematic illustration of the bicubic remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapbic

Bicubic interpolation

Performs a bicubic interpolation on all input fields.

genbic

Generate bicubic interpolation weights

Generates bicubic interpolation weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator *remap* to apply this remapping weights to a data file with the same source grid. Set the parameter **map3d=true** to generate all mapfiles of the first 3D field with varying masks. In this case the mapfiles will be named <outfile><xxx>.nc. xxx will have five digits with the number of the mapfile.

Parameters

targetgrid

[STRING] Target grid description file or name

map3d

[BOOL] Generate all mapfiles of the first 3D field

Environment

REMAP_EXTRAPOLATE is used to switch the extrapolation feature 'on' or 'off'. By default the extrapolation is enabled for cyclic grids.

Example

Say *infile* contains fields on a quadrilateral curvilinear grid. To remap all fields bicubic to a regular Gaussian F32 grid, type:

```
cdo remapbic,F32 infile outfile
```

Author

Uwe Schulzweida

2.12.3 Remapknn

Name

remapknn, remapnn, remapdis, genknn, gennn, gendis - k -nearest neighbor remapping

Synopsis

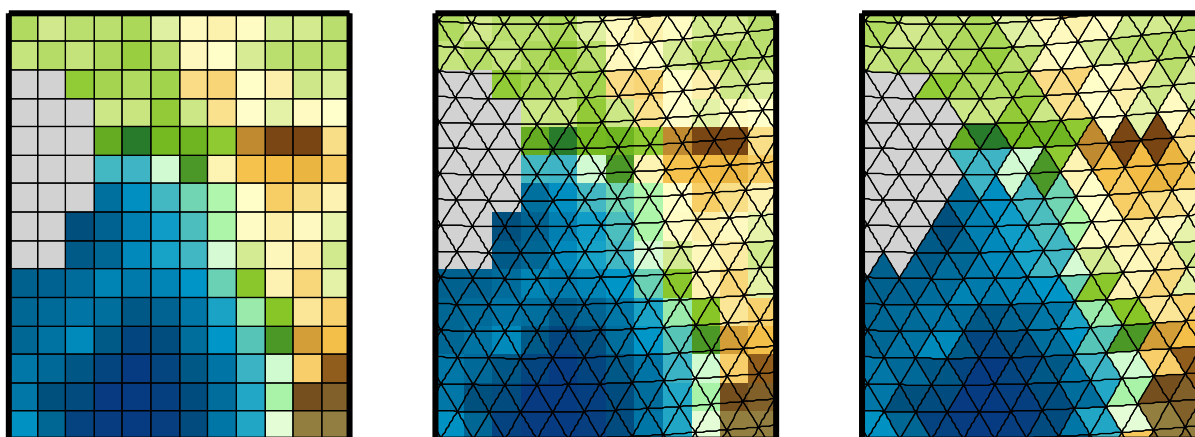
cdo <operator>.parameter infile outfile

Description

This module contains operators for a k -nearest neighbor remapping of fields between grids in spherical coordinates. The default number for k is 1. If k is greater than 1, the *weighted* parameter can be used to choose between different weighting methods. The default method is *weighted=dist*, for an inverse distance weighting. Here is a list of all available weighting methods.

dist	Inverse distance weighted
avg	Simple arithmetic average
gauss	Gaussian filter

Below is a schematic illustration of the nearest neighbor remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapknn

k -nearest neighbor remapping

Performs a k -nearest neighbor remapping on all input fields.

remapnn

Nearest neighbor remapping

remapnn,<targetgrid> corresponds to **remapknn**,grid=<targetgrid>,extrapolate=true

remapdis

Distance weighted average remapping

remapdis,<targetgrid> corresponds to **remapknn**,grid=<targetgrid>,k=4,kmin=1,weighted=dist,extrapolate=true

genknn

Generate k -nearest neighbor remap weights

Generates k -nearest neighbor remapping weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator *remap* to apply this remapping weights to a data file with the same source grid. Set the parameter **map3d=true** to generate all mapfiles of the first 3D field with varying masks. In this case the mapfiles will be named `<outfile><xxx>.nc`. `xxx` will have five digits with the number of the mapfile.

gennn

Generate nearest neighbor remap weights

gennn,<targetgrid> corresponds to **genknn**,grid=<targetgrid>,extrapolate=true

gendis

Generate distance weighted average remap weights

gendis,<targetgrid> corresponds to **genknn**,grid=<targetgrid>,k=4,kmin=1,weighted=dist,extrapolate=true

Parameters**grid**

[STRING] Target grid description file or name

k

[INTEGER] Number of nearest neighbors [default: 1]

kmin

[INTEGER] Minimum number of nearest neighbors [default: k]

weighted

[WORD] Weighting method (dist/avg/gauss) [default: dist]

gauss_scale

[FLOAT] Scaling factor for gauss weighting method [default: 0.1]

extrapolate

[BOOL] Extrapolate [default: false]

map3d

[BOOL] Generate all mapfiles of the first 3D field [default: false]

Environment

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

CDO_GRIDSEARCH_RADIUS sets the grid search radius in degree (default: *CDO_GRIDSEARCH_RADIUS*=180).

See Also

Remapbil, Remapbic, Remapcon

Author

Uwe Schulzweida

2.12.4 Remapcon

Name

remapcon, gencon - First order conservative remapping

Synopsis

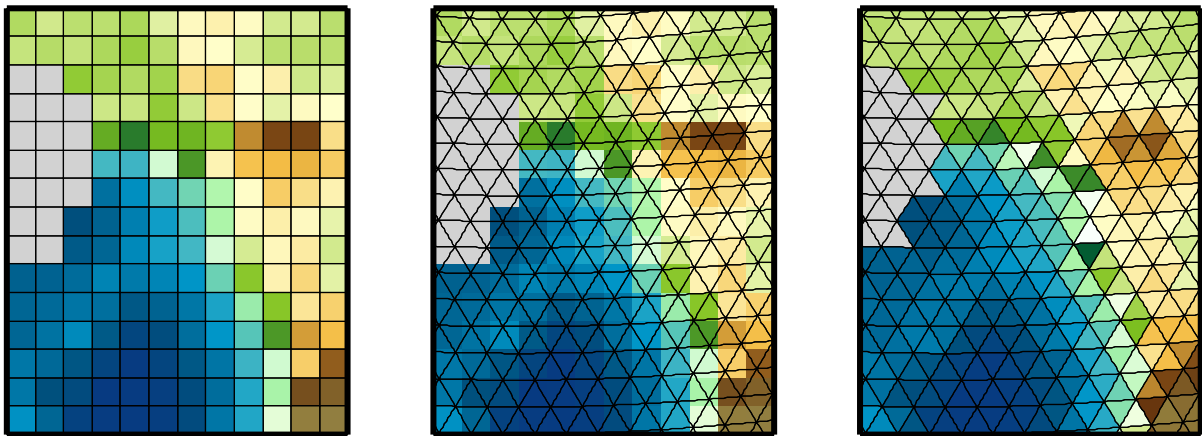
cdo remapcon,*targetgrid* *infile* *outfile*

cdo gencon,*targetgrid*[,*map3d*] *infile* *outfile*

Description

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

Below is a schematic illustration of the 1st order conservative remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remapcon

First order conservative remapping

Performs a first order conservative remapping on all input fields.

gencon

Generate 1st order conservative remap weights

Generates first order conservative remapping weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator *remap* to apply this remapping weights to a data file with the same source grid. Set the parameter **map3d=true** to generate all mapfiles of the first 3D field with varying masks. In this case the mapfiles will be named *<outfile><xxx>.nc*. *xxx* will have five digits with the number of the mapfile.

Parameters

targetgrid

[STRING] Target grid description file or name

map3d

[BOOL] Generate all mapfiles of the first 3D field

Environment

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

REMAP_AREA_MIN is used to set the minimum destination area fraction (default: REMAP_AREA_MIN=0.0).

Example

Say *infile* contains fields on a quadrilateral curvilinear grid. To remap all fields conservative to a regular Gaussian F32 grid, type:

```
cdo remapcon,F32 infile outfile
```

Author

Uwe Schulzweida

2.12.5 Remaplaf

Name

remaplaf, genlaf - Largest area fraction remapping

Synopsis

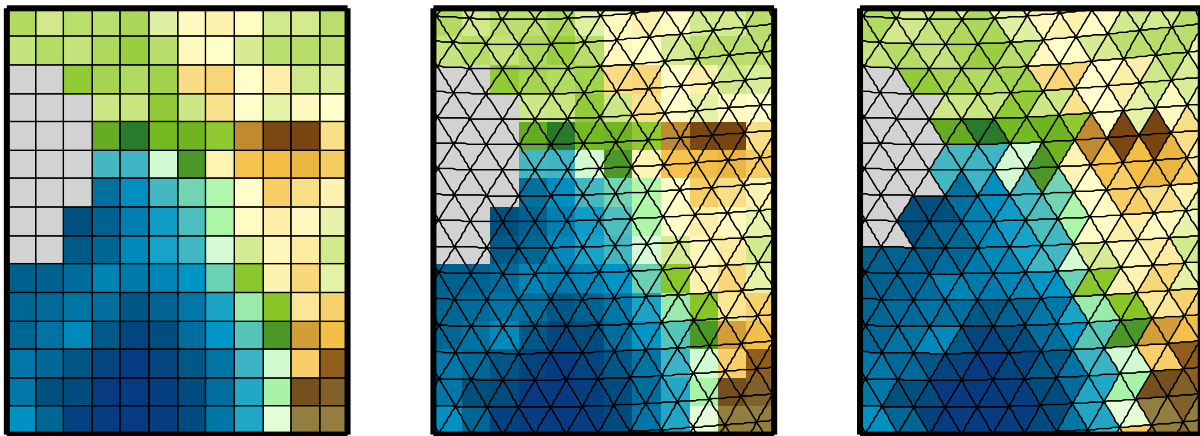
cdo remaplaf,*targetgrid* *infile* *outfile*

cdo genlaf,*targetgrid*[,*map3d*] *infile* *outfile*

Description

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

Below is a schematic illustration of the largest area fraction conservative remapping:



The figure on the left side shows the input data on a regular lon/lat source grid and on the right side the remapped result on an unstructured triangular target grid. The figure in the middle shows the input data with the target grid. Grid cells with missing value are grey colored.

Operators

remaplaf

Largest area fraction remapping

Performs a largest area fraction remapping on all input fields.

genlaf

Generate largest area fraction remap weights

Generates largest area fraction remapping weights for the first input field and writes the result to a file. The format of this file is NetCDF following the SCRIP convention. Use the operator *remap* to apply this remapping weights to a data file with the same source grid.

Parameters

targetgrid

[STRING] Target grid description file or name

map3d

[BOOL] Generate all mapfiles of the first 3D field

Environment

REMAP_AREA_MIN is used to set the minimum destination area fraction (default: REMAP_AREA_MIN=0.0).

Author

Uwe Schulzweida

2.12.6 Remap

Name

remap - Grid remapping

Synopsis

cdo remap,*targetgrid*,*weights* *infile* *outfile*

Description

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

Parameters

targetgrid

[STRING] Target grid description file or name

weights

[STRING] Interpolation weights (SCRIP NetCDF file)

Environment

CDO_GRIDSEARCH_RADIUS sets the grid search radius in degree (default: *CDO_GRIDSEARCH_RADIUS*=180).

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

REMAP_AREA_MIN is used to set the minimum destination area fraction (default: *REMAP_AREA_MIN*=0.0).

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

Example

Say *infile* contains fields on a quadrilateral curvilinear grid. To remap all fields bilinear to a regular Gaussian F32 grid use:

```
cdo genbil,F32 infile remapweights.nc
cdo remap,F32,remapweights.nc infile outfile
```

The result will be the same as::

```
cdo remapbil,F32 infile outfile
```

Author

Uwe Schulzweida

2.12.7 Remapeta

Name

remapeta - Remap vertical hybrid levels

Synopsis

```
cdo remapeta,vct[,oro] infile outfile
```

Description

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

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

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

Parameters

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.

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 *Z-axis description* for an example how to define a hybrid Z-axis.

Example

To remap between different hybrid model level data use:

```
cdo remapeta,vct infile outfile
```

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

0	0.000000000000000000	0.000000000000000000
1	2000.0000000000000000	0.000000000000000000
2	4000.0000000000000000	0.000000000000000000
3	6046.1093750000000000	0.00033899326808751
4	8267.9296875000000000	0.00335718691349030
5	10609.5117187500000000	0.01307003945112228
6	12851.1015625000000000	0.03407714888453484
7	14698.5000000000000000	0.07064980268478394
8	15861.1289062500000000	0.12591671943664551
9	16116.2382812500000000	0.20119541883468628
10	15356.9218750000000000	0.29551959037780762
11	13621.4609375000000000	0.40540921688079834
12	11101.5585937500000000	0.52493220567703247
13	8127.1445312500000000	0.64610791206359863
14	5125.1406250000000000	0.75969839096069336
15	2549.9689941406250000	0.85643762350082397
16	783.1950683593750000	0.92874687910079956
17	0.000000000000000000	0.97298520803451538
18	0.000000000000000000	0.99228149652481079
19	0.000000000000000000	1.000000000000000000

Author

Uwe Schulzweida, Ingo Kirchner

2.12.8 Vertintml

Name

ml2pl - Model to pressure level interpolation

Synopsis

cdo ml2pl,plevels *infile outfile*

Description

Interpolates 3D variables on hybrid sigma pressure level to pressure levels. A basic linear method is used for interpolation. To calculate the pressure on model levels, the a and b coefficients defining the model levels and the surface pressure are required. The a and b coefficients are normally part of the model level data. If not available, the surface pressure can be derived from the logarithm of the surface pressure. To extrapolate the temperature, the surface geopotential is also needed. The geopotential height must be present at the hybrid layer interfaces (model half-layers)! All needed variables are identified by their GRIB1 code number or NetCDF CF standard name. Supported parameter tables are: WMO standard table number 2 and ECMWF local table number 128.

Name	Units	GRIB1 code	CF standard name
log surface pressure	Pa	152	
surface pressure	Pa	134	surface_air_pressure
air temperature	K	130	air_temperature
surface geopotential	m2 s-2	129	surface_geopotential
geopotential height	m	156	geopotential_height

Use the alias **ml2plx** to fill in missing values with the next available value of the same vertical column. Only the temperature is extrapolated in this case. The extrapolation method originates from the ECHAM postprocessing. This operator requires all variables on the same horizontal grid. Missing values in the input data are not supported.

Parameters

plevels

[FLOAT] Pressure levels in pascal

Note

This is a specific implementation for data from the ECHAM model, it may not work with data from other sources. The components of the hybrid coordinate must always be available at the hybrid layer interfaces even if the data is defined at the hybrid layer midpoints.

Example

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

```
cdo ml2pl,92500,85000,50000,20000 infile outfile
```

Author

Uwe Schulzweida

2.12.9 Vertintap

Name

ap2pl - Vertical pressure interpolation

Synopsis

```
cdo ap2pl,levels infile outfile
```

Description

Interpolate 3D variables on hybrid sigma height coordinates to pressure levels. A basic linear method is used for interpolation. The input file must contain the 3D air pressure in pascal. The air pressure is identified by the NetCDF CF standard name *air_pressure*. Use the alias **ap2plx** to fill in missing values with the next available value of the same vertical column. This operator requires all variables on the same horizontal grid. Missing values in the input data are not supported.

Parameters

levels

[FLOAT] Comma-separated list of pressure levels in pascal

Note

This is a specific implementation for NetCDF files from the ICON model, it may not work with data from other sources.

Example

To interpolate 3D variables on hybrid sigma height level to pressure levels of 925, 850, 500 and 200 hPa use:

```
cdo ap2pl,92500,85000,50000,20000 infile outfile
```

Author

Uwe Schulzweida

2.12.10 Vertintgh

Name

gh2hl - Geometric height interpolation

Synopsis

cdo gh2hl,*levels infile outfile*

Description

Interpolate 3D variables on hybrid sigma height coordinates to height levels. A basic linear method is used for interpolation. The input file must contain the 3D geometric height in meter. The geometric height is identified by the NetCDF CF standard name *geometric_height_at_full_level_center*. Use the alias **gh2hlx** to fill in missing values with the next available value of the same vertical column. This operator requires all variables on the same horizontal grid. Missing values in the input data are not supported.

Parameters

levels

[FLOAT] Comma-separated list of height levels in meter

Note

This is a specific implementation for NetCDF files from the ICON model, it may not work with data from other sources.

Example

To interpolate 3D variables on hybrid sigma height level to height levels of 20, 100, 500, 1000, 5000, 10000 and 20000 meter use:

```
cdo gh2hl,20,100,500,1000,5000,10000,20000 infile outfile
```

Author

Uwe Schulzweida

2.12.11 Intlevel

Name

intlevel - Linear level interpolation

Synopsis

cdo intlevel,parameters *infile outfile*

Description

This operator performs a linear vertical interpolation of 3D variables. The 1D target levels can be specified with the level parameter or read in via a Z-axis description file.

Parameters

level

[FLOAT] Comma-separated list of target levels

zdescription

[STRING] Path to a file containing a description of the Z-axis

zvarname

[STRING] Use zvarname as the vertical 3D source coordinate instead of the 1D coordinate variable

extrapolate

[BOOL] Fill target layers out of the source layer range with the nearest source layer

Example

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

```
cdo intlevel,level=10,50,100,500,1000 infile outfile
```

Author

Uwe Schulzweida

2.12.12 Intlevel3d

Name

intlevel3d - Linear level interpolation from/to 3D vertical coordinates

Synopsis

```
cdo intlevel3d,tgtcoordinate infile1 infile2 outfile
```

Description

This operator performs a linear vertical interpolation of 3D variables fields with given 3D vertical coordinates. **infile1** contains the 3D data variables and **infile2** the 3D vertical source coordinate. The parameter **tgtcoordinate** is a datafile with the 3D vertical target coordinate. Use the alias **intlevel3dx** to fill in missing values with the next available value of the same vertical column.

Parameters

tgtcoordinate

[STRING] filename for 3D vertical target coordinates

Example

To interpolate 3D variables from one set of 3D height levels into another one where

- **infile2** contains a single 3D variable, which represents the source 3D vertical coordinate
- **infile1** contains the source data, which the vertical coordinate from **infile2** belongs to
- **tgtcoordinate** only contains the target 3D height levels

```
cdo intlevel3d,tgtcoordinate infile1 infile2 outfile
```

Author

Ralf Müller

2.12.13 Inttime

Name

inttime, intntime - Time interpolation

Synopsis

cdo inttime,date,time[,inc] infile outfile

cdo intntime,n infile outfile

Description

This module performs linear interpolation between timesteps. Interpolation is only performed if both values exist. If both values are missing values, the result is also a missing value. If only one value exists, it is taken if the time weighting is greater than or equal to 0.5. So no new value will be created at existing time steps, if the value is missing there.

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.

Parameters

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

Assume 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 infile outfile
```

Author

Uwe Schulzweida

2.12.14 Intyear

Name

intyear - Interpolation between two years

Synopsis

```
cdo intyear,years infile1 infile2 obase
```

Description

This operator performs linear interpolation between two years, timestep by timestep. The input files need to have the same structure with the same variables. The output files will be named `<obase><yyyy><suffix>` where `yyyy` will be the year and `suffix` is the filename extension derived from the file format.

Parameters

years

[INTEGER] Comma-separated list or first/last[/inc] range of years

Environment

`CDO_FILE_SUFFIX` sets the filename suffix.

Note

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

Example

Assume there are two monthly mean datasets over a year. The first dataset has 12 timesteps for the year 1985 and the second one for the year 1990. To interpolate the years between 1985 and 1990 month by month use:

```
cdo intyear,1986,1987,1988,1989 infile1 infile2 year
```

Example result of `'dir year*'` for NetCDF datasets:

```
year1986.nc year1987.nc year1988.nc year1989.nc
```

Author

Uwe Schulzweida

2.13 Transformation

This section contains modules to perform spectral transformations.

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

<i>Spectral</i>	<i>sp2gp</i>	Spectral to gridpoint
	<i>gp2sp</i>	Gridpoint to spectral
<i>Specconv</i>	<i>sp2sp</i>	Spectral to spectral
<i>Wind2</i>	<i>dv2ps</i>	D and V to vel. potential and stream function
<i>Wind</i>	<i>dv2uv</i>	Divergence and vorticity to U and V wind
	<i>uv2dv</i>	U and V wind to divergence and vorticity
<i>Fourier</i>	<i>fourier</i>	Fourier transformation

2.13.1 Spectral

Name

sp2gp, gp2sp - Spectral transformation

Synopsis

cdo <operator>[,parameters] *infile outfile*

Description

This module transforms fields on a global regular Gaussian grid to spectral coefficients and vice versa. The transformation is achieved by applying Fast Fourier Transformation (FFT) first and direct Legendre Transformation afterwards in gp2sp. In sp2gp the inverse Legendre Transformation and inverse FFT are used. Missing values are not supported.

The relationship between the spectral resolution, governed by the truncation number T , and the grid resolution depends on the number of grid points at which the shortest wavelength field is represented. For a grid with $2N$ points between the poles (so $4N$ grid points in total around the globe) the relationship is:

linear grid: the shortest wavelength is represented by 2 grid points $\rightarrow 4N \simeq 2(TL + 1)$

quadratic grid: the shortest wavelength is represented by 3 grid points $\rightarrow 4N \simeq 3(TQ + 1)$

cubic grid: the shortest wavelength is represented by 4 grid points $\rightarrow 4N \simeq 4(TC + 1)$

The quadratic grid is used by ECHAM and ERA15. ERA40 is using a linear Gaussian grid reflected by the TL notation.

The following table shows the calculation of the number of latitudes and the triangular truncation for the different grid types:

Gridtype	Number of latitudes: nlat	Triangular truncation: ntr
linear	$NINT((ntr*2 + 1)/2)$	$(nlat*2 - 1) / 2$
quadratic	$NINT((ntr*3 + 1)/2)$	$(nlat*2 - 1) / 3$
cubic	$NINT((ntr*4 + 1)/2)$	$(nlat*2 - 1) / 4$

Operators

sp2gp

Spectral to gridpoint

Convert all spectral fields to a global regular Gaussian grid. The optional parameter **trunc** must be greater than the input truncation.

gp2sp

Gridpoint to spectral

Convert all Gaussian gridpoint fields to spectral fields. The optional parameter **trunc** must be lower than the input truncation.

Parameters

type

[STRING] Type of the grid: quadratic, linear, cubic (default: type=quadratic)

trunc

[INTEGER] Triangular truncation

Note

To speed up the calculations, the Legendre polynoms are kept in memory. This requires a relatively large amount of memory. This is for example 12GB for T1279 data.

Example

To transform spectral coefficients from T106 to F80 regular Gaussian grid use:

```
cdo sp2gp infile outfile
```

To transform spectral coefficients from TL159 to F80 regular Gaussian grid use:

```
cdo sp2gp,type=linear infile outfile
```

Author

Uwe Schulzweida

2.13.2 Specconv

Name

sp2sp - Spectral to spectral

Synopsis

cdo sp2sp,*trunc infile outfile*

Description

Changed the triangular truncation of all spectral fields. This operator performs downward conversion by cutting the resolution. Upward conversions are achieved by filling in zeros.

Parameters

trunc

[INTEGER] New spectral resolution

Author

Uwe Schulzweida

2.13.3 Wind2

Name

dv2ps - D and V to vel. potential and stream function

Synopsis

cdo dv2ps *infile outfile*

Description

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.

2.13.4 Wind

Name

dv2uv, uv2dv - Wind transformation

Synopsis

cdo <operator>[,gridtype] infile outfile

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 global regular Gaussian grid. The Gaussian latitudes need to be ordered from north to south. Missing values are not supported.

The relationship between the spectral resolution, governed by the truncation number T, and the grid resolution depends on the number of grid points at which the shortest wavelength field is represented. For a grid with 2N points between the poles (so 4N grid points in total around the globe) the relationship is:

linear grid: the shortest wavelength is represented by 2 grid points $\rightarrow 4N \simeq 2(TL + 1)$

quadratic grid: the shortest wavelength is represented by 3 grid points $\rightarrow 4N \simeq 3(TQ + 1)$

cubic grid: the shortest wavelength is represented by 4 grid points $\rightarrow 4N \simeq 4(TC + 1)$

The quadratic grid is used by ECHAM and ERA15. ERA40 is using a linear Gaussian grid reflected by the TL notation.

The following table shows the calculation of the number of latitudes and the triangular truncation for the different grid types:

Gridtype	Number of latitudes: nlat	Triangular truncation: ntr
linear	$NINT((ntr*2 + 1)/2)$	$(nlat*2 - 1) / 2$
quadratic	$NINT((ntr*3 + 1)/2)$	$(nlat*2 - 1) / 3$
cubic	$NINT((ntr*4 + 1)/2)$	$(nlat*2 - 1) / 4$

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.

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.

Parameters

gridtype

[STRING] Type of the grid: quadratic, linear, cubic (default: quadratic)

Note

To speed up the calculations, the Legendre polynoms are kept in memory. This requires a relatively large amount of memory. This is for example 12GB for T1279 data.

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 infile outfile
```

Author

Uwe Schulzweida

2.13.5 Fourier

Name

fourier - Fourier transformation

Synopsis

cdo *fourier,epsilon infile outfile*

Description

The fourier operator performs the fourier transformation or the inverse fourier transformation of all input fields. If the number of timesteps is a power of 2 then the algorithm of the Fast Fourier Transformation (FFT) is used.

It is

$$o(t, x) = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} i(t, x) e^{\epsilon 2\pi i j}$$

where a user given *epsilon* = -1 leads to the forward transformation and a user given *epsilon* = 1 leads to the backward transformation.

If the input stream *infile* consists only of complex fields, then the fields of *outfile*, computed by

```
cdo -f ext fourier,1 -fourier,-1 infile outfile
```

are the same than that of *infile*. For real input files see function *retocomplex*.

Parameters

epsilon

[INTEGER] -1: forward transformation; 1: backward transformation

Note

Complex numbers can only be stored in NetCDF4 and EXTRA format.

Author

Uwe Schulzweida

2.14 Import/Export

This section contains modules to import and export data files which cannot read or write directly with **CDO**.

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

<i>Importbinary</i>	<i>import_binary</i>	Import binary data sets
<i>Importcmsaf</i>	<i>import_cmsaf</i>	Import CM-SAF HDF5 files
<i>Input</i>	<i>input</i>	ASCII input
	<i>inputsrv</i>	SERVICE ASCII input
	<i>inputtext</i>	EXTRA ASCII input
<i>Output</i>	<i>output</i>	ASCII output
	<i>outputf</i>	Formatted output
	<i>outputint</i>	Integer output
	<i>outputsrv</i>	SERVICE ASCII output
	<i>outputtext</i>	EXTRA ASCII output
<i>Outputtab</i>	<i>outputtab</i>	Table output
<i>Outputgmt</i>	<i>gmtxyz</i>	GMT xyz format
	<i>gmtcells</i>	GMT multiple segment format

2.14.1 Importbinary

Name

import_binary - Import binary data sets

Synopsis

cdo import_binary *infile.ctl outfile*

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 infile.ctl outfile.nc
```

Here is an example of a GrADS data descriptor file:

```
DSET ^infile.bin
OPTIONS sequential
UNDEF -9e+33
XDEF 360 LINEAR -179.5 1
YDEF 180 LINEAR -89.5 1
ZDEF 1 LINEAR 1 1
TDEF 1 LINEAR 00:00Z15jun1989 12hr
VARS 1
param 1 99 description of the variable
ENDVARS
```

The binary data file infile.bin contains one parameter on a global 1 degree lon/lat grid written with FORTRAN record length headers (sequential).

Author

Uwe Schulzweida

2.14.2 Importcmsaf

Name

import_cmsaf - Import CM-SAF HDF5 files

Synopsis

cdo import_cmsaf *infile outfile*

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] library (version 5.0 or higher) is needed for full support of the remapping functionality.

Example

A typical sequence of commands with this operator could look like this:

```
cdo -f nc remapbil,r360x180 -import_cmsaf cmsaf_product.hdf output.nc
```

(bilinear remapping to a predefined global grid with 1 deg resolution and conversion to NetCDF).

If you work with CM-SAF data on original satellite project, an additional file with information on geolocation is required, to perform such spatial operations:

```
cdo -f nc remapbil,r720x360 -setgrid,cmsaf_latlon.h5 -import_cmsaf cmsaf.hdf out.nc
```

Some CM-SAF data are stored as scaled integer values. For some operations, it could be desirable (or necessary) to increase the accuracy of the converted products:

```
cdo -b f32 -f nc fldmean -sellonlatbox,0,10,0,10 -remapbil,r720x360 \  
-import_cmsaf cmsaf_product.hdf output.nc
```

Author

Uwe Schulzweida, Frank Kaspar

2.14.3 Input

Name

input, inputsrv, inputtext - Formatted input

Synopsis

cdo <operator> outfile

cdo input,grid[,zaxis] outfile

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 *outfile*. The numbers read are exactly that ones which are written out by the *output* operator.

inputsrv

SERVICE ASCII input

Reads fields with ASCII numbers from standard input and stores them in *outfile*. Each field should have a header of 8 integers (SERVICE likely). The numbers that are read are exactly that ones which are written out by the *outputsrv* operator.

inputtext

EXTRA ASCII input

Read fields with ASCII numbers from standard input and stores them in *outfile*. Each field should have header of 4 integers (EXTRA likely). The numbers read are exactly that ones which are written out by the *outputtext* operator.

Parameters

grid

[STRING] Grid description file or name

zaxis

[STRING] Z-axis description file

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 outfile.grb < my_ascii_data
```

Author

Uwe Schulzweida

2.14.4 Output

Name

output, outputf, outputint, outputsrv, outputtext - Formatted output

Synopsis

cdo <operator> *infile*s

cdo outputf[,*format*[,*nelem*]] *infile*s

Description

This module prints all values of all input datasets to standard output. All input fields need to have the same horizontal grid. All input files need to have the same structure with the same variables. The format of the output depends on the chosen operator.

Operators

output

ASCII output

Prints all values to standard output. Each row has 6 elements with the C-style format "%13.6g".

outputf

Formatted output

Prints all values to standard output. The format and number of elements for each row have to be specified by the parameters **format** and **nelem**. The default for **nelem** is 1.

outputint

Integer output

Prints all values rounded to the nearest integer to standard output.

outputsrv

SERVICE ASCII output

Prints all values to standard output. Each field with a header of 8 integers (SERVICE likely).

outputtext

EXTRA ASCII output

Prints all values to standard output. Each field with a header of 4 integers (EXTRA likely).

Parameters

format

[STRING] C-style format for one element (e.g. %13.6g)

nelem

[INTEGER] Number of elements for each row (default: nelem = 1)

Example

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

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

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

Author

Uwe Schulzweida

2.14.5 Outputtab

Name

outputtab - Table output

Synopsis

cdo outputtab,parameters infiles

Description

This operator prints a table of all input datasets to standard output. **infiles** is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. All input fields need to have the same horizontal grid.

The contents of the table depends on the chosen parameters. The format of each table parameter is keyname[:len]. len is the optional length of a table entry. The number of significant digits of floating point parameters can be set with the **CDO** option --precision, the default is 7. Here is a list of all valid keynames:

Keyname	Type	Description
value	FLOAT	Value of the variable [len:8]
name	STRING	Name of the variable [len:8]
param	STRING	Parameter ID (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]]) [len:11]
code	INTEGER	Code number [len:4]
x	FLOAT	X coordinate of the original grid [len:6]
y	FLOAT	Y coordinate of the original grid [len:6]
lon	FLOAT	Longitude coordinate in degrees [len:6]
lat	FLOAT	Latitude coordinate in degrees [len:6]
lev	FLOAT	Vertical level [len:6]
xind	INTEGER	Grid x index [len:4]
yind	INTEGER	Grid y index [len:4]
timestep	INTEGER	Timestep number [len:6]
date	STRING	Date (format YYYY-MM-DD) [len:10]
time	STRING	Time (format hh:mm:ss) [len:8]
year	INTEGER	Year [len:5]
month	INTEGER	Month [len:2]
day	INTEGER	Day [len:2]
nohead	INTEGER	Disable output of header line

Parameters

keynames

[STRING] Comma-separated list of keynames, one for each column of the table

Example

To print a table with name, date, lon, lat and value information use:

```
cdo outputtab,name,date,lon,lat,value infile
```

Here is an example output of a time series with the yearly mean temperature at lon=10/lat=53.5:

#	name	date	lon	lat	value
	tsurf	1991-12-31	10	53.5	8.83903
	tsurf	1992-12-31	10	53.5	8.17439
	tsurf	1993-12-31	10	53.5	7.90489

(continues on next page)

(continued from previous page)

tsurf	1994-12-31	10	53.5	10.0216
tsurf	1995-12-31	10	53.5	9.07798

Author

Uwe Schulzweida

2.14.6 Outputgmt

Name

gmtxyz, gmtcells - GMT output

Synopsis

cdo <operator> *infile*

Description

This module prints the first field of the input dataset to standard output. The output can be used to generate 2D Lon/Lat plots with [GMT]. The format of the output depends on the chosen operator.

Operators

gmtxyz

GMT xyz format

The operator exports the first field to the GMT xyz ASCII format. The output can be used to create contour plots with the GMT module pscontour.

gmtcells

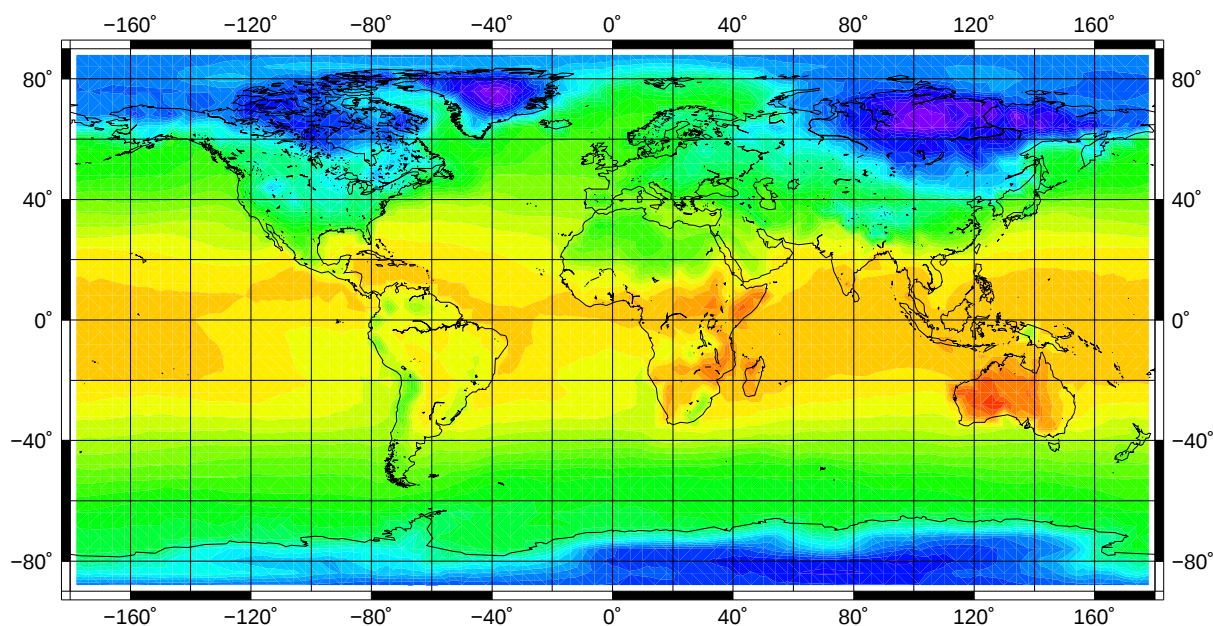
GMT multiple segment format

The operator exports the first field to the GMT multiple segment ASCII format. The output can be used to create shaded gridfill plots with the GMT module psxy.

Example

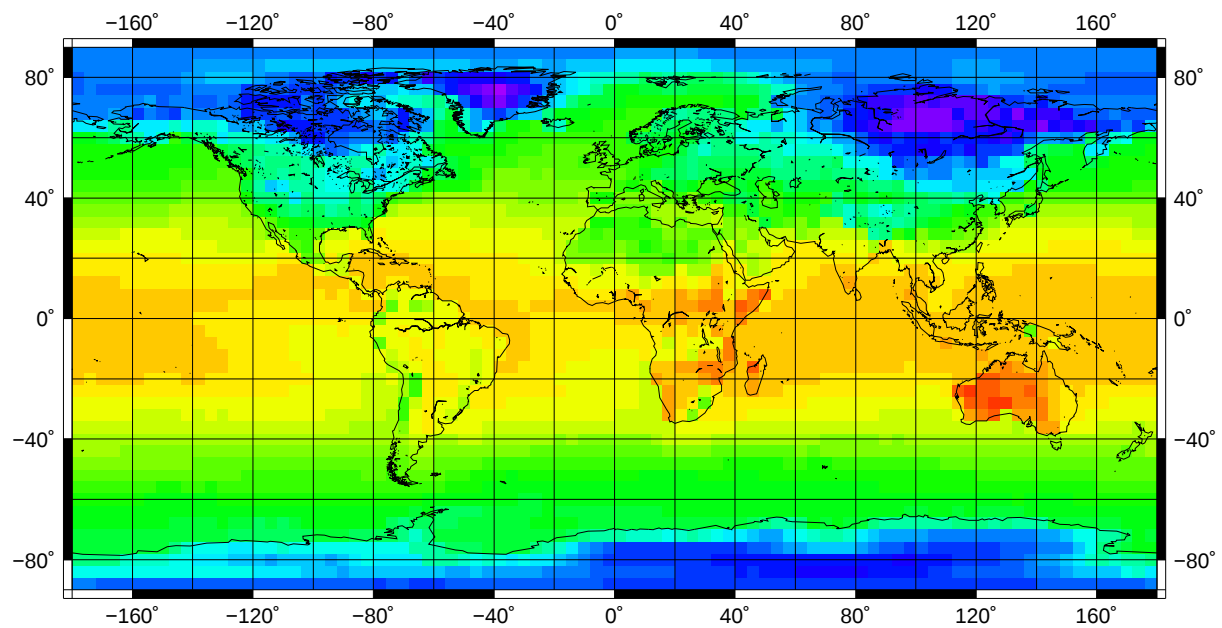
1) GMT shaded contour plot of a global temperature field with a resolution of 4 degree. The contour interval is 3 with a rainbow color table:

```
cdo gmtxyz temp > data.gmt
makecpt -T213/318/3 -Crainbow > gmt.cpt
pscontour -K -JQ0/10i -Rd -I -Cgmt.cpt data.gmt > gmtplot.ps
pscoast -O -J -R -Dc -W -B40g20 >> gmtplot.ps
```



2) GMT shaded gridfill plot of a global temperature field with a resolution of 4 degree. The contour interval is 3 with a rainbow color table:

```
cdo gmtcells temp > data.gmt  
makecpt -T213/318/3 -Crainbow > gmt.cpt  
psxy -K -JQ0/10i -Rd -L -Cgmt.cpt -m data.gmt > gmtplot.ps  
pscoast -O -J -R -Dc -W -B40g20 >> gmtplot.ps
```



Author

Uwe Schulzweida

2.15 Graphic with Magics

Magics is the latest generation of the ECMWF's Meteorological plotting software MAGICS. Magics supports the plotting of contours, wind fields, observations, satellite images, symbols, text, axis and graphs (including box plots). Data fields to be plotted may be presented in various formats, for instance GRIB 1 and 2 code data, gaussian grid, regularly spaced grid and fitted data, BUFR and NetCDF format or retrieved from an ODB database. The produced meteorological plots can be saved in various formats, such as PostScript, EPS, PDF, GIF, PNG and SVG. [\[Magics\]](#)

In order to rapidly generate high quality pictures from the data obtained from the existing **CDO** operators, the **CDO** has been interfaced with the Magics library. As a first step, some **CDO** plotting operators are created to cater to the most essential/frequently used plotting features viz., graph, contour, vector. These operators rely on the Magics and generate output files in the various formats supported by Magics. These operators can be used as terminal operators and chained with the existing operators.

Magics provides a vast number of parameters to control the attributes of various plotting features. Keeping in view, the usability of **CDO** users, currently only a few of these parameters are supported and accessible to the **CDO** users as command line arguments for the respective operators. The users are requested to refer to the Magics manual [\[Magics\]](#) for detailed description of the various parameters available for the various features. The description of the plotting operators and the various arguments that can be passed to these operators is provided in the subsequent sections.

This section gives a description of all **CDO** operators to generate plots with magics. These operators were implemented by Kameswarrao Modali (DKRZ) in 2013.

Note

Support for these **CDO** operators is limited.

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

<i>Magplot</i>	<i>contour</i>	Contour plot
	<i>shaded</i>	Shaded contour plot
	<i>grfill</i>	Shaded gridfill plot
<i>Magvector</i>	<i>vector</i>	Lon/Lat vector plot
<i>Maggraph</i>	<i>graph</i>	Line graph plot

2.15.1 Magplot

Name

contour, shaded, grfill - Lon/Lat plot

Synopsis

cdo <operator>.parameter infile obase

Description

The operators in this module generates 2D Lon/Lat plots. The data for the plot is read from `infile`. Only data on rectilinear Lon/Lat grids are supported. The output file will be named <obase>_<param>.<device> where param is the parameter name and device is the device name. The default output file format is PostScript, this can be changed with the device parameter. The type of the plot depends on the chosen operator.

Here is a list of all common plot parameters:

Keyname	Type	Description
device	STRING	Output device (ps, eps, pdf, png, gif, gif_animation, jpeg, svg, kml)
projection	STRING	Projection (cylindrical, polar_stereographic, robinson, mercator)
style	STRING	Contour line style (solid, dash, dot, chain_dash, chain_dot)
min	FLOAT	Minimum value
max	FLOAT	Maximum value
lon_max	FLOAT	Maximum longitude of the image
lon_min	FLOAT	Minimum longitude of the image
lat_max	FLOAT	Maximum latitude of the image
lat_min	FLOAT	Minimum latitude of the image
count	INTEGER	Number of Contour levels / Colour bands
interval	FLOAT	Interval in data units between two bands lines
list	INTEGER	List of levels to be plotted
RGB	STRING	TRUE or FALSE, to indicate, if the input colour is in RGB format
step_freq	INTEGER	Frequency of time steps to be considered for making the animation (device=gif_animation). Default value is "1" (all time steps). Will be ignored if input file has multiple variables.
file_split	STRING	TRUE or FALSE, to split the output file for each variable, if input has multiple variables. Default value is "FALSE". Valid only for "PS" format.

Operators

contour

Contour plot

The operator **contour** generates the discrete contour lines of the input field values. The following additional parameters are valid for contour operator, module in addition to the common plot parameters:

Keyname	Type	Description
colour	STRING	Colour for drawing the contours
thickness	FLOAT	Thickness of the contour line
style	STRING	Line Style can be "SOLID", "DASH", "DOT", "CHAIN_DASH", "CHAIN_DOT"

shaded

Shaded contour plot

The operator **shaded** generates the filled contours of the given input field values. The following additional parameters are valid for shaded contour and gridfill operator, in addition to the common plot parameters.

Keyname	Type	Description
colour_min	STRING	Colour for the Minimum colour band
colour_max	STRING	Colour for the Maximum colour band
colour_triad	STRING	Direction of colour sequencing for shading "CW" or "ACW", to denote "clockwise" and "anticlockwise" respectively. To be used in conjunction with "colour_min", "colour_max" options. Default is "ACW"
colour_table	STRING	File with user specified colours with the format as

Example file for 6 colours in RGB format:

```
6
RGB(0.0;0.0;1.0)
RGB(0.0;0.0;0.5)
RGB(0.0;0.5;0.5)
RGB(0.0;1.0;0.0)
RGB(0.5;0.5;0.0)
RGB(1.0;0.0;0.0)
```

grfill

Shaded gridfill plot

The operator **grfill** is similar to satellite imaging and shades each cell (pixel) according to the value of the field at that cell.

Parameters

parameter

[STRING] Comma-separated list of plot parameters

Note

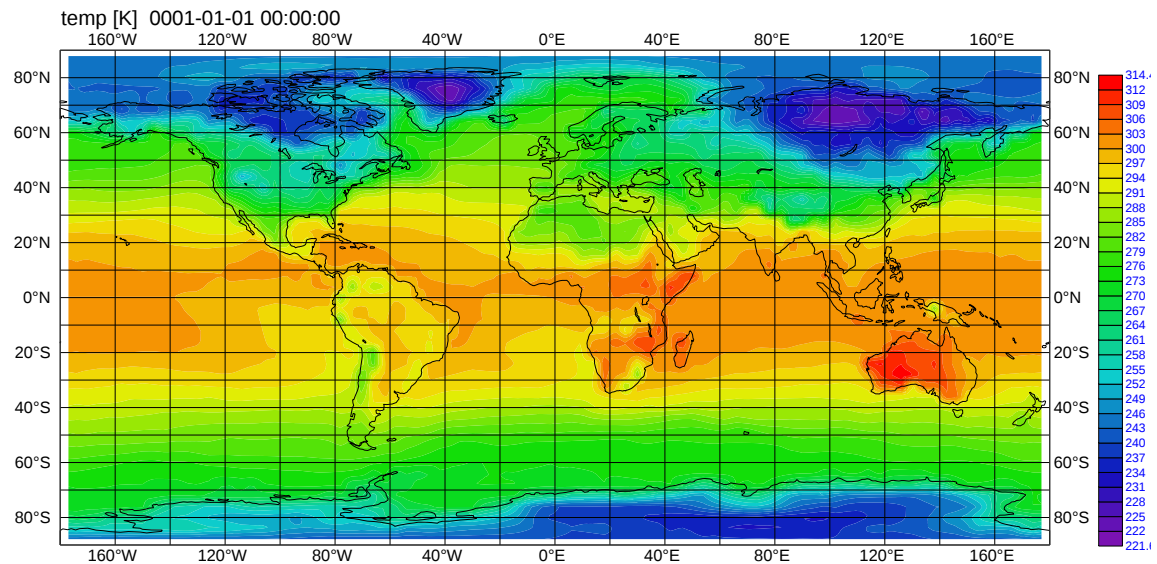
All colour parameter can be either standard name or in RGB format. The valid standard name strings for "colour" are:

"red", "green", "blue", "yellow", "cyan", "magenta", "black", "avocado", "beige", "brick", "brown", "burgundy", "charcoal", "chestnut", "coral", "cream", "evergreen", "gold", "grey", "khaki", "kellygreen", "lavender", "mustard", "navy", "ochre", "olive", "peach", "pink", "rose", "rust", "sky", "tan", "tangerine", "turquoise", "violet", "reddishpurple", "purplered", "purplishred", "orangishred", "redorange", "reddishorange", "orange", "yellowishorange", "orangeyellow", "orangishyellow", "greenishyellow", "yellowgreen", "yellowishgreen", "bluishgreen", "bluegreen", "greenishblue", "purplishblue", "bluepurple", "bluishpurple", "purple", "white"

Example

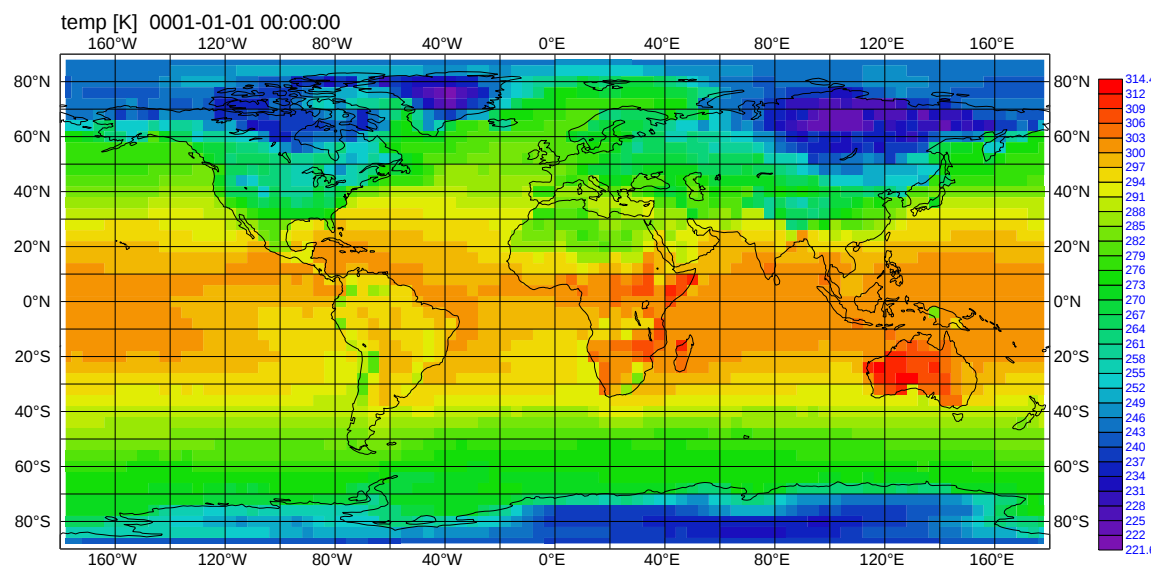
- 1) Shaded contour plot of a global temperature field with a resolution of 4 degree. The contour interval is 3 with a rainbow color table.

```
cdo shaded,interval=3,colour_min=violet,colour_max=red,colour_triad=cw temp plot
```



- 2) Shaded gridfill plot of a global temperature field with a resolution of 4 degree. The contour interval is 3 with a rainbow color table.

```
cdo grfill,interval=3,colour_min=violet,colour_max=red,colour_triad=cw temp plot
```



Author

Kameswarrao Modali

2.15.2 Magvector

Name

vector - Lon/Lat vector plot

Synopsis

cdo vector,parameter infile obase

Description

This operator generates 2D Lon/Lat vector plots. The data for the plot is read from **infile**. The input is expected to contain two velocity components. Only data on rectilinear Lon/Lat grids are supported. The output file will be named <obase>.<device> where device is the device name. The default output file format is PostScript, this can be changed with the device parameter.

Here is a list of all vector plot parameters:

Keyname	Type	Description
device	STRING	Output device (ps, eps, pdf, png, gif, gif_animation, jpeg, svg, kml)
projection	STRING	Projection (cylindrical, polar_stereographic, robinson, mercator)
thin_fac	FLOAT	Controls the actual number of wind arrows or flags plotted (default 2).
unit_vec	FLOAT	Wind speed in m/s represented by a unit vector (1.0cm)
step_freq	INTEGER	Frequency of time steps to be considered for making the animation (device=gif_animation). Default value is "1" (all time steps). Will be ignored if input file has multiple variables.

Parameters

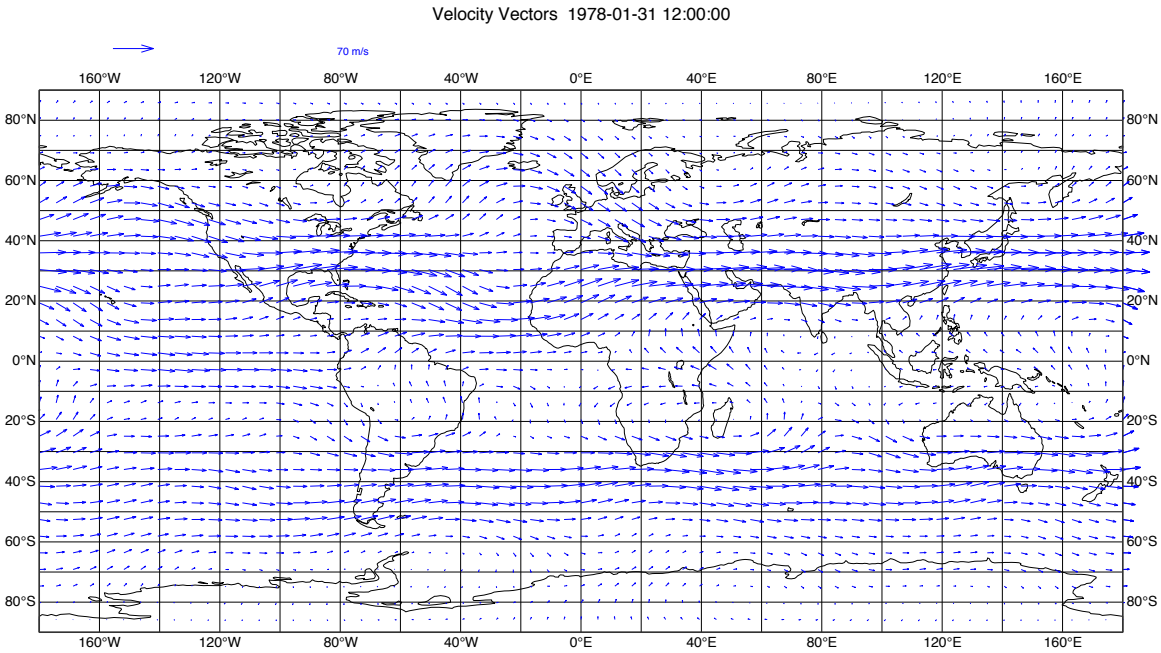
parameter

[STRING] Comma-separated list of plot parameters

Example

Vector plot of global wind vectors with a resolution of 5 degree. The unit vector is set to 70 and all wind arrows are plotted.

```
cdo vector,thin_fac=1,unit_vec=70 uvdata plot
```



Author

Kameswarrao Modali

2.15.3 Maggraph

Name

graph - Line graph plot

Synopsis

cdo graph,*parameter infiles obase*

Description

This operator generates line graph plots. The data for the plot is read from *infiles*. The result is written to *out file*. The default output file format is PostScript, this can be changed with the device parameter.

Here is a list of all graph plot parameters:

Keyname	Type	Description
device	STRING	Output device (ps, eps, pdf, png, gif, gif_animation, jpeg, svg, kml)
ymin	FLOAT	Minimum value of the y-axis data
ymax	FLOAT	Maximum value of the y-axis data
linewidth	INT	Line width (default 8)
stat	STRING	"TRUE" or "FALSE", to switch on the mean computation. Default is "FALSE". Will be overridden to "FALSE", if input files have unequal number of time steps or different start/end times.
sigma	FLOAT	Standard deviation value for generating shaded back ground around the mean value. To be used in conjunction with 'stat="TRUE"'
obsv	STRING	To indicate if the input files have an observation data, by setting to "TRUE". Default value is "FALSE". The observation data should be the first file in the input file list. The observation data is always plotted in black colour.

Parameters

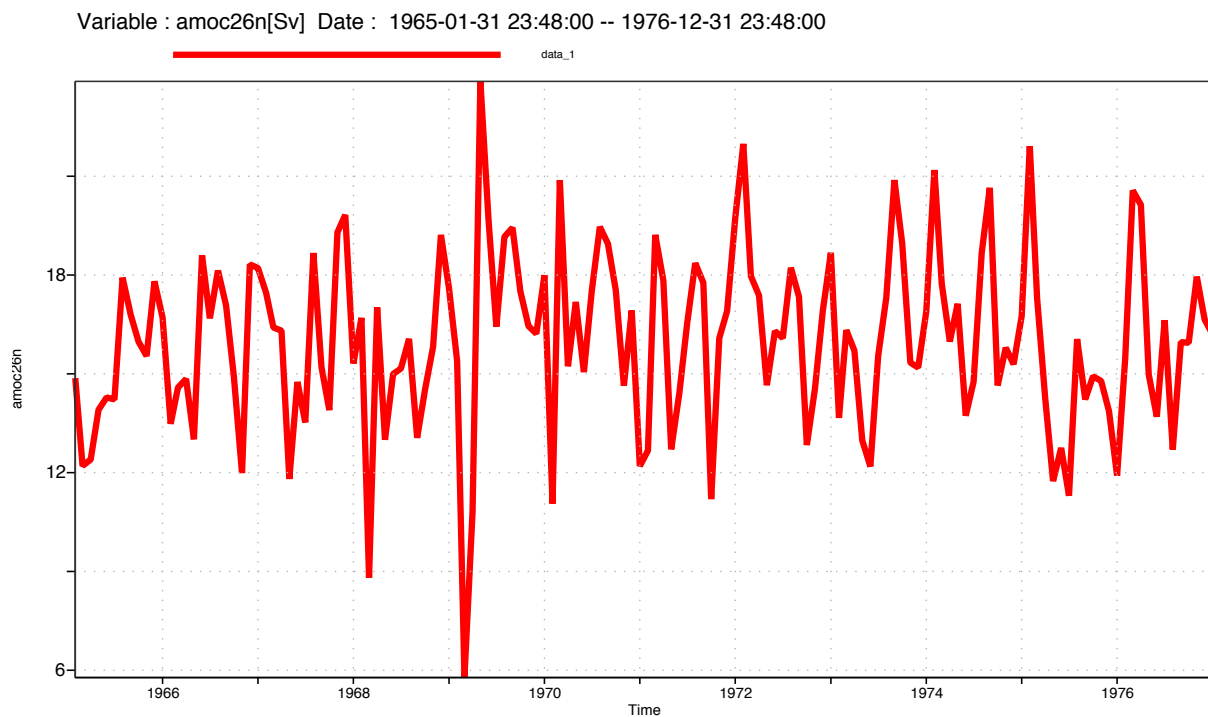
parameter

[STRING] Comma-separated list of plot parameters

Example

Graph plot of an atlantic MOC time series from 1965 to 1976:

```
cdo graph amoc plot
```



Author

Kameswarrao Modali

2.16 Climate Model Output Rewriting

This section contains the CMOR operator.

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

<i>CMOR</i>	<i>cmor</i>	Climate Model Output Rewriting to produce CMIP-compliant data
-------------	-------------	---

2.16.1 CMOR

Name

cmor - Climate Model Output Rewriting to produce CMIP-compliant data

Synopsis

```
cdo cmor,MIPtable[,cmor_name=VarList[,key=value[,...]]] infile
```

Description

The **CDO** operator **cmor** converts an *infile* into a CMIP-compliant format by using the CMOR library. Each output file contains a single output variable. The name of the output files are generated by CMOR according to a template based on the DRS (Data reference Syntax) of the project. CMOR checks and applies the information delivered through the project dependent *MIPtable* on the *infile*. Additional information which is required for the conversion can be configured via keyvalues as optional parameters.

By specifying a variable selector keyvalue, e.g. *cmor_name=tas*, the user can pre-select a subset of *infile* variables. If *name* or *code* is specified, a corresponding *cmor_name* which can also be found in the *MIPtable* is also required to map the *infile* variable to the CMOR-variable. For mapping more variables at the operator call, one can specify a mapping table via keyword *mapping_table*.

Global attributes must be collected in info files and can be specified via keyword *info*. All required and optional global attributes as well as information about table file formats are given in the 'cdo cmor manual'.

If questions remain, do not hesitate to ask and send an email to wachsmannATdkrz.de.

Parameters

MIPtable

[STRING] Name of the MIP table as used by CMOR.

cmor_name | cn

[STRING]

Variable selector and specified in the MIP table.

Comma-separated list of CMOR variable names.

Default is to process all variables.

name | n

[STRING]

Variable selector.

Name of a selected *infile* variable.

code | c

[INTEGER]

Variable selector.

Three digits (GRIB) Code of a selected *infile* variable.

info | i

[STRING]

Preprocessing.

Comma-separated list of filenames.

Contains global attributes and control keywords.

Default: CWD/.cdocmorinfo

grid_info | gi

[STRING]

Preprocessing.

NetCDF or table formatted file with model grid description.

Horizontal and vertical axes are substituted with the ones from grid info file.

mapping_table | mt

[STRING]

Preprocessing.

Fortran Namelist containing variable information e.g. renaming.

keep_all_attributes | kaa

[STRING]

Preprocessing.

'y' for passing all infile attributes. 'n' for discarding all infile attributes.

drs | d

[CHARACTER]

Output control.

Do(=y, default) or do not(=n) move output into the project DRS structure.

drs_root | dr

[STRING]

Output control. CMOR output root directory.

Default: CWD.

output_mode | om

[CHARACTER] Output control. Either 'r' for replace (default) or 'a' for append mode.

last_chunk | lc

[STRING] Output control. Filename of chunk to which shall be appended.

max_size | ms

[INTEGER] Output control. Limit of output file size in GigaByte.

deflate_level | dl

[INTEGER] Output control. Compression level. -1: No compression. 0: Only shuffle.

version_date | vd

[INTEGER] Output control. Subdirectory name in CMIP6 DRS.

required_time_units | rtu

[STRING]

Temporal description.

Time axis reference date specified by the experiment.

Format: 'days since YYYY-day-month hh:mm:ss'.

cell_methods | cm

[CHARACTER]

Temporal description.

Cell_methods of time axis.

Value is one of 'm' (default), 'p', 'c', 'n', 'd'

units | u

[STRING]

Variable attribute. Units of the variable.

Must be known by library UDunits.

variable_comment | vc

[STRING] Variable attribute. Variable comment.

positive | p

[CHARACTER] Variable attribute. Positive flux direction, either 'u' for upward or 'd' for downward.

z_axis | za

[STRING] Name of the coordinate variable associated with the z-axis of the target variable.

character_axis | ca

[STRING]

Name of the coordinate variable associated with a character axis of the target variable.

Valid axes names are: basin, vegtype or oline.

t_axis | ta

[STRING]

Sets time values and time bounds to the nearest value required by the project given by the value of t_axis.

Valid value is: cmip

Example

Process file with CMOR. In this case a grid mapping is used. The file `cmor.rc` contains metadata CMOR expects for http://cmor.llnl.gov/mydoc_cmor3_api/#cmordatasetjson `cmor_dataset_json()`:

```
cdo cmor,Tables/CMIP6_day.json,__grid_table=Tables/CMIP6_grids.json,__info=cmor.rc,
↪ CNRM-CERFACS_r11c.nc
```

Author

Fabian Wachsmann

2.17 Climate indices

This section contains modules to compute climate indices of daily temperature and precipitation extremes. The definitions of these climate indices are from the European Climate Assessment (ECA) project.

The climate indices were implemented in **CDO** by Ralf Quast (Brockmann Consult) on behalf of the Service Gruppe Anpassung (SGA) in 2006. SGA was part of the Model and Data Group (M&D) at the MPI for Meteorology. In 2010, the Model and Data Group became the Data Management department at DKRZ (Deutsches Klimarechenzentrum) and the SGA was disintegrated.

Around 2018, Fabian Wachsmann (DKRZ) added a number of ETCCDI conforming indices.

Note

Support for these **CDO** operators is limited.

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

<i>Eca_cdd</i>	<i>eca_cdd</i>	Consecutive dry days index per time period
	<i>etccdi_cdd</i>	Consecutive dry days index per time period
<i>Eca_cfd</i>	<i>eca_cfd</i>	Consecutive frost days index per time period
<i>Eca_csu</i>	<i>eca_csu</i>	Consecutive summer days index per time period
<i>Eca_cwd</i>	<i>eca_cwd</i>	Consecutive wet days index per time period
	<i>etccdi_cwd</i>	Consecutive wet days index per time period
<i>Eca_cwdi</i>	<i>eca_cwdi</i>	Cold wave duration index wrt mean of reference period
<i>Eca_cwfi</i>	<i>eca_cwfi</i>	Cold-spell days index wrt 10th percentile of reference period
	<i>etccdi_csdi</i>	Cold-spell duration index
<i>Eca_etr</i>	<i>eca_etr</i>	Intra-period extreme temperature range
<i>Eca_fd</i>	<i>eca_fd</i>	Frost days index per time period
	<i>etccdi_fd</i>	Frost days index per time period
<i>Eca_gsl</i>	<i>eca_gsl</i>	Thermal Growing season length index
<i>Eca_hd</i>	<i>eca_hd</i>	Heating degree days per time period
<i>Eca_hwdi</i>	<i>eca_hwdi</i>	Heat wave duration index wrt mean of reference period
<i>Eca_hwfi</i>	<i>eca_hwfi</i>	Warm spell days index wrt 90th percentile of reference period
	<i>etccdi_wsdi</i>	Warm Spell Duration Index
<i>Eca_id</i>	<i>eca_id</i>	Ice days index per time period
	<i>etccdi_id</i>	Ice days index per time period
<i>Eca_r75p</i>	<i>eca_r75p</i>	Moderate wet days wrt 75th percentile of reference period
<i>Eca_r75ptot</i>	<i>eca_r75ptot</i>	Precipitation percent due to R75p days
<i>Eca_r90p</i>	<i>eca_r90p</i>	Wet days wrt 90th percentile of reference period

continues on next page

Table 14 – continued from previous page

<i>Eca_r90ptot</i>	<i>eca_r90ptot</i>	Precipitation percent due to R90p days
<i>Eca_r95p</i>	<i>eca_r95p</i>	Very wet days wrt 95th percentile of reference period
<i>Eca_r95ptot</i>	<i>eca_r95ptot</i>	Precipitation percent due to R95p days
<i>Eca_r99p</i>	<i>eca_r99p</i>	Extremely wet days wrt 99th percentile of reference period
<i>Eca_r99ptot</i>	<i>eca_r99ptot</i>	Precipitation percent due to R99p days
<i>Eca_pd</i>	<i>eca_pd</i>	Precipitation days index per time period
	<i>etccdi_r1mm</i>	Precipitation days index per time period
	<i>eca_r10mm</i>	Heavy precipitation days index per time period
	<i>eca_r20mm</i>	Very heavy precipitation days index per time period
<i>Eca_rr1</i>	<i>eca_rr1</i>	Wet days index per time period
<i>Eca_rx1day</i>	<i>eca_rx1day</i>	Highest one day precipitation amount per time period
	<i>etccdi_rx1day</i>	Maximum 1-day Precipitation
<i>Eca_rx5day</i>	<i>eca_rx5day</i>	Highest five-day precipitation amount per time period
	<i>etccdi_rx5day</i>	Highest five-day precipitation amount per time period
<i>Eca_sdii</i>	<i>eca_sdii</i>	Simple daily intensity index per time period
<i>Eca_su</i>	<i>eca_su</i>	Summer days index per time period
	<i>etccdi_su</i>	Summer days index per time period
<i>Eca_tg10p</i>	<i>eca_tg10p</i>	Cold days percent wrt 10th percentile of reference period
<i>Eca_tg90p</i>	<i>eca_tg90p</i>	Warm days percent wrt 90th percentile of reference period
<i>Eca_tn10p</i>	<i>eca_tn10p</i>	Cold nights percent wrt 10th percentile of reference period
<i>Eca_tn90p</i>	<i>eca_tn90p</i>	Warm nights percent wrt 90th percentile of reference period
<i>Eca_tr</i>	<i>eca_tr</i>	Tropical nights index per time period
	<i>etccdi_tr</i>	Tropical nights index per time period
<i>Eca_tx10p</i>	<i>eca_tx10p</i>	Very cold days percent wrt 10th percentile of reference period
<i>Eca_tx90p</i>	<i>eca_tx90p</i>	Very warm days percent wrt 90th percentile of reference period
<i>Eca_etccdi</i>	<i>etccdi_tx90p</i>	Percentage of Days when Daily Maximum Temperature is Above the 90th Percentile
	<i>etccdi_tx10p</i>	Percentage of Days when Daily Maximum Temperature is Below the 10th Percentile
	<i>etccdi_tn90p</i>	Percentage of Days when Daily Minimum Temperature is Above the 90th Percentile

continues on next page

Table 14 – continued from previous page

<i>etccdi_tn10p</i>	Percentage of Days when Daily Minimum Temperature is Below the 10th Percentile
<i>etccdi_r95p</i>	Annual Total Precipitation when Daily Precipitation Exceeds the 95th Percentile of Wet Day Precipitation
<i>etccdi_r99p</i>	Annual Total Precipitation when Daily Precipitation Exceeds the 99th Percentile of Wet Day Precipitation

2.17.1 Eca_cdd

Name

eca_cdd, etccdi_cdd - Consecutive dry days index per time period

Synopsis

cdo <operator>[,*R*[,*N*[,*params*]]] *infile outfile*

Description

Let *infile* 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 *N* days. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_cdd

Consecutive dry days index per time period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_cdd

Consecutive dry days index per time period

The default output frequency is yearly. Periods within overlapping years are accounted for the first year. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

R

[FLOAT] Precipitation threshold (unit: mm; default: *R*=1 mm)

N

[INTEGER] Minimum number of days exceeded (default: *N*=5)

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_cdd rrfile outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.2 Eca_cfd

Name

eca_cfd - Consecutive frost days index per time period

Synopsis

cdo eca_cfd[,*N*] *infile outfile*

Description

Let *infile* be a time series of the daily minimum temperature *TN*, then the largest number of consecutive days where $TN < 0^{\circ}\text{C}$ is counted. Note that *TN* have to be given in units of Kelvin. A further output variable is the number of frost periods of more than *N* days. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

Parameters

N

[INTEGER] Minimum number of days exceeded (default: *N*=5)

Example

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

```
cdo eca_cfd tnfile outfile
```

Author

Ralf Quast

2.17.3 Eca_csu

Name

eca_csu - Consecutive summer days index per time period

Synopsis

cdo eca_csu[,*T*[,*N*]] *infile outfile*

Description

Let *infile* be a time series of the daily maximum temperature TX, then the largest number of consecutive days where $TX > T$ is counted. The number *T* is an optional parameter with default $T = 25^{\circ}\text{C}$. Note that *TN* have to be given in units of Kelvin, whereas *T* have to be given in degrees Celsius. A further output variable is the number of summer periods of more than *N* days. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

Parameters

T

[FLOAT] Temperature threshold (unit: $^{\circ}\text{C}$; default: $T=25^{\circ}\text{C}$)

N

[INTEGER] Minimum number of days exceeded (default: $N=5$)

Example

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

```
cdo eca_csu txfile outfile
```

Author

Ralf Quast

2.17.4 Eca_cwd

Name

eca_cwd, etccdi_cwd - Consecutive wet days index per time period

Synopsis

cdo <operator>[,params] *infile outfile*

Description

Let *infile* be a time series of the daily precipitation amount *RR*, then the largest number of consecutive days where *RR* is at least *R* is counted. *R* is an optional parameter with default *R* = 1 mm. A further output variable is the number of wet periods of more than *N* days. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_cwd

Consecutive wet days index per time period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_cwd

Consecutive wet days index per time period

The default output frequency is yearly. Periods within overlapping years are accounted for the first year. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

R

[FLOAT] Precipitation threshold (unit: mm; default: R=1mm)

N

[INTEGER] Minimum number of days exceeded (default: N=5)

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_cwd rrfile outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.5 Eca_cwdi

Name

eca_cwdi - Cold wave duration index wrt mean of reference period

Synopsis

```
cdo eca_cwdi[,nday[,T]] infile1 infile2 outfile
```

Description

Let **infile1** be a time series of the daily minimum temperature TN, and let **infile2** be the mean TNnorm of daily minimum temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least *nday* consecutive days, $TN < TN_{norm} - T$. The numbers *nday* and *T* are optional parameters with default *nday* = 6 and *T* = 5°C. A further output variable is the number of cold waves longer than or equal to *nday* days. TNnorm is calculated as the mean of minimum temperatures of a five day window centered on each calendar day of a given climate reference period. Note that both TN and TNnorm have to be given in the same units. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile1**.

Parameters

nday

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

T

[FLOAT] Temperature offset (unit: °C; default: T=5°C)

Example

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

```
cdo eca_cwdi tnfile tnnormfile outfile
```

Author

Ralf Quast

2.17.6 Eca_cwfi

Name

eca_cwfi, etccdi_csdi - Cold-spell days index wrt 10th percentile of reference period

Synopsis

cdo <operator>[,nday[,params]] *infile1 infile2 outfile*

Description

Let *infile1* be a time series of the daily mean temperature TG, and *infile2* be the 10th percentile TGn10 of daily mean temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least *nday* consecutive days, $TG < TGn10$. The number *nday* is an optional parameter with default *nday* = 6. A further output variable is the number of cold-spell periods longer than or equal to *nday* days. TGn10 is calculated as the 10th percentile of daily mean temperatures of a five day window centered on each calendar day of a given climate reference period. Note that both TG and TGn10 have to be given in the same units.

Operators

eca_cwfi

Cold-spell days index wrt 10th percentile of reference period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_csdi

Cold-spell duration index

The default output frequency is yearly. Periods within overlapping years are accounted for the first year. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

nday

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

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_cwfi tgfile tgn10file outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.7 Eca_etr

Name

eca_etr - Intra-period extreme temperature range

Synopsis

```
cdo eca_etr infile1 infile2 outfile
```

Description

Let *infile1* and *infile2* 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 *outfile* is the date of the last contributing timesteps in *infile1* and *infile2*.

Example

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

```
cdo eca_etr txfile tnfile outfile
```

Author

Ralf Quast

2.17.8 Eca_fd

Name

eca_fd, etccdi_fd - Frost days index per time period

Synopsis

cdo <operator>[,parameter] *infile outfile*

Description

Let *infile* be a time series of the daily minimum temperature *TN*, then the number of days where $TN < 0^{\circ}\text{C}$ is counted. Note that *TN* have to be given in units of Kelvin. Parameter is a comma-separated list of "key=value" pairs.

Operators

eca_fd

Frost days index per time period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_fd

Frost days index per time period

The default output frequency is yearly. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_fd tnfile outfile
```

Author

Ralf Quast

2.17.9 Eca_gsl

Name

eca_gsl - Thermal Growing season length index

Synopsis

```
cdo eca_gsl[,nday[,T[,fland]]] infile1 infile2 outfile
```

Description

Let **infile1** be a time series of the daily mean temperature TG, and **infile2** be a land-water mask. Within a period of 12 months, the thermal growing season length is officially defined as the number of days between:

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

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

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

The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile**.

Parameters

nday

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

T

[FLOAT] Temperature threshold (unit: °C; default: T=5°C)

fland

[FLOAT] Land fraction threshold (default: fland=0.5)

Example

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

```
cdo eca_gsl tgfile maskfile outfile
```

Author

Ralf Quast

2.17.10 Eca_hd

Name

eca_hd - Heating degree days per time period

Synopsis

cdo eca_hd[,*T1*[,*T2*]] *infile outfile*

Description

Let *infile* 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.

Parameters

T1

[FLOAT] Temperature limit (unit: °C; default: T1=17°C)

T2

[FLOAT] Temperature limit (unit: °C; default: T2=T1)

Example

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

```
cdo eca_hd tgfile outfile
```

Author

Ralf Quast

2.17.11 Eca_hwdi

Name

eca_hwdi - Heat wave duration index wrt mean of reference period

Synopsis

```
cdo eca_hwdi[,nday[,T]] infile1 infile2 outfile
```

Description

Let *infile1* be a time series of the daily maximum temperature TX, and let *infile2* 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, $TX > TX_{norm} + T$. The numbers *nday* and *T* are optional parameters with default *nday*=6 and *T*=5°C. A further output variable is the number of heat waves longer than or equal to *nday* days. TXnorm is calculated as the mean of maximum temperatures of a five day window centered 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 *outfile* is the date of the last contributing timestep in *infile1*.

Parameters

nday

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

T

[FLOAT] Temperature offset (unit: °C; default: T=5°C)

Author

Ralf Quast

2.17.12 Eca_hwfi

Name

eca_hwfi, etccdi_wsgi - Warm spell index

Synopsis

cdo <operator>[,nday[,freq]] *infile1 infile2 outfile*

Description

Let *infile1* be a time series of the daily mean temperature TG, and *infile2* be the 90th percentile TGn90 of daily mean temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least *nday* consecutive days, $TG > TGn90$. The number *nday* is an optional parameter with default *nday*=6. A further output variable is the number of warm-spell periods longer than or equal to *nday* days. TGn90 is calculated as the 90th percentile of daily mean temperatures of a five day window centered on each calendar day of a given climate reference period. Note that both TG and TGn90 have to be given in the same units. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_hwfi

Warm spell days index wrt 90th percentile of reference period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_wsgi

Warm Spell Duration Index

The default output frequency is yearly. Periods within overlapping years are accounted for the first year. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

nday

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

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_hwfi tgfile tgn90file outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.13 Eca_id

Name

eca_id, etccdi_id - Ice days index per time period

Synopsis

cdo <operator>[,freq] infile outfile

Description

Let **infile** be a time series of the daily maximum temperature TX, then the number of days where $TX < 0^{\circ}\text{C}$ is counted. Note that TX have to be given in units of Kelvin. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_id

Ice days index per time period

The operator counts over the entire time series. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile**.

etccdi_id

Ice days index per time period

The default output frequency is yearly. The date information of a timestep in **outfile** is the mid of the frequency interval.

Parameters

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_id txfile outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.14 Eca_r75p

Name

eca_r75p - Moderate wet days wrt 75th percentile of reference period

Synopsis

cdo eca_r75p *infile1 infile2 outfile*

Description

Let **infile1** be a time series **RR** of the daily precipitation amount at wet days (precipitation ≥ 1 mm) and **infile2** 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 **infile2** is generated by the operator **ydaypct1,75**. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile1**.

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 outfile
```

Author

Ralf Quast

2.17.15 Eca_r75ptot

Name

eca_r75ptot - Precipitation percent due to R75p days

Synopsis

cdo eca_r75ptot *infile1 infile2 outfile*

Description

Let **infile1** be a time series RR of the daily precipitation amount at wet days (precipitation ≥ 1 mm) and **infile2** 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 **infile2** is generated by the operator `ydaypctl,75`. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile1**.

Author

Ralf Quast

2.17.16 Eca_r90p

Name

eca_r90p - Wet days wrt 90th percentile of reference period

Synopsis

cdo eca_r90p *infile1 infile2 outfile*

Description

Let **infile1** be a time series RR of the daily precipitation amount at wet days (precipitation ≥ 1 mm) and **infile2** 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 **infile2** is generated by the operator `ydaypct1,90`.

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 outfile
```

Author

Ralf Quast

2.17.17 Eca_r90ptot

Name

eca_r90ptot - Precipitation percent due to R90p days

Synopsis

```
cdo eca_r90ptot infile1 infile2 outfile
```

Description

Let **infile1** be a time series RR of the daily precipitation amount at wet days (precipitation ≥ 1 mm) and **infile2** 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 **infile2** is generated by the operator `ydaypctl,90`. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile1**.

Author

Ralf Quast

2.17.18 Eca_r95p

Name

eca_r95p - Very wet days wrt 95th percentile of reference period

Synopsis

cdo eca_r95p *infile1 infile2 outfile*

Description

Let **infile1** be a time series RR of the daily precipitation amount at wet days (precipitation ≥ 1 mm) and **infile2** 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 **infile2** is generated by the operator ydaypct1,95. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile1**.

Example

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

```
cdo eca_r95p rrfile rrn95file outfile
```

Author

Ralf Quast

2.17.19 Eca_r95ptot

Name

eca_r95ptot - Precipitation percent due to R95p days

Synopsis

```
cdo eca_r95ptot infile1 infile2 outfile
```

Description

Let *infile1* be a time series RR of the daily precipitation amount at wet days (precipitation ≥ 1 mm) and *infile2* 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 *infile2* is generated by the operator `ydaypctl,95`. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile1*.

Author

Ralf Quast

2.17.20 Eca_r99p

Name

eca_r99p - Extremely wet days wrt 99th percentile of reference period

Synopsis

cdo eca_r99p *infile1 infile2 outfile*

Description

Let **infile1** be a time series **RR** of the daily precipitation amount at wet days (precipitation ≥ 1 mm) and **infile2** 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 $RR > RRn99$ is calculated. **RRn99** is calculated as the 99th percentile of all wet days of a given climate reference period. Usually **infile2** is generated by the operator **ydaypct1,99**. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile1**.

Example

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

```
cdo eca_r99p rrfile rrn99file outfile
```

Author

Ralf Quast

2.17.21 Eca_r99ptot

Name

eca_r99ptot - Precipitation percent due to R99p days

Synopsis

```
cdo eca_r99ptot infile1 infile2 outfile
```

Description

Let *infile1* be a time series RR of the daily precipitation amount at wet days (precipitation ≥ 1 mm) and *infile2* 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 $RR > RRn99$ 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 *infile2* is generated by the operator `ydaypctl,99`. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile1*.

Author

Ralf Quast

2.17.22 Eca_pd

Name

eca_pd, etccdi_r1mm, eca_r10mm, eca_r20mm - Precipitation days index per time period

Synopsis

cdo eca_pd,*x* *infile outfile*

cdo etccdi_r1mm[,*parameter*] *infile outfile*

cdo eca_r10mm *infile outfile*

cdo eca_r20mm *infile outfile*

Description

Let *infile* 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_r10mm* and *eca_r20mm* are specific ECA operators with a daily precipitation amount of 10 and 20 mm respectively. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile* except for the *etccdi* operator. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_pd

Precipitation days index per time period

Generic ECA operator with daily precipitation sum exceeding *x* mm.

etccdi_r1mm

Precipitation days index per time period

The default output frequency is yearly. The date information of a timestep in *outfile* is the mid of the frequency interval.

eca_r10mm

Heavy precipitation days index per time period

Specific ECA operator with daily precipitation sum exceeding 10 mm.

eca_r20mm

Very heavy precipitation days index per time period

Specific ECA operator with daily precipitation sum exceeding 20 mm.

Parameters

x

[FLOAT] Daily precipitation amount threshold in [mm]

freq

[STRING] Output frequency (year, month)

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_rr1* 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:

```
cdo eca_pd,25 infile outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.23 Eca_rr1

Name

eca_rr1 - Wet days index per time period

Synopsis

cdo eca_rr1[,*R*] *infile outfile*

Description

Let *infile* 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 *outfile* is the date of the last contributing timestep in *infile*.

Parameters

R

[FLOAT] Precipitation threshold (unit: mm; default: R=1 mm)

Example

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

```
cdo eca_rr1 rrfile outfile
```

Author

Ralf Quast

2.17.24 Eca_rx1day

Name

eca_rx1day, etccdi_rx1day - Highest one day precipitation amount per time period

Synopsis

cdo <operator>[,freq] infile outfile

Description

Let *infile* be a time series of the daily precipitation amount *RR*, then the maximum of *RR* is written to *outfile*. If the optional parameter *mode* is set to 'm' the maximum daily precipitation amounts are determined for each month. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_rx1day

Highest one day precipitation amount per time period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_rx1day

Maximum 1-day Precipitation

The default output frequency is yearly. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_rx1day rrfile outfile
```

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

```
cdo eca_rx1day,freq=month rrfile outfile
```

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

```
cdo timmax rrfile outfile
cdo monmax rrfile outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.25 Eca_rx5day

Name

eca_rx5day, etccdi_rx5day - Highest five-day precipitation amount per time period

Synopsis

cdo <operator>[,x[,params]] *infile outfile*

Description

Let *infile* be a time series of 5-day precipitation totals *RR*, then the maximum of *RR* is written to *outfile*. 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*=50mm. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_rx5day

Highest five-day precipitation amount per time period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_rx5day

Highest five-day precipitation amount per time period

The default output frequency is yearly. Periods within overlapping years are accounted for the first year. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

x

[FLOAT] Precipitation threshold (unit: mm; default: x=50mm)

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_rx5day rrfile outfile
```

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

```
cdo timmax rrfile outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.26 Eca_sdii

Name

eca_sdii - Simple daily intensity index per time period

Synopsis

```
cdo eca_sdii[,R] infile outfile
```

Description

Let *infile* be a time series of the daily precipitation amount *RR*, then the mean precipitation amount at wet days ($RR \geq R$) is written to *outfile*. *R* is an optional parameter with default $R=1\text{mm}$. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

Parameters

R

[FLOAT] Precipitation threshold (unit: mm; default: $R=1\text{ mm}$)

Example

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

```
cdo eca_sdii rrfile outfile
```

Author

Ralf Quast

2.17.27 Eca_su

Name

eca_su, etccdi_su - Summer days index per time period

Synopsis

cdo <operator>[,*T*[,*params*]] *infile outfile*

Description

Let *infile* 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 $T=25^{\circ}\text{C}$. Note that *TX* have to be given in units of Kelvin, whereas *T* have to be given in degrees Celsius. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_su

Summer days index per time period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_su

Summer days index per time period

The default output frequency is yearly. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

T

[FLOAT] Temperature threshold (unit: $^{\circ}\text{C}$; default: $T=25^{\circ}\text{C}$)

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_su txfile outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.28 Eca_tg10p

Name

eca_tg10p - Cold days percent wrt 10th percentile of reference period

Synopsis

cdo eca_tg10p *infile1 infile2 outfile*

Description

Let *infile1* be a time series of the daily mean temperature TG, and *infile2* 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 centered 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 *outfile* is the date of the last contributing timestep in *infile1*.

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 outfile
```

Author

Ralf Quast

2.17.29 Eca_tg90p

Name

eca_tg90p - Warm days percent wrt 90th percentile of reference period

Synopsis

cdo eca_tg90p *infile1 infile2 outfile*

Description

Let *infile1* be a time series of the daily mean temperature TG, and *infile2* 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 centered 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 *outfile* is the date of the last contributing timestep in *infile1*.

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 outfile
```

Author

Ralf Quast

2.17.30 Eca_tn10p

Name

eca_tn10p - Cold nights percent wrt 10th percentile of reference period

Synopsis

```
cdo eca_tn10p infile1 infile2 outfile
```

Description

Let *infile1* be a time series of the daily minimum temperature TN, and *infile2* 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 centered 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 *outfile* is the date of the last contributing timestep in *infile1*.

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 outfile
```

Author

Ralf Quast

2.17.31 Eca_tn90p

Name

eca_tn90p - Warm nights percent wrt 90th percentile of reference period

Synopsis

cdo eca_tn90p *infile1 infile2 outfile*

Description

Let *infile1* be a time series of the daily minimum temperature TN, and *infile2* be the 90th percentile TN_{n90} of daily minimum temperatures for any period used as reference. Then the percentage of time where TN > TN_{n90} is calculated. TN_{n90} is calculated as the 90th percentile of daily minimum temperatures of a five day window centered on each calendar day of a given climate reference period. Note that both TN and TN_{n90} have to be given in the same units. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile1*.

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 outfile
```

Author

Ralf Quast

2.17.32 Eca_tr

Name

eca_tr, etccdi_tr - Tropical nights index per time period

Synopsis

cdo <operator>[,*T*[,*params*]] *infile outfile*

Description

Let *infile* 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 $T = 20^{\circ}\text{C}$. Note that *TN* have to be given in units of Kelvin, whereas *T* have to be given in degrees Celsius. Parameter is a comma-separated list of "key=values" pairs.

Operators

eca_tr

Tropical nights index per time period

The operator counts over the entire time series. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

etccdi_tr

Tropical nights index per time period

The default output frequency is yearly. The date information of a timestep in *outfile* is the mid of the frequency interval.

Parameters

T

[FLOAT] Temperature threshold (unit: $^{\circ}\text{C}$; default: $T=20^{\circ}\text{C}$)

freq

[STRING] Output frequency (year, month)

Example

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

```
cdo eca_tr tnfile outfile
```

Author

Ralf Quast, Fabian Wachsmann

2.17.33 Eca_tx10p

Name

eca_tx10p - Very cold days percent wrt 10th percentile of reference period

Synopsis

cdo eca_tx10p *infile1 infile2 outfile*

Description

Let *infile1* be a time series of the daily maximum temperature TX, and *infile2* 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 centered 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 *outfile* is the date of the last contributing timestep in *infile1*.

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:

```
cdo eca_tx10p txfile txn10file outfile
```

Author

Ralf Quast

2.17.34 Eca_tx90p

Name

eca_tx90p - Very warm days percent wrt 90th percentile of reference period

Synopsis

```
cdo eca_tx90p infile1 infile2 outfile
```

Description

Let **infile1** be a time series of the daily maximum temperature TX, and **infile2** be the 90th percentile TXn90 of daily maximum temperatures for any period used as reference. Then the percentage of time where TX > TXn90. is calculated. TXn90 is calculated as the 90th percentile of daily maximum temperatures of a five day window centered on each calendar day of a given climate reference period. Note that both TX and TXn90 have to be given in the same units. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile1**.

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:

```
cdo eca_tx90p txfile txn90file outfile
```

Author

Ralf Quast

2.17.35 Eca_etccdi

Name

etccdi_tx90p, etccdi_tx10p, etccdi_tn90p, etccdi_tn10p, etccdi_r95p, etccdi_r99p - ETCCDI conform index for a reference period calculated with bootstrapping

Synopsis

cdo <operator>,n,startboot,endboot[,m] *infile1 infile2 infile3 outfile*

Description

This module enables to compute Climate Extremes Indices according to the method recommended by the Expert Team on Climate Change Detection and Indices. It differs from the corresponding eca_* indices by applying bootstrapping for a reference period (see Zhang et al. 2005) given by startboot and endboot and using the R-type 8 method for percentile calculation. A requirement for correct percentile calculation is that `CDO_PCTL_NBINS` $\geq$ `window*(endboot-startboot+1)*(sizeof(double)/sizeof(int))+2`. This demands for high working storage since the entire data of the bootstrapping interval need to be held in storage. Otherwise, a histogram is used to calculate the percentile. *infile2* (*infile3*) contains the daily minimum (maximum) of the bootstrapping interval. If `m=m`, the output variable will be saved monthly, otherwise with yearly frequency.

Operators

etccdi_tx90p

Percentage of Days when Daily Maximum Temperature is Above the 90th Percentile

etccdi_tx10p

Percentage of Days when Daily Maximum Temperature is Below the 10th Percentile

etccdi_tn90p

Percentage of Days when Daily Minimum Temperature is Above the 90th Percentile

etccdi_tn10p

Percentage of Days when Daily Minimum Temperature is Below the 10th Percentile

etccdi_r95p

Annual Total Precipitation when Daily Precipitation Exceeds the 95th Percentile of Wet Day Precipitation

etccdi_r99p

Annual Total Precipitation when Daily Precipitation Exceeds the 99th Percentile of Wet Day Precipitation

Parameters

n

[INTEGER] Window days, number of timesteps

startboot

[INTEGER] First year of bootstrapping interval

endboot

[INTEGER] Last year of bootstrapping interval

m

[CHARACTER] Output frequency

Environment

CDO_PCTL_NBINS sets the number of histogram bins (default: CDO_PCTL_NBINS=101).

Example

To compute the percentage of timesteps of each month with a daily maximum temperature greater than the 90th percentile of the daily maximum temperatures for a reference period 1960-1989 and a 5 consecutive days window use:

```
cdo etccdi_tx90p,5,1960,1989,m txfile -ydrunmin,5 txfile -ydrunmax,5 txfile outfile
```

Author

Fabian Wachsmann

2.18 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:

<i>Gradsdes</i>	<i>gradsdes</i>	GrADS data descriptor file
<i>Afterburner</i>	<i>after</i>	ECHAM standard post processor
<i>Filter</i>	<i>bandpass</i>	Bandpass filtering
	<i>lowpass</i>	Lowpass filtering
	<i>highpass</i>	Highpass filtering
<i>Gridcell</i>	<i>gridarea</i>	Grid cell area
	<i>gridweights</i>	Grid cell weights
<i>Smooth</i>	<i>smooth</i>	Smooth grid points
	<i>smooth9</i>	9 point smoothing
<i>Deltat</i>	<i>deltat</i>	Difference between timesteps
<i>Replacevalues</i>	<i>setvals</i>	Set list of old values to new values
	<i>setrtoc</i>	Set range to constant
	<i>setrtoc2</i>	Set range to constant others to constant2
<i>Getgridcell</i>	<i>gridcellindex</i>	Get grid cell index
<i>Vargen</i>	<i>const</i>	Create a constant field
	<i>random</i>	Create a field with random numbers
	<i>topo</i>	Create a field with topography
	<i>seq</i>	Create a time series
	<i>stdatm</i>	Create values for pressure and temperature for hydrostatic atmosphere
<i>Timsort</i>	<i>timsort</i>	Temporal sorting
<i>WindTrans</i>	<i>uvDestag</i>	Destaggering of u/v wind components
	<i>rotuvNorth</i>	Rotate u/v wind to North pole
	<i>projuvLatLon</i>	Cylindrical Equidistant projection
<i>Rotuv</i>	<i>rotuvb</i>	Backward wind rotation
<i>Mrotuvb</i>	<i>mrotuvb</i>	Backward rotation of MPIOM data
<i>Mastrfu</i>	<i>mastrfu</i>	Mass stream function
<i>Pressure</i>	<i>pressure_half</i>	Pressure on half-levels
	<i>pressure</i>	Pressure on full-levels
	<i>delta_pressure</i>	Pressure difference of half-levels
<i>Derivepar</i>	<i>sealevelpressure</i>	Sea level pressure
	<i>gheight</i>	Geopotential height on full-levels
	<i>gheight_half</i>	Geopotential height on half-levels
	<i>air_density</i>	Air density
<i>Adisit</i>	<i>adisit</i>	Potential temperature to in-situ temperature
	<i>adipot</i>	In-situ temperature to potential temperature
<i>Rhopot</i>	<i>rhopot</i>	Calculates potential density
<i>Histogram</i>	<i>histcount</i>	Histogram count
	<i>histsum</i>	Histogram sum
	<i>histmean</i>	Histogram mean
	<i>histfreq</i>	Histogram frequency
<i>Sethalo</i>	<i>sethalo</i>	Set the bounds of a field
<i>Wct</i>	<i>wct</i>	Windchill temperature
<i>Fdns</i>	<i>fdns</i>	Frost days where no snow index per time period

continues on next page

Table 15 – continued from previous page

<i>Strwin</i>	<i>strwin</i>	Strong wind days index per time period
<i>Strbre</i>	<i>strbre</i>	Strong breeze days index per time period
<i>Strgal</i>	<i>strgal</i>	Strong gale days index per time period
<i>Hurr</i>	<i>hurr</i>	Hurricane days index per time period
<i>CMORlite</i>	<i>cmorlite</i>	CMOR lite
<i>Verifygrid</i>	<i>verifygrid</i>	Verify grid coordinates
<i>Healpix</i>	<i>hpdegrade</i>	Degrade healpix
	<i>hpupgrade</i>	Upgrade healpix
<i>Symmetrize</i>	<i>symmetrize</i>	Mirrors data at the equator
<i>NCL_wind</i>	<i>uv2vr_cfd</i>	U and V wind to relative vorticity
	<i>uv2dv_cfd</i>	U and V wind to divergence

2.18.1 Gradsdes

Name

gradsdes - GrADS data descriptor file

Synopsis

cdo gradsdes[,mapversion] infile

Description

Creates a [GrADS] data descriptor file. Supported file formats are GRIB1, NetCDF, SERVICE, EXTRA and IEG. For GRIB1 files the GrADS map file is also generated. For SERVICE and EXTRA files the grid have to be specified with the **CDO** option '-g <grid>'. This module takes **infile** in order to create filenames for the descriptor (**infile.ct1**) and the map (**infile.gmp**) file.

Parameters

mapversion

[INTEGER] Format version of the GrADS map file for GRIB1 datasets. Use 1 for a machine specific version 1 GrADS map file, 2 for a machine independent version 2 GrADS map file and 4 to support GRIB files >2GB.

A version 2 map file can be used only with GrADS version 1.8 or newer.

A version 4 map file can be used only with GrADS version 2.0 or newer.

The default is 4 for files >2GB, otherwise 2.

Example

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

```
cdo gradsdes infile.grb
```

This will create a descriptor file with the name **infile.ct1** and the map file **infile.gmp**.

Assume the input GRIB1 dataset has 3 variables over 12 timesteps on a regular Gaussian F16 grid. The contents of the resulting GrADS data description file is approximately:

```
DSET ^infile.grb
DTYPE GRIB
INDEX ^infile.gmp
XDEF 64 LINEAR 0.000000 5.625000
YDEF 32 LEVELS -85.761 -80.269 -74.745 -69.213 -63.679 -58.143
          -52.607 -47.070 -41.532 -35.995 -30.458 -24.920
          -19.382 -13.844 -8.307 -2.769 2.769 8.307
          13.844 19.382 24.920 30.458 35.995 41.532
          47.070 52.607 58.143 63.679 69.213 74.745
          80.269 85.761
ZDEF 4 LEVELS 925 850 500 200
TDEF 12 LINEAR 12:00Z1jan1987 1mo
TITLE infile.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]
```

(continues on next page)

(continued from previous page)

```
tslm1    0  139,1,0  surface temperature of land  [K]  
ENDVARS
```

Author

Uwe Schulzweida

2.18.2 Afterburner

Name

after - ECHAM standard post processor

Synopsis

cdo after[,vct] *infile outfile*

Description

The *afterburner* is the standard post processor for [ECHAM] GRIB and NetCDF data which provides the following operations:

- Extract specified variables and levels
- Compute derived variables
- Transform spectral data to Gaussian grid representation
- Vertical interpolation to pressure levels
- Compute temporal means

This operator reads selection parameters as namelist from stdin. Use the UNIX redirection "<namelistfile" to read the namelist from file.

The input files can't be combined with other **CDO** operators because of an optimized reader for this operator.

Namelist

Namelist parameter and their defaults:

TYPE=0, CODE=-1, LEVEL=-1, INTERVAL=0, MEAN=0, EXTRAPOLATE=1

TYPE controls the transformation and vertical interpolation. Transforming spectral data to Gaussian grid representation and vertical interpolation to pressure levels are performed in a chain of steps. The TYPE parameter may be used to stop the chain at a certain step. Valid values are:

TYPE	=	0	:	Hybrid	level	spectral	coefficients
TYPE	=	10	:	Hybrid	level	fourier	coefficients
TYPE	=	11	:	Hybrid	level	zonal	mean sections
TYPE	=	20	:	Hybrid	level	gauss	grids
TYPE	=	30	:	Pressure	level	gauss	grids
TYPE	=	40	:	Pressure	level	fourier	coefficients
TYPE	=	41	:	Pressure	level	zonal	mean sections
TYPE	=	50	:	Pressure	level	spectral	coefficients
TYPE	=	60	:	Pressure	level	fourier	coefficients
TYPE	=	61	:	Pressure	level	zonal	mean sections
TYPE	=	70	:	Pressure	level	gauss	grids

Vorticity, divergence, streamfunction and velocity potential need special treatment in the vertical transformation. They are not available as types 30, 40 and 41. If you select one of these combinations, type is automatically switched to the equivalent types 70, 60 and 61. The type of all other variables will be switched too, because the type is a global parameter.

CODE selects the variables by the ECHAM GRIB1 code number (1-255). The default value -1 processes all detected codes. Derived variables computed by the afterburner:

Code	Name	Longname	Units	Level	Needed Codes
34	low_cld	low cloud		single	223 on modellevel
35	mid_cld	mid cloud		single	223 on modellevel
36	hih_cld	high cloud		single	223 on modellevel
131	u	u-velocity	m/s	atm (ml+pl)	138, 155
132	v	v-velocity	m/s	atm (ml+pl)	138, 155
135	omega	vertical velocity	Pa/s	atm (ml+pl)	138, 152, 155
148	stream	streamfunction	m ² /s	atm (ml+pl)	131, 132
149	velopot	velocity potential	m ² /s	atm (ml+pl)	131, 132
151	slp	mean sea level pressure	Pa	surface	129, 130, 152
156	geopoth	geopotential height	m	atm (ml+pl)	129, 130, 133, 152
157	rhumidity	relative humidity		atm (ml+pl)	130, 133, 152
189	scfcs	surface solar cloud forcing		surface	176-185
190	tcfcs	surface thermal cloud forcing		surface	177-186
191	scf0	top solar cloud forcing		surface	178-187
192	tcf0	top thermal cloud forcing		surface	179-188
259	windspeed	windspeed	m/s	atm (ml+pl)	sqrt(u*u+v*v)
260	precip	total precipitation		surface	142+143

LEVEL selects the hybrid or pressure levels. The allowed values depends on the parameter **TYPE**. The default value -1 processes all detected levels.

INTERVAL selects the processing interval. The default value 0 process data on monthly intervals. **INTERVAL=1** sets the interval to daily.

MEAN=1 compute and write monthly or daily mean fields. The default value 0 writes out all timesteps.

EXTRAPOLATE=0 switch off the extrapolation of missing values during the interpolation from model to pressure level (only available with **MEAN=0** and **TYPE=30**). The default value 1 extrapolate missing values.

Possible combinations of **TYPE**, **CODE** and **MEAN**:

TYPE	CODE		MEAN
0/10/11	130	temperature	0
0/10/11	131	u-velocity	0
0/10/11	132	v-velocity	0
0/10/11	133	specific humidity	0
0/10/11	138	vorticity	0
0/10/11	148	streamfunction	0
0/10/11	149	velocity potential	0
0/10/11	152	LnPs	0
0/10/11	155	divergence	0
>11	all	codes	0/1

Parameters

vct

[STRING] File with VCT in ASCII format

Example

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

```
cdo after infile outfile << EON
  TYPE=30 LEVEL=92500,85000,50000,20000
EON
```

2.18.3 Filter

Name

bandpass, lowpass, highpass - Time series filtering

Synopsis

cdo bandpass,*fmin,fmax infile outfile*

cdo lowpass,*fmin infile outfile*

cdo highpass,*fmax infile outfile*

Description

This module takes the time series for each gridpoint in **infile** and (fast Fourier) transforms it into the frequency domain. According to the particular operator and its parameters certain frequencies are filtered (set to zero) in the frequency domain and the spectrum is (inverse fast Fourier) transformed back into the time domain. To determine the frequency the time-axis of **infile** 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 **infile** 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 **outfile**. Hints for efficient usage:

- to get reliable results the time-series has to be detrended (cdo detrend)
- the lowest frequency greater zero that can be contained in infile is $1/(N*dT)$,
- the greatest frequency is $1/(2dT)$ (Nyquist frequency),

with N the number of timesteps and dT the time increment of **infile** in years.

Missing value support for operators in this module is not implemented, yet!

Operators

bandpass

Bandpass filtering

Bandpass filtering (pass for frequencies between **fmin** and **fmax**). Suppresses all variability outside the frequency range specified by [**fmin,fmax**].

highpass

Highpass filtering

Highpass filtering (pass for frequencies greater than **fmin**). Suppresses all variability with frequencies lower than **fmin**.

lowpass

Lowpass filtering

Lowpass filtering (pass for frequencies lower than **fmax**). Suppresses all variability with frequencies greater than **fmax**.

Parameters

fmin

[FLOAT] Minimum frequency per year that passes the filter.

fmax

[FLOAT] Maximum frequency per year that passes the filter.

Note

For better performance of these operators use the **CDO** configure option `--with-fftw3`.

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 infile outfile
```

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

```
cdo lowpass,1 -del29feb infile outfile
```

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 infile outfile
```

with $x \leq 0.5$ and $y \geq 0.5$.

Author

Cedrick Ansonge

2.18.4 Gridcell

Name

gridarea, gridweights - Grid cell quantities

Synopsis

cdo <operator>[,parameters] *infile outfile*

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 coordinates. The area of a grid cell is calculated using spherical triangles from the coordinates of the center and the vertices. The base is a unit sphere which is scaled with the radius of the planet. The default planet radius is 6371000 meter. The parameter **radius** or the environment variable [*PLANET_RADIUS*](#) can be used to change the default. 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 planet radius to square meters.

gridweights

Grid cell weights

Writes the grid cell area weights to the output stream.

Parameters

radius

[FLOAT] Planet radius in meter

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.

Author

Uwe Schulzweida

2.18.5 Smooth

Name

smooth, smooth9 - Smooth grid points

Synopsis

cdo smooth[,parameters] *infile outfile*

cdo smooth9 *infile outfile*

Description

Smooth all grid points of a horizontal grid.

Operators

smooth

Smooth grid points

Performs a N point smoothing on all input fields. The number of points used depend on the search radius (radius) and the maximum number of points (maxpoints). Per default all points within the search radius of 1degree are used. The weights for the points depend on the weighting method and the distance. The implemented weighting method is linear with constant default weights of 0.25 at distance 0 (weight0) and at the search radius (weightR).

smooth9

9 point smoothing

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.

Parameters

nsmooth

[INTEGER] Number of times to smooth, default nsmooth=1

radius

[STRING] Search radius, default radius=1deg (units: deg, rad, km, m)

maxpoints

[INTEGER] Maximum number of points, default maxpoints=<gridsize>

weighted

[STRING] Weighting method, default weighted=linear

weight0

[FLOAT] Weight at distance 0, default weight0=0.25

weightR

[FLOAT] Weight at the search radius, default weightR=0.25

Author

Uwe Schulzweida, Cedrick Ansorge

2.18.6 Deltat

Name

deltat - Difference between timesteps

Synopsis

cdo deltat *infile outfile*

Description

This operator computes the difference between each timestep.

Author

Uwe Schulzweida

2.18.7 Replacevalues

Name

setvals, setrtoc, setrtoc2 - Replace data values

Synopsis

cdo setvals,*oldval,newval infile outfile*

cdo setrtoc,*rmin,rmax,c infile outfile*

cdo setrtoc2,*rmin,rmax,c,c2 infile outfile*

Description

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

Operators

setvals

Set list of old values to new values

Supply a list of n pairs of old and new values.

setrtoc

Set range to constant

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

setrtoc2

Set range to constant others to constant2

$$o(t, x) = \begin{cases} \mathbf{c} & \text{if } i(t, x) \geq \mathbf{rmin} \wedge i(t, x) \leq \mathbf{rmax} \\ \mathbf{c2} & \text{if } i(t, x) < \mathbf{rmin} \vee i(t, x) > \mathbf{rmax} \end{cases}$$

Parameters

oldval,newval,...

[FLOAT] Pairs of old and new values

rmin

[FLOAT] Lower bound

rmax

[FLOAT] Upper bound

c

[FLOAT] New value - inside range

c2

[FLOAT] New value - outside range

Author

Etienne Tourigny

2.18.8 Getgridcell

Name

gridcellindex - Get grid cell index

Synopsis

cdo gridcellindex,*parameters infile*

Description

Get the grid cell index of one grid point selected by the parameter lon and lat. The indices range from 1 to N.

Parameters

lon

[FLOAT] Longitude of the grid cell in degree

lat

[FLOAT] Latitude of the grid cell in degree

Example

The grid cell index of a data set on an F80 regular Gaussian grid at lon=10/lat=53.5 is 10250:

```
cdo gridcellindex,lon=10,lat=53.5 F80data
```

Author

Uwe Schulzweida

2.18.9 Vargen

Name

const, random, topo, seq, stdatm - Generate a field

Synopsis

cdo const,*const,grid outfile*

cdo random,*grid[,seed] outfile*

cdo topo,*grid outfile*

cdo seq,*start,end[,inc] outfile*

cdo stdatm,*levels outfile*

Description

Generates a dataset with one or more fields.

Operators

const

Create a constant field

Creates a constant field. All field elements of the grid have the same value.

random

Create a field with random numbers

Creates a field with rectangularly distributed random numbers in the interval [0,1].

topo

Create a field with topography

Creates a field with topography data, per default on a global half degree grid.

seq

Create a time series

Creates a time series with field size 1 and field elements beginning with a start value in time step 1 which is increased from one time step to the next.

stdatm

Create values for pressure and temperature for hydrostatic atmosphere

Creates pressure and temperature values for the given list of vertical levels. The formulas are:

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

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

with the following constants

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

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

$P_0 = 1013.25\text{hPa}$: surface pressure

$H = 10000.0\text{m}$: scale height

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

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

This is the solution for the hydrostatic equations and is only valid for the troposphere (constant positive lapse rate). The temperature increase in the stratosphere and other effects of the upper atmosphere are not taken into account.

Parameters

const

[FLOAT] Constant

seed

[INTEGER] The seed for a new sequence of pseudo-random numbers [default: 1]

grid

[STRING] Target grid description file or name

start

[FLOAT] Start value of the loop

end

[FLOAT] End value of the loop

inc

[FLOAT] Increment of the loop [default: 1]

levels

[FLOAT] Target levels in metre above surface

Example

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

```
cdo enlarge,gridfile -stdatm,10000,8000,5000,3000,2000,1000,500,200,0 outfile
```

Author

Uwe Schulzweida, Ralf Müller

2.18.10 Timsort

Name

timsort - Temporal sorting

Synopsis

cdo timsort *infile outfile*

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 infile outfile
```

Author

Uwe Schulzweida

2.18.11 WindTrans

Name

uvDestag, rotuvNorth, projuvLatLon - Wind transformation

Synopsis

cdo uvDestag,u,v[, -/+0.5, -/+0.5] *infile outfile*

cdo rotuvNorth,u,v *infile outfile*

cdo projuvLatLon,u,v *infile outfile*

Description

This module contains special operators for datasets with wind components on a rotated lon/lat grid, e.g. data from the regional model HIRLAM or [\[REMO\]](#).

Operators

uvDestag

Destaggering of u/v wind components

This is a special operator for destaggering of wind components. If the file contains a grid with temperature (name='t' or code=11) then grid_temp will be used for destaggered wind.

rotuvNorth

Rotate u/v wind to North pole

This is an operator for transformation of wind-vectors from grid-relative to north-pole relative for the whole file. (FAST implementation with JACOBIANS)

projuvLatLon

Cylindrical Equidistant projection

This is an operator for transformation of wind-vectors from the globe-spherical coordinate system into a flat Cylindrical Equidistant (lat-lon) projection. (FAST JACOBIAN implementation)

Parameters

u,v

[STRING] Pair of u,v wind components (use variable names or code numbers)

-/+0.5,-/+0.5

[STRING] Destaggered grid offsets are optional (default -0.5,-0.5)

Example

Typical operator sequence on HIRLAM NWP model output (LAMH_D11 files):

```
cdo uvDestag,33,34  inputfile inputfile_destag
cdo rotuvNorth,33,34 inputfile_destag inputfile_rotuvN
```

Author

Michal Koutek

2.18.12 Rotuv

Name

rotuvb - Backward wind rotation

Synopsis

cdo rotuvb,u,v[, ...] *infile outfile*

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.

Parameters

u,v

[STRING] Pairs of zonal and meridional velocity components (use variable names or code numbers)

Note

This is a specific implementation for data from the REMO model, it may not work with data from other sources.

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 infile outfile
```

2.18.13 Mrotuvb

Name

mrotuvb - Backward rotation of MPIOM data

Synopsis

cdo mrotuvb *infile1 infile2 outfile*

Description

MPIOM data are on a rotated Arakawa C grid. The velocity components U and V are located on the edges of the cells and point in the direction of the grid lines and rows. With mrotuvb the velocity vector is rotated in latitudinal and longitudinal direction. Before the rotation, U and V are interpolated to the scalar points (cell center). U is located with the coordinates for U in infile1 and V in infile2. mrotuvb assumes a positive meridional flow for a flow from grid point(i,j) to grid point(i,j+1) and positive zonal flow for a flow from grid point(i+1,j) to point(i,j).

Note

This is a specific implementation for data from the MPIOM model, it may not work with data from other sources.

Author

Uwe Schulzweida

2.18.14 Mastrfu

Name

mastrfu - Mass stream function

Synopsis

cdo *mastrfu infile outfile*

Description

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

Example

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

```
cdo mastrfu infile outfile
```

2.18.15 Pressure

Name

pressure_half, pressure, delta_pressure - Pressure on model levels

Synopsis

cdo <operator> *infile outfile*

Description

This module contains operators to calculate the pressure on model levels. To calculate the pressure on model levels, the a and b coefficients defining the model levels and the surface pressure are required. The a and b coefficients are normally part of the model level data. If not available, the surface pressure can be derived from the logarithm of the surface pressure. The surface pressure is identified by the GRIB1 code number or NetCDF CF standard name.

Name	Units	GRIB1 code	CF standard name
log surface pressure	Pa	152	
surface pressure	Pa	134	surface_air_pressure

Operators

pressure_half

Pressure on half-levels

This operator computes the pressure on model half-levels in pascal. The model half-level pressure (p_{half}) is given by:

$$p_{half} = a + b * sp$$

with:

a, b : coefficients defining the model levels

sp : surface pressure

pressure

Pressure on full-levels

This operator computes the pressure on model full-levels in pascal. The pressure on model full-levels (p_{full}) is in the middle of the layers defined by the model half-levels:

$$p_{full} = (p_{half_above} + p_{half_below})/2$$

delta_pressure

Pressure difference of half-levels

This operator computes the pressure difference between to model half-levels.

$$delta_p = p_{half_below} - p_{half_above}$$

Author

Uwe Schulzweida

2.18.16 Derivepar

Name

sealevelpressure, gheight, gheight_half, air_density - Derived model parameters

Synopsis

cdo <operator> *infile outfile*

Description

This module contains operators that calculate derived model parameters. All necessary input variables are identified by their GRIB1 code number or the NetCDF CF standard name. Supported GRIB1 parameter tables are: WMO standard table number 2 and ECMWF local table number 128.

CF standard name	Units	GRIB 1 code
surface_air_pressure	Pa	134
air_temperature	K	130
specific_humidity	kg/kg	133
surface_geopotential	m ² s ⁻²	129
geopotential_height	m	156

Operators

sealevelpressure

Sea level pressure

This operator computes the sea level pressure (air_pressure_at_sea_level). Required input fields are surface_air_pressure, surface_geopotential and air_temperature on full hybrid sigma pressure levels.

gheight

Geopotential height on full-levels

This operator computes the geopotential height (geopotential_height) on model full-levels in metres. Required input fields are surface_air_pressure, surface_geopotential, specific_humidity and air_temperature on full hybrid sigma pressure levels. Note, this procedure is an approximation, which doesn't take into account for the effects of e.g. cloud ice and water, rain and snow.

gheight_half

Geopotential height on half-levels

This operator computes the geopotential height (geopotential_height) on model half-levels in metres. Required input fields are surface_air_pressure, surface_geopotential, specific_humidity and air_temperature on full hybrid sigma pressure levels. Note, this procedure is an approximation, which doesn't take into account for the effects of e.g. cloud ice and water, rain and snow.

air_density

Air density

This operator computes the air density, it depends on pressure, humidity and temperature. Required input fields are surface_air_pressure, specific_humidity and air_temperature on full hybrid sigma pressure levels. The air density (ρ) is calculated with the following formula:

$$\rho = P / (R_s * T_v)$$

P : air pressure in Pascal

T_v : virtual temperature in Kelvin

R_s : specific gas constant for dry air; 287.085 J/(kg*K)

$$Tv = T * [1 + a * q]$$

T : air temperature in Kelvin

q : specific humidity

a : gas constants of air and water vapor; 0.6078

Author

Uwe Schulzweida

2.18.17 Adisit

Name

adisit, adipot - Potential temperature to in-situ temperature and vice versa

Synopsis

cdo <operator>[,pressure] *infile outfile*

Operators

adisit

Potential temperature to in-situ temperature

This is a special operator for the post processing of the ocean and sea ice model [MPIOM]. It converts potential temperature adiabatically to in-situ temperature to(t, s, p). Required input fields are sea water potential temperature (name=tho; code=2) and sea water salinity (name=sao; code=5). Pressure is calculated from the level information or can be specified by the optional parameter. Output fields are sea water temperature (name=to; code=20) and sea water salinity (name=s; code=5).

adipot

In-situ temperature to potential temperature

This is a special operator for the post processing of the ocean and sea ice model [MPIOM]. It converts in-situ temperature to potential temperature tho(to, s, p). Required input fields are sea water in-situ temperature (name=t; code=2) and sea water salinity (name=sao,s; code=5). Pressure is calculated from the level information or can be specified by the optional parameter. Output fields are sea water temperature (name=tho; code=2) and sea water salinity (name=s; code=5).

Parameters

pressure

[FLOAT] Pressure in bar (constant value assigned to all levels)

Author

Uwe Schulzweida, Helmut Haak

2.18.18 Rhopot

Name

rhopot - Calculates potential density

Synopsis

cdo rhopot[*,pressure*] *infile outfile*

Description

This is a special operator for the post processing of the ocean and sea ice model [MPIOM]. It calculates the sea water potential density (name=rhopoto; code=18). Required input fields are sea water in-situ temperature (name=to; code=20) and sea water salinity (name=sao; code=5). Pressure is calculated from the level information or can be specified by the optional parameter.

Parameters

pressure

[FLOAT] Pressure in bar (constant value assigned to all levels)

Example

To compute the sea water potential density from the potential temperature use this operator in combination with *adisit*:

```
cdo rhopot -adisit infile outfile
```

2.18.19 Histogram

Name

histcount, histsum, histmean, histfreq - Histogram

Synopsis

cdo <operator>, *bounds infile outfile*

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.

Parameters

bounds

[FLOAT] Comma-separated list of the bin bounds (-inf and inf valid)

Author

Ralf Quast

2.18.20 Sethalo

Name

sethalo - Set the bounds of a field

Synopsis

cdo sethalo,*parameter infile outfile*

Description

This operator sets the boundary in the east, west, south and north of the rectangular understood fields. Positive values of the parameters increase the boundary in the selected direction. Negative values decrease the field at the selected boundary. The new rows and columns are filled with the missing value. With the optional parameter value a different fill value can be used. Global cyclic fields are filled cyclically at the east and west borders, if the fill value is not set by the user. All input fields need to have the same horizontal grid.

Parameters

east

[INTEGER] East halo

west

[INTEGER] West halo

south

[INTEGER] South halo

north

[INTEGER] North halo

value

[FLOAT] Fill value (default is the missing value)

Author

Uwe Schulzweida

2.18.21 Wct

Name

wct - Windchill temperature

Synopsis

cdo wct *infile1 infile2 outfile*

Description

Let *infile1* and *infile2* be time series of temperature and wind speed fields, then a corresponding time series of resulting windchill temperatures is written to *outfile*. The wind chill temperature calculation is only valid for a temperature of $T \leq 33^{\circ}\text{C}$ and a wind speed of $v \geq 1.39 \text{ m/s}$. Whenever these conditions are not satisfied, a missing value is written to *outfile*. Note that temperature and wind speed fields have to be given in units of $^{\circ}\text{C}$ and m/s , respectively.

Author

Ralf Quast

2.18.22 Fdns

Name

fdns - Frost days where no snow index per time period

Synopsis

cdo fdns *infile1 infile2 outfile*

Description

Let **infile1** be a time series of the daily minimum temperature TN and **infile2** be a corresponding series of daily surface snow amounts. Then the number of days where $TN < 0^{\circ}\text{C}$ and the surface snow amount is less than 1 cm is counted. The temperature TN have to be given in units of Kelvin. The date information of a timestep in **outfile** is the date of the last contributing timestep in **infile**.

2.18.23 Strwin

Name

strwin - Strong wind days index per time period

Synopsis

cdo strwin[,v] *infile outfile*

Description

Let *infile* be a time series of the daily maximum horizontal wind speed VX , then the number of days where $VX > v$ is counted. The horizontal wind speed v is an optional parameter with default $v = 10.5$ m/s. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to v . Note that both VX and v have to be given in units of m/s. Also note that the horizontal wind speed is defined as the square root of the sum of squares of the zonal and meridional wind speeds. The date information of a timestep in *outfile* is the date of the last contributing timestep in *infile*.

Parameters

v

[FLOAT] Horizontal wind speed threshold (m/s, default $v = 10.5$ m/s)

2.18.24 Strbre**Name**

strbre - Strong breeze days index per time period

Synopsis

cdo strbre[,v] *infile outfile*

Description

Let *infile* 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 *outfile* is the date of the last contributing timestep in *infile*.

2.18.25 Strgal

Name

strgal - Strong gale days index per time period

Synopsis

cdo strgal[,v] *infile outfile*

Description

Let *infile* 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 *outfile* is the date of the last contributing timestep in *infile*.

2.18.26 Hurr**Name**

hurr - Hurricane days index per time period

Synopsis

cdo hurr *infile outfile*

Description

Let *infile* 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 *outfile* is the date of the last contributing timestep in *infile*.

2.18.27 CMORlite

Name

cmorlite - CMOR lite

Synopsis

cdo cmorlite,table[,convert] *infile outfile*

Description

The *CMOR* (Climate Model Output Rewriter) library comprises a set of functions, that can be used to produce CF-compliant NetCDF files that fulfill the requirements of many of the climate community's standard model experiments. These experiments are collectively referred to as MIP's. Much of the metadata written to the output files is defined in MIP-specific tables, typically made available from each MIP's web site.

The **CDO** operator cmorlite process the header and variable section of such MIP tables and writes the result with the internal IO library [CDI]. In addition to the CMOR 2 and 3 table format, the **CDO** parameter table format is also supported. The following parameter table entries are available:

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

Most of the above entries are stored as variables attributes, some of them are handled differently. The variable name is used as a search key for the parameter table. `valid_min`, `valid_max`, `ok_min_mean_abs` and `ok_max_mean_abs` are used to check the range of the data.

Parameters

table

[STRING] Name of the CMOR table as specified from PCMDI

convert

[STRING] Converts the units if necessary

Example

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

```
prompt> cat mypartab
&parameter
```

(continues on next page)

(continued from previous page)

```
name           = t
out_name        = ta
standard_name   = air_temperature
units           = "K"
missing_value   = 1.0e+20
valid_min       = 157.1
valid_max       = 336.3
/
```

To apply this parameter table to a dataset use:

```
cdo -f nc cmorlite,mypartab,convert infile outfile
```

This command renames the variable `t` to `ta`. The standard name of this variable is set to `air_temperature` and the unit is set to `[K]` (converts the unit if necessary). The missing value will be set to `1.0e+20`. In addition it will be checked whether the values of the variable are in the range of `157.1` to `336.3`. The result will be stored in NetCDF.

Author

Uwe Schulzweida

2.18.28 Verifygrid

Name

verifygrid - Verify grid coordinates

Synopsis

cdo verifygrid *infile*

Description

This operator verifies the coordinates of all horizontal grids found in *infile*. Among other things, it searches for duplicate cells, non-convex cells, and whether the center is located outside the cell bounds. Use the **CDO** option **-v** to output the position of these cells. This information can be useful to avoid problems when interpolating the data.

2.18.29 Healpix

Name

hpdegrade, hpupgrade - Change healpix resolution

Synopsis

cdo [options] <operator> ,parameters infile outfile

Description

Degrade or upgrade the resolution of a healpix grid.

Operators

hpdegrade

Degrade healpix

Degrade the resolution of a healpix grid. The value of the target pixel is the mean of the source pixels.

hpupgrade

Upgrade healpix

Upgrade the resolution of a healpix grid. The values of the target pixels are the value of the source pixel.

Parameters

nside

[INTEGER] The nside of the target healpix, must be a power of two [default: same as input].

zoom

[INTEGER] **zoom** is the refinement level and the relation to **nside** is: $nside = 2^{zoom}$.

order

[STRING] Pixel ordering of the target healpix ('nested' or 'ring').

power

[FLOAT] If non-zero, divide the result by $(nside[in]/nside[out])**power$. $power=-2$ keeps the sum of the map invariant.

Options

-p, **--async_read true** to read input data asynchronously.

Author

Uwe Schulzweida

2.18.30 Symmetrize

Name

symmetrize - Mirrors data at the equator

Synopsis

cdo symmetrize[,*parameters*] *infile outfile*

Description

This operator symmetrizes global fields relative to the equator. By default, data with positive latitudes are mirrored. With the parameter **lat**=negative, it is the data with negative latitudes. The result for fields on a global lon/lat or Gaussian grid is perfectly symmetrical. For fields on an unstructured grid, the result is the nearest neighbour of the other hemisphere. Use the **grid** parameter to specify the path to a grid description file if the unstructured data is available without grid coordinates.

Parameters

lat

[STRING] lat=negative mirrors data with negative latitudes

grid

[STRING] Grid description file or name

Author

Uwe Schulzweida, Bjorn Stevens

2.18.31 NCL_wind

Name

uv2vr_cfd, uv2dv_cfd - Wind transformation

Synopsis

cdo <operator>[,u,v,boundOpt,outMode] *infile outfile*

Description

This module contains **CDO** operators with an interface to NCL functions. The corresponding NCL functions have the same name. A more detailed description of those NCL function can be found on the NCL homepage <https://www.ncl.ucar.edu>.

Operators

uv2vr_cfd

U and V wind to relative vorticity

Computes relative vorticity for a latitude-longitude grid using centered finite differences. The grid need not be global and missing values are allowed.

uv2dv_cfd

U and V wind to divergence

Computes divergence for a latitude-longitude grid using centered finite differences. The grid need not be global and missing values are allowed.

Parameters

u STRING

Name of variable u (default: u)

v STRING

Name of variable v (default: v)

boundOpt INTEGER

Boundary condition option (0-3) (default: 0/1 for cyclic grids)

outMode STRING

Output mode new/append (default: new)

Author

Uwe Schulzweida

3 Contributors

3.1 History

CDO was originally developed by Uwe Schulzweida at the Max Planck Institute for Meteorology (MPI-M). The first public release is available since 2003. The MPI-M, together with the DKRZ, has a long history in the development of tools for processing climate data. **CDO** was inspired by some of these tools, such as the [\[PINGO\]](#) package and the GRIB-Modules.

PINGO (Procedural Interface for GRIB formatted Objects) was developed by Jürgen Waszkewitz, Peter Lenzen, and Nathan Gillet in 1995 at the DKRZ, Hamburg (Germany). **CDO** has a similar user interface and uses some of the PINGO operators.

The GRIB-Modules were developed by Heiko Borgert and Wolfgang Welke in 1991 at the MPI-M. **CDO** is using a similar module structure and also some of the operators.

3.2 External sources

CDO has incorporated code from several sources:

afterburner

is a postprocessing application for ECHAM data and ECMWF analysis data, originally developed by Edilbert Kirk, Michael Ponater and Arno Hellbach. The afterburner code was modified for the **CDO** operators [after](#), [ml2pl](#), [sp2gp](#), [gp2sp](#).

SCRIP

is a software package used to generate interpolation weights for remapping fields from one grid to another in spherical geometry [\[SCRIP\]](#). It was developed at the Los Alamos National Laboratory by Philip W. Jones. The SCRIP library was converted from Fortran to ANSI C and is used as the base for the remapping operators in **CDO**.

YAC

(Yet Another Coupler) was jointly developed by DKRZ and MPI-M by Moritz Hanke and René Redler [\[YAC\]](#). **CDO** is using the clipping and cell search routines for the conservative remapping with [remapcon](#).

libkdtree

a C99 implementation of the kd-tree algorithm developed by Jörg Dietrich.

CDO uses tools from the GNU project, including automake, and libtool.

3.3 Contributors

The primary contributors to the **CDO** development have been:

Uwe Schulzweida : Concept, design and implementation of **CDO**, project coordination, and releases.

Luis Kornbluh : He has supported **CDO** from the beginning.

His main contributions are GRIB performance and compression, GME and unstructured grid support. Luis also helps with design and planning.

Ralf Müller : He is working on **CDO** since 2009.

His main contributions are the implementation of the User Portal, the Ruby and Python interface for all **CDO** operators, the building process and the Windows support. The **CDO** User Portal was funded by the European Commission infrastructure project IS-ENES. Ralf also helps a lot with the user support.

Implemented operators: [intlevel3d](#), [consecsum](#), [consects](#), [ngrid](#), [ngridpoints](#), [reducegrid](#)

Oliver Heidmann : He is working on **CDO** since 2015.

His main contributions are refactoring to C++ and the new command line parser.

Karin Meier-Fleischer : She is working in the **CDO** user support since 2017.

Fabian Wachsmann : He is working on **CDO** for the CMIP6 project since 2016.

His main task is the implementation and support of the cmor operator.

He has also implemented the ETCCDI Indices of Daily Temperature and Precipitation Extremes.

Cedrick Ansonge : He worked on the software package **CDO** from 2007-2011.

Implemented operators: *eof*, *eof3d*, *enscrps*, *ensbrs*, *maskregion*, *bandpass*, *lowpass*, *highpass*, *smooth9*

Ralf Quast : He worked on **CDO** on behalf of the Service Gruppe Anpassung (SGA), DKRZ in 2006.

He implemented all ECA Indices of Daily Temperature and Precipitation Extremes,

all percentile operators, module *Ydrunstat* and *wct*.

Kameswarrao Modali : He worked on **CDO** from 2012-2013.

Implemented operators: contour, shaded, grfill, vector, graph.

Michal Koutek : Implemented operators: *selmulti*, *delmulti*, *changemulti*, *samplegrid*,

uvDestag, *rotuvNorth*, *projuvLatLon*.

Etienne Tourigny : Implemented operators: *setclonlatbox*, *setindexbox*,

splitsel, *histfreq*, *setvals*, *setrtoc*, *setrtoc2*.

Karl-Hermann Wieners : Implemented operators: *aexpr*, *aexprf*, *selzaxisname*.

Asela Rajapakse : He worked on **CDO** from 2016-2017 as part of the EUDAT project.

Implemented operator: *verifygrid*

Estanislao Gavilan : Improved the **CDO** documentation for the installation section.

Many users have contributed to **CDO** by sending bug reports, patches and suggestions over time. Very helpful is also the active participation in the user forum of some users. Here is an incomplete list:

Jaison-Thomas Ambadan, Harald Anlauf, Andy Aschwanden, Stefan Bauer, Simon Blessing, Renate Brokopf, Michael Boettinger, Tim Brücher, Reinhard Budich, Martin Claus, Traute Crüger, Brendan de Tracey, Irene Fischer-Bruns, Chris Fletscher, Helmut Frank, Kristina Fröhlich, Oliver Fuhrer, Monika Esch, Pier Giuseppe Fogli, Beate Gayer, Veronika Gayler, Marco Giorgetta, David Gobbett, Holger Goettel, Helmut Haak, Stefan Hagemann, Angelika Heil, Barbara Hennemuth, Daniel Hernandez, Nathanael Huebbe, Thomas Jahns, Frank Kaspar, Daniel Klocke, Edi Kirk, Stefanie Legutke, Leonidas Linardakis, Stephan Lorenz, Frank Lunkeit, Uwe Mikolajewicz, Laura Niederdrenk, Dirk Notz, Hans-Jürgen Panitz, Ronny Petrik, Swantje Preuschmann, Florian Prill, Asela Rajapakse, Daniel Reinert, Hannes Reuter, Mathis Rosenhauer, Reiner Schnur, Martin Schultz, Dennis Shea, Kevin Sieck, Martin Stendel, Bjorn Stevens, Martina Stockhaus, Claas Teichmann, Adrian Tompkins, Jörg Trentmann, Álvaro M. Valdebenito, Geert Jan van Oldenborgh, Jin-Song von Storch, David Wang, Joerg Wegner, Heiner Widmann, Claudia Wunram, Klaus Wyser

Please let me know if your name was omitted!

4 Environment Variables

The following environment variables affect the behavior of **CDO**:

CDO_DOWNLOAD_PATH

Path where **CDO** can store downloads.

CDO_FILE_SUFFIX

Sets the filename suffix. This suffix will be added to the output file name instead of the filename extension derived from the file format. **CDO_FILE_SUFFIX=NULL** will disable the adding of a file suffix. A filename suffix is used in **CDO** operators which generate file names (e.g. *Split*).

CDO_GRIDSEARCH_RADIUS

Sets the grid search radius in degrees (default: **CDO_GRIDSEARCH_RADIUS=180**). Used by the operators *setmisstonn* and *remapknn*.

CDO_HISTORY_INFO

'false' don't write information to the global *history* attribute (default: **CDO_HISTORY_INFO=true**).

CDO_ICON_GRIDS

Root directory of the installed ICON grids (e.g. */pool/data/ICON*).

CDO_PCTL_NBINS

Sets the number of histogram bins (default: **CDO_PCTL_NBINS=101**). Histograms are used to calculate the percentile over time.

CDO_REMAP_NORM

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

CDO_RESET_HISTORY

'true' resets the global *history* attribute (default: **CDO_RESET_HISTORY=false**).

CDO_VERSION_INFO

'false' disables the global NetCDF attribute **CDO** (default: **CDO_VERSION_INFO=true**).

REMAP_AREA_MIN

This variable is used to set the minimum destination area fraction (default: **REMAP_AREA_MIN=0.0**). Used by the operators *remapcon* and *remaplaf*.

REMAP_EXTRAPOLATE

This variable is used to switch the extrapolation feature 'on' or 'off'. Extrapolation is used in remapping.

5 Parallelized operators

Some of the **CDO** operators are parallelized with OpenMP. To use **CDO** with multiple OpenMP threads, you have to set the number of threads with the option '-P'. Here is an example to distribute the bilinear interpolation on 8 OpenMP threads:

```
cdo -P 8 remapbil,targetgrid infile outfile
```

The following **CDO** operators are parallelized with OpenMP:

Module	Operator	Description
Afterburner	after	ECHAM standard post processor
Detrend	detrend	Detrend
EcaEtccdi	etccdi_tx90p	% of days when daily max temperature is > the 90th percentile
EcaEtccdi	etccdi_tx10p	% of days when daily max temperature is < the 10th percentile
EcaEtccdi	etccdi_tn90p	% of days when daily min temperature is > the 90th percentile
EcaEtccdi	etccdi_tn10p	% of days when daily min temperature is < the 10th percentile
EcaEtccdi	etccdi_r95p	Annual tot precip when daily precip exceeds the 95th percentile of Wet Day Precipitation
EcaEtccdi	etccdi_r99p	Annual tot precip when daily precip exceeds the 99th percentile of Wet Day Precipitation
Ensstat	ens<STAT>	Statistical values over an ensemble
EOF	eof	Empirical Orthogonal Functions
Fillmiss	setmisstonn	Set missing value to nearest neighbor
Fillmiss	setmisstodis	Set missing value to distance-weighted average
Filter	bandpass	Bandpass filtering
Filter	lowpass	Lowpass filtering
Filter	highpass	Highpass filtering
Fourier	fourier	Fourier transformation
Genweights	genbil	Generate bilinear interpolation weights
Genweights	genbic	Generate bicubic interpolation weights
Genweights	gendis	Generate distance-weighted average remap weights
Genweights	gennn	Generate nearest neighbor remap weights
Genweights	gencon	Generate 1st order conservative remap weights
Genweights	genlaf	Generate largest area fraction remap weights
Gridboxstat	gridbox<STAT>	Statistical values over grid boxes
Intlevel	intlevel	Linear level interpolation
Intlevel3d	intlevel3d	Linear level interpolation from/to 3D vertical coordinates
Remapeta	remapeta	Remap vertical hybrid level
Remap	remapbil	Bilinear interpolation
Remap	remapbic	Bicubic interpolation

continues on next page

Table 16 – continued from previous page

Module	Operator	Description
Remap	remapdis	Distance-weighted average remapping
Remap	remapnn	Nearest neighbor remapping
Remap	remapcon	First order conservative remapping
Remap	remaplaf	Largest area fraction remapping
Smooth	smooth	Smooth grid points
Spectral	sp2gp, gp2sp	Spectral transformation
Vertintap	ap2pl, ap2hl	Vertical interpolation on hybrid sigma height coordinates
Vertintgh	gh2hl	Vertical height interpolation
Vertintml	ml2pl, ml2hl	Vertical interpolation on hybrid sigma pressure coordinates

6 Standard name table

The following CF standard names are supported by **CDO**.

CF standard name	Units	GRIB 1 code	variable name
surface_geopotential	m2 s-2	129	geosp
air_temperature	K	130	ta
specific_humidity	1	133	hus
surface_air_pressure	Pa	134	aps
air_pressure_at_sea_level	Pa	151	psl
geopotential_height	m	156	zg

7 Grid description examples

7.1 Example of a curvilinear grid description

Here is an example for the **CDO** description of a curvilinear grid. `xvals/yvals` describe the positions of the 6x5 quadrilateral grid cells. The first 4 values of `xbounds/ybounds` are the corners of the first grid cell.

```
gridtype = curvilinear
gridsize = 30
xsize    = 6
ysize    = 5
xvals    = -21 -11  0  11  21  30 -25 -13  0  13
           25  36 -31 -16  0  16  31  43 -38 -21
           0  21  38  52 -51 -30  0  30  51  64
xbounds  = -23 -14 -17 -28 -14 -5 -6 -17 -5 5 6 -6
           5  14  17  6  14  23  28  17  23  32  38  28
           -28 -17 -21 -34 -17 -6 -7 -21 -6 6 7 -7
           6  17  21  7  17  28  34  21  28  38  44  34
           -34 -21 -27 -41 -21 -7 -9 -27 -7 7 9 -9
           7  21  27  9  21  34  41  27  34  44  52  41
           -41 -27 -35 -51 -27 -9 -13 -35 -9 9 13 -13
           9  27  35  13  27  41  51  35  41  52  63  51
           -51 -35 -51 -67 -35 -13 -21 -51 -13 13 21 -21
           13  35  51  21  35  51  67  51  51  63  77  67
yvals    = 29  32  32  32  29  26  39  42  42  42
           39  35  48  51  52  51  48  43  57  61
           62  61  57  51  65  70  72  70  65  58
ybounds  = 23  26  36  32  26  27  37  36  27  27  37  37
           27  26  36  37  26  23  32  36  23  19  28  32
           32  36  45  41  36  37  47  45  37  37  47  47
           37  36  45  47  36  32  41  45  32  28  36  41
           41  45  55  50  45  47  57  55  47  47  57  57
           47  45  55  57  45  41  50  55  41  36  44  50
           50  55  64  58  55  57  67  64  57  57  67  67
           57  55  64  67  55  50  58  64  50  44  51  58
           58  64  72  64  64  67  77  72  67  67  77  77
           67  64  72  77  64  58  64  72  58  51  56  64
```

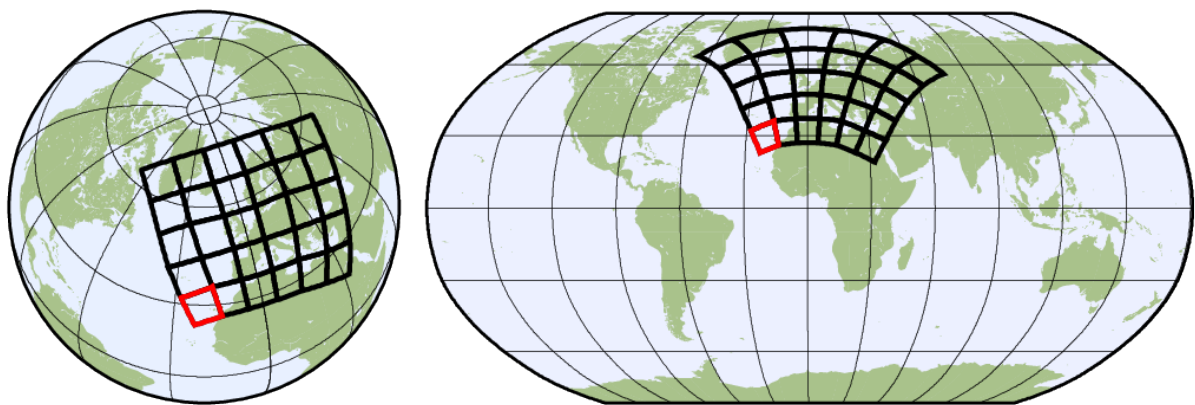


Fig. 1: Orthographic and Robinson projection of the curvilinear grid, the first grid cell is colored red

7.2 Example description for an unstructured grid

Here is an example of the **CDO** description for an unstructured grid. `xvals/yvals` describe the positions of 30 independent hexagonal grid cells. The first 6 values of `xbounds/ybounds` are the corners of the first grid cell. The grid cell corners have to rotate counterclockwise. The first grid cell is colored red.

```

gridtype = unstructured
gridsize = 30
nvertex = 6
xvals    = -36  36   0  -18  18  108  72  54  90  180  144  126  162 -108 -144
           -162 -126 -72 -90 -54   0  72  36  144  108 -144  180 -72 -108 -36
xbounds  = 339   0   0  288  288  309      21  51  72  72   0   0
           0  16  21   0  339  344      340   0  -0  344  324  324
           20  36  36  16   0   0      93  123  144  144  72  72
           72  88  93  72  51  56      52  72  72  56  36  36
           92 108 108  88  72  72      165 195 216 216 144 144
           144 160 165 144 123 128      124 144 144 128 108 108
           164 180 180 160 144 144      237 267 288 288 216 216
           216 232 237 216 195 200      196 216 216 200 180 180
           236 252 252 232 216 216      288 304 309 288 267 272
           268 288 288 272 252 252      308 324 324 304 288 288
           345 324 324  36  36  15      36  36 108 108  87  57
           20  15  36  57  52  36      108 108 180 180 159 129
           92  87 108 129 124 108      180 180 252 252 231 201
           164 159 180 201 196 180      252 252 324 324 303 273
           236 231 252 273 268 252      308 303 324 345 340 324
yvals    =  58  58  32   0   0  58  32   0   0  58  32   0   0  58  32
           0   0  32   0   0 -58 -58 -32 -58 -32 -58 -32 -58 -32 -32
ybounds  =  41  53  71  71  53  41      41  41  53  71  71  53
           11  19  41  53  41  19      -19 -7  11  19  7 -11
           -19 -11  7  19  11 -7      41  41  53  71  71  53
           11  19  41  53  41  19      -19 -7  11  19  7 -11
           -19 -11  7  19  11 -7      41  41  53  71  71  53
           11  19  41  53  41  19      -19 -7  11  19  7 -11
           -19 -11  7  19  11 -7      41  41  53  71  71  53
           11  19  41  53  41  19      -19 -7  11  19  7 -11
           -19 -11  7  19  11 -7      11  19  41  53  41  19
           -19 -7  11  19  7 -11      -19 -11  7  19  11 -7
           -41 -53 -71 -71 -53 -41      -53 -71 -71 -53 -41 -41
           -19 -41 -53 -41 -19 -11      -53 -71 -71 -53 -41 -41
           -19 -41 -53 -41 -19 -11      -53 -71 -71 -53 -41 -41
           -19 -41 -53 -41 -19 -11      -53 -71 -71 -53 -41 -41
           -19 -41 -53 -41 -19 -11      -19 -41 -53 -41 -19 -11

```

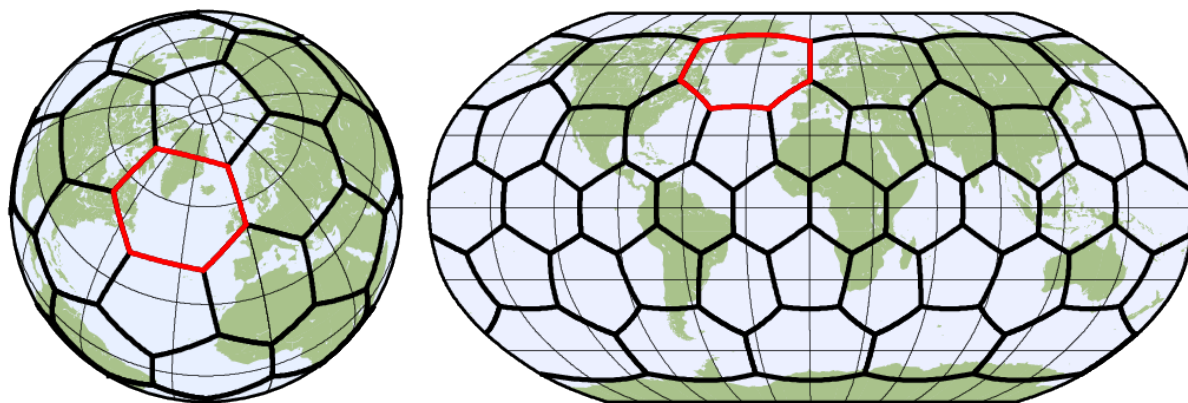


Fig. 2: Orthographic and Robinson projection of the unstructured grid

BIBLIOGRAPHY

- [BitInformation.jl] M Klöwer, M Razinger, JJ Dominguez, PD Düben and TN Palmer, 2021. Compressing atmospheric data into its real information content. *Nature Computational Science* 1, 713–724. 10.1038/s43588-021-00156-2
- [CDI] [Climate Data Interface](#), from the [Max Planck Institute for Meteorology](#)
- [CM-SAF] [Satellite Application Facility on Climate Monitoring](#), from the [German Weather Service \(Deutscher Wetterdienst, DWD\)](#)
- [CMOR-ref] [Climate Model Output Rewriter](#), from the [Program For Climate Model Diagnosis and Intercomparison \(PCMDI\)](#)
- [ecCodes] [API for GRIB decoding/encoding](#), from the [European Centre for Medium-Range Weather Forecasts \(ECMWF\)](#)
- [ECHAM] [The atmospheric general circulation model ECHAM5](#), from the [Max Planck Institute for Meteorology](#)
- [GMT] [The Generic Mapping Tool](#), from the [School of Ocean and Earth Science and Technology \(SOEST\)](#)
- [GrADS] [Grid Analysis and Display System](#), from the [Center for Ocean-Land-Atmosphere Studies \(COLA\)](#)
- [GRIB] [GRIB version 1](#), from the [World Meteorological Organisation \(WMO\)](#)
- [HDF5] [HDF version 5](#), from the [HDF Group](#)
- [INTERA] [INTERA Software Package](#), from the [Max Planck Institute for Meteorology](#)
- [Magics] [Magics Software Package](#), from the [European Centre for Medium-Range Weather Forecasts \(ECMWF\)](#)
- [MPIOM] [Ocean and sea ice model](#), from the [Max Planck Institute for Meteorology](#)
- [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 Meteorology](#)
- [REMO] [Regional Model](#), from the [Max Planck Institute for Meteorology](#)
- [Preisendorfer] [Rudolph W. Preisendorfer: Principal Component Analysis in Meteorology and Oceanography](#), Elsevier (1988)
- [PROJ] [Cartographic Projections Library](#), originally written by Gerald Evenden then of the USGS.
- [SCRIP] [SCRIP Software Package](#), from the [Los Alamos National Laboratory](#)
- [gzip] [Gzip compression software](#), developed at [University of New Mexico](#).
- [vonStorch] [Hans von Storch, Walter Zwiers: Statistical Analysis in Climate Research](#), Cambridge University Press (1999)
- [YAC] [YAC - A Coupling Library for Earth System Models](#), from [DKRZ and MPI for Meteorologie](#)

Symbols

- C
 - cdo command line option, 4
- F
 - cdo command line option, 5
- O
 - cdo command line option, 6
- P
 - cdo command line option, 6
- R
 - cdo command line option, 6
- S
 - cdo command line option, 6
- V
 - cdo command line option, 7
- absolute_taxis
 - cdo command line option, 4
- async_read
 - cdo command line option, 4
- check_data_range
 - cdo command line option, 4
- chunksize
 - cdo command line option, 4
- chunkspec
 - cdo command line option, 4
- cmor
 - cdo command line option, 5
- color
 - cdo command line option, 4
- copy_chunkspec
 - cdo command line option, 5
- diagnostic
 - cdo command line option, 6
- double
 - cdo command line option, 5
- eccodes
 - cdo command line option, 5
- filter
 - cdo command line option, 5
- force
 - cdo command line option, 5
- hdr_pad
 - cdo command line option, 5
- header_pad
 - cdo command line option, 5
- help
 - cdo command line option, 5
- netcdf_hdr_pad
 - cdo command line option, 5
- no_history
 - cdo command line option, 6
- nofile
 - cdo command line option, 5
- nsb
 - cdo command line option, 6
- operators
 - cdo command line option, 6
- overwrite
 - cdo command line option, 6
- pedantic
 - cdo command line option, 6
- percentile
 - cdo command line option, 6
- precision
 - cdo command line option, 6
- query
 - cdo command line option, 6
- reduce_dim
 - cdo command line option, 6
- reduce_dims
 - cdo command line option, 6
- regular
 - cdo command line option, 6
- relative_taxis
 - cdo command line option, 6
- remove_chunkspec
 - cdo command line option, 6
- shuffle
 - cdo command line option, 7
- silent
 - cdo command line option, 7
- single
 - cdo command line option, 7
- sortname
 - cdo command line option, 7
- timestat_date
 - cdo command line option, 7
- verbose
 - cdo command line option, 7
- version
 - cdo command line option, 7
- worker
 - cdo command line option, 7

cdo command line option, 7

-a
cdo command line option, 4

-b
cdo command line option, 4

-c
cdo command line option, 4

-f
cdo command line option, 5

-g
cdo command line option, 5

-h
cdo command line option, 5

-k
cdo command line option, 5

-m
cdo command line option, 5

-p
cdo command line option, 6

-r
cdo command line option, 6

-s
cdo command line option, 7

-t
cdo command line option, 7

-v
cdo command line option, 7

-w
cdo command line option, 7

-z
cdo command line option, 7

A

abs, 133

acos, 133

add, 136

addc, 135

addtrend, 232

adipot, 353

Adisit, 353

adisit, 353

aexpr, 129

aexprf, 129

after, 334

Afterburner, 334

air_density, 351

ap2pl, 253

Arith, 136

Arithc, 135

Arithdays, 145

Arithlat, 146

asin, 133

atan, 133

atan2, 136

B

bandpass, 336

Bitrounding, 49

bitrounding, 49

bottomvalue, 85

C

cat, 43

cdo command line option

- C, 4
- F, 5
- O, 6
- P, 6
- R, 6
- S, 6
- V, 7
- absolute_taxis, 4
- async_read, 4
- check_data_range, 4
- chunksize, 4
- chunkspec, 4
- cmor, 5
- color, 4
- copy_chunkspec, 5
- diagnostic, 6
- double, 5
- eccodes, 5
- filter, 5
- force, 5
- hdr_pad, 5
- header_pad, 5
- help, 5
- netcdf_hdr_pad, 5
- no_history, 6
- nofile, 5
- nsb, 6
- operators, 6
- overwrite, 6
- pedantic, 6
- percentile, 6
- precision, 6
- query, 6
- reduce_dim, 6
- reduce_dims, 6
- regular, 6
- relative_taxis, 6
- remove_chunkspec, 6
- shuffle, 7
- silent, 7
- single, 7
- sortname, 7
- timestat_date, 7
- verbose, 7
- version, 7
- worker, 7
- a, 4
- b, 4
- c, 4
- f, 5
- g, 5
- h, 5

-k, 5
 -m, 5
 -p, 6
 -r, 6
 -s, 7
 -t, 7
 -v, 7
 -w, 7
 -z, 7
 CDO_DOWNLOAD_PATH, 19
 CDO_FILE_SUFFIX, 57, 58, 60, 61, 236, 258
 CDO_GRIDSEARCH_RADIUS, 244, 249
 CDO_ICON_GRIDS, 19
 CDO_PCTL_NBINS, 23, 179, 185, 188, 191, 194, 200, 203, 212, 215, 218, 222, 329
 CDO_REMAP_NORM, 246, 249
 CDO_WEIGHT_MODE, 236
 Change, 108
 changemulti, 71
 chcode, 108
 chlevel, 108
 chlevelc, 108
 chlevelv, 108
 chname, 108
 chparam, 108
 chunit, 108
 cinfo, 27
 clone, 43
 CMOR, 287
 cmor, 287
 CMORlite, 363
 cmorlite, 363
 codetab, 40
 Collgrid, 64
 collgrid, 64
 Comp, 92
 Compc, 94
 Cond, 87
 Cond2, 88
 Condc, 89
 Consecstat, 155
 consecsum, 155
 consects, 155
 const, 343
 contour, 279
 Copy, 43
 copy, 43
 cos, 133

D
 dayadd, 138
 Dayarith, 138
 dayavg, 189
 daydiv, 138
 daymax, 189
 daymean, 189
 daymin, 189
 daymul, 138
 Daypctl, 191
 daypctl, 191
 dayrange, 189
 Daystat, 189
 daystd, 189
 daystd1, 189
 daysub, 138
 daysum, 189
 dayvar, 189
 dayvar1, 189
 delattribute, 100
 delcode, 73
 delete, 69
 delgridcell, 81
 delmulti, 71
 delname, 73
 delparam, 73
 delta_pressure, 350
 Deltat, 340
 deltat, 340
 Derivepar, 351
 Detrend, 230
 detrend, 230
 dhouravg, 206
 dhourmax, 206
 dhourmean, 206
 dhourmin, 206
 dhourrange, 206
 Dhourstat, 206
 dhourstd, 206
 dhourstd1, 206
 dhoursum, 206
 dhourvar, 206
 dhourvar1, 206
 Diff, 33
 diff, 33
 diffn, 33
 DIR_DCW, 24
 Distgrid, 62
 distgrid, 62
 div, 136
 divc, 135
 divcoslat, 146
 divdpm, 145
 divdpy, 145
 dminuteavg, 208
 dminutemax, 208
 dminutemean, 208
 dminutemin, 208
 dminuterange, 208
 Dminutestat, 208
 dminutestd, 208
 dminutestd1, 208
 dminutesum, 208
 dminutevar, 208
 dminutevar1, 208
 Duplicate, 52
 duplicate, 52

dv2ps, 263
dv2uv, 264

E

Eca_cdd, 293
eca_cdd, 293
Eca_cfd, 294
eca_cfd, 294
Eca_csu, 295
eca_csu, 295
Eca_cwd, 296
eca_cwd, 296
Eca_cwdi, 297
eca_cwdi, 297
Eca_cwfi, 298
eca_cwfi, 298
Eca_etccdi, 328
Eca_etr, 299
eca_etr, 299
Eca_fd, 300
eca_fd, 300
Eca_gsl, 301
eca_gsl, 301
Eca_hd, 302
eca_hd, 302
Eca_hwdi, 303
eca_hwdi, 303
Eca_hwfi, 304
eca_hwfi, 304
Eca_id, 305
eca_id, 305
Eca_pd, 314
eca_pd, 314
eca_r10mm, 314
eca_r20mm, 314
Eca_r75p, 306
eca_r75p, 306
Eca_r75ptot, 307
eca_r75ptot, 307
Eca_r90p, 308
eca_r90p, 308
Eca_r90ptot, 309
eca_r90ptot, 309
Eca_r95p, 310
eca_r95p, 310
Eca_r95ptot, 311
eca_r95ptot, 311
Eca_r99p, 312
eca_r99p, 312
Eca_r99ptot, 313
eca_r99ptot, 313
Eca_rr1, 316
eca_rr1, 316
Eca_rx1day, 317
eca_rx1day, 317
Eca_rx5day, 318
eca_rx5day, 318
Eca_sdii, 319

eca_sdii, 319
Eca_su, 320
eca_su, 320
Eca_tg10p, 321
eca_tg10p, 321
Eca_tg90p, 322
eca_tg90p, 322
Eca_tn10p, 323
eca_tn10p, 323
Eca_tn90p, 324
eca_tn90p, 324
Eca_tr, 325
eca_tr, 325
Eca_tx10p, 326
eca_tx10p, 326
Eca_tx90p, 327
eca_tx90p, 327
Enlarge, 120
enlarge, 120
ensavg, 158
ensbrs, 162
enscrps, 162
enskurt, 158
ensmax, 158
ensmean, 158
ensmedian, 158
ensmin, 158
enspctl, 158
ensrange, 158
ensrkhistspace, 161
ensrkhisttime, 161
ensroc, 161
ensskew, 158
Ensstat, 158
Ensstat2, 161
ensstd, 158
ensstd1, 158
enssum, 158
Ensval, 162
ensvar, 158
ensvar1, 158
environment variable
 CDO_DOWNLOAD_PATH, 19, 371
 CDO_FILE_SUFFIX, 57, 58, 60, 61, 236, 258, 371
 CDO_GRIDSEARCH_RADIUS, 244, 249, 371
 CDO_HISTORY_INFO, 371
 CDO_ICON_GRIDS, 19, 371
 CDO_PCTL_NBINS, 23, 179, 185, 188, 191, 194,
 200, 203, 212, 215, 218, 222, 329, 371
 CDO_REMAP_NORM, 246, 249, 371
 CDO_RESET_HISTORY, 371
 CDO_SVD_MODE, 234
 CDO_VERSION_INFO, 371
 CDO_WEIGHT_MODE, 234, 236
 DIR_DCW, 24
 FNORM_PRECISION, 234
 MAX_JACOBI_ITER, 234
 PLANET_RADIUS, 338

REMAP_AREA_MIN, 246, 248, 249, 371
 REMAP_EXTRAPOLATE, 240, 242, 244, 249, 371
 REMAPETA_PTOP, 250
 EOF, 234
 eof, 234
 eof3d, 234
 Eofcoeff, 236
 eofcoeff, 236
 eofspatial, 234
 eoftime, 234
 eq, 92
 eqc, 94
 etccdi_cdd, 293
 etccdi_csdi, 298
 etccdi_cwd, 296
 etccdi_fd, 300
 etccdi_id, 305
 etccdi_r1mm, 314
 etccdi_r95p, 328
 etccdi_r99p, 328
 etccdi_rx1day, 317
 etccdi_rx5day, 318
 etccdi_su, 320
 etccdi_tn10p, 328
 etccdi_tn90p, 328
 etccdi_tr, 325
 etccdi_tx10p, 328
 etccdi_tx90p, 328
 etccdi_wsdi, 304
 exp, 133
 Expr, 129
 expr, 129
 exprf, 129

F

Fdns, 358
 fdns, 358
 Filedes, 40
 Filter, 336
 fldavg, 164
 Fldcor, 225
 fldcor, 225
 fldcount, 164
 Fldcovar, 227
 fldcovar, 227
 fldint, 164
 fldkurt, 164
 fldmax, 164
 fldmean, 164
 fldmedian, 164
 fldmin, 164
 fldpctl, 164
 fldrange, 164
 fldskew, 164
 Fldstat, 164
 fldstd, 164
 fldstd1, 164
 fldsum, 164

fldvar, 164
 fldvar1, 164
 Fourier, 266
 fourier, 266

G

ge, 92
 gec, 94
 genbic, 241
 genbil, 239
 gencon, 245
 gendis, 243
 genknn, 243
 genlaf, 247
 genlevelbounds, 112
 gennn, 243
 Getgridcell, 342
 gh2hl, 254
 gheight, 351
 gheight_half, 351
 gmtcells, 276
 gmtxyz, 276
 gp2sp, 260
 Gradsdes, 332
 gradsdess, 332
 graph, 284
 grfill, 279
 gridarea, 338
 gridboxavg, 171
 gridboxkurt, 171
 gridboxmax, 171
 gridboxmean, 171
 gridboxmedian, 171
 gridboxmin, 171
 gridboxrange, 171
 gridboxskew, 171
 Gridboxstat, 171
 gridboxstd, 171
 gridboxstd1, 171
 gridboxsum, 171
 gridboxvar, 171
 gridboxvar1, 171
 Gridcell, 338
 gridcellindex, 342
 griddes, 40
 gridweights, 338
 gt, 92
 gtc, 94

H

Healpix, 366
 highpass, 336
 histcount, 355
 histfreq, 355
 histmean, 355
 Histogram, 355
 histsum, 355
 houravg, 186

hourmax, 186
hourmean, 186
hourmin, 186
Hourpctl, 188
hourpctl, 188
hourrange, 186
Hourstat, 186
hourstd, 186
hourstd1, 186
hoursum, 186
hourvar, 186
hourvar1, 186
hpdegrade, 366
hpupgrade, 366
Hurr, 362
hurr, 362

I

ifnotthen, 87
ifnotthenc, 89
ifthen, 87
ifthenc, 89
ifthenelse, 88
import_binary, 268
import_cmsaf, 269
Importbinary, 268
Importcmsaf, 269
Info, 27
info, 27
infon, 27
Input, 271
input, 271
inputext, 271
inputsrv, 271
int, 133
Intlevel, 255
intlevel, 255
Intlevel3d, 256
intlevel3d, 256
intntime, 257
Inttime, 257
inttime, 257
Intyear, 258
intyear, 258
Invert, 113
invertlat, 113
Invertlev, 113
invertlev, 113
isosurface, 85

L

le, 92
lec, 94
ln, 133
log10, 133
lowpass, 336
lt, 92
ltc, 94

M

Maggraph, 284
Magplot, 279
Magvector, 282
map, 27
Mapreduce, 90
Maskbox, 116
maskindexbox, 116
masklonlatbox, 116
Maskregion, 115
maskregion, 115
Mastrfu, 349
mastrfu, 349
Math, 133
max, 136
maxc, 135
meravg, 169
Merge, 54
merge, 54
Mergegrid, 53
mergegrid, 53
mergetime, 54
merkurt, 169
mermax, 169
mermean, 169
mermedian, 169
mermin, 169
merpctl, 169
merrange, 169
merskew, 169
Merstat, 169
merstd, 169
merstd1, 169
mersum, 169
mervar, 169
mervar1, 169
min, 136
minc, 135
ml2pl, 252
monadd, 139
Monarith, 139
monavg, 192
monddiv, 139
monmax, 192
monmean, 192
monmin, 192
monmul, 139
Monpctl, 194
monpctl, 194
monrange, 192
Monstat, 192
monstd, 192
monstd1, 192
monsub, 139
monsum, 192
monvar, 192
monvar1, 192
Mrotuvb, 348

mrotuvb, 348
mul, 136
mulc, 135
mulcoslat, 146
muldpm, 145
muldpy, 145

N

NCL_wind, 368
ndate, 35
ne, 92
nec, 94
ngridpoints, 35
ngrid, 35
Ninfo, 35
nint, 133
nlevel, 35
nmon, 35
not, 133
npar, 35
ntime, 35
nyear, 35

O

Output, 272
output, 272
outputtext, 272
outputf, 272
Outputgmt, 276
outputint, 272
outputsrv, 272
Outputtab, 274
outputtab, 274

P

Pack, 45
pack, 45
partab, 40
PLANET_RADIUS, 338
pow, 133
Pressure, 350
pressure, 350
pressure_half, 350
projuvLatLon, 346

Q

Query, 67
query, 67

R

random, 343
reci, 133
reducegrid, 90
reference_manual, 24
Regres, 229
regres, 229
Remap, 249

remap, 249
REMAP_AREA_MIN, 246, 248, 249
REMAP_EXTRAPOLATE, 240, 242, 244, 249
remapavg, 173
Remapbic, 241
remapbic, 241
Remapbil, 239
remapbil, 239
Remapcon, 245
remapcon, 245
remapdis, 243
Remapeta, 250
remapeta, 250
Remapknn, 243
remapknn, 243
remapkurt, 173
Remaplaf, 247
remaplaf, 247
remapmax, 173
remapmean, 173
remapmedian, 173
remapmin, 173
remapnn, 243
remaprange, 173
remapskew, 173
Remapstat, 173
remapstd, 173
remapstd1, 173
remapsum, 173
remapvar, 173
remapvar1, 173
Replace, 51
replace, 51
Replacevalues, 341
Rhopot, 354
rhopot, 354
Rotuv, 347
rotuvb, 347
rotuvNorth, 346
runavg, 180
runmax, 180
runmean, 180
runmin, 180
Runpctl, 182
runpctl, 182
runrange, 180
Runstat, 180
runstd, 180
runstd1, 180
runsum, 180
runvar, 180
runvar1, 180

S

Samplegrid, 82
samplegrid, 82
sealevelpressure, 351
seasavg, 201

seasmax, 201
seasmean, 201
seasmin, 201
Seaspctl, 203
seaspctl, 203
seasrange, 201
Seasstat, 201
seasstd, 201
seasstd1, 201
seassum, 201
seasvar, 201
seasvar1, 201
Selbox, 78
selcircle, 80
selcode, 73
seldate, 76
selday, 76
Select, 69
select, 69
selgrid, 73
Selgridcell, 81
selgridcell, 81
selhour, 76
selindexbox, 78
sellevel, 73
sellevidx, 73
sellonlatbox, 78
selltype, 73
selmonth, 76
Selmulti, 71
selmulti, 71
selname, 73
selparam, 73
Selregion, 80
selregion, 80
selseason, 76
selsmon, 76
selstdname, 73
Selsurface, 85
seltabnum, 73
Seltime, 76
seltime, 76
Seltimeidx, 84
seltimeidx, 84
seltimestep, 76
Selvar, 73
selyear, 76
Selyearidx, 83
selyearidx, 83
selzaxis, 73
selzaxisname, 73
seq, 343
Set, 104
Setattribute, 100
setattribute, 100
Setbox, 118
setcalendar, 106
Setchunkspec, 47
setchunkspec, 47
setcindexbox, 118
setclonlatbox, 118
setcode, 104
setcodetab, 104
setctomiss, 121
setdate, 106
setday, 106
Setfilter, 48
setfilter, 48
Setgrid, 110
setgrid, 110
setgridarea, 110
Setgridcell, 126
setgridcell, 126
setgridmask, 110
setgridtype, 110
Sethalo, 356
sethalo, 356
setlevel, 104
setltype, 104
setmaxsteps, 104
Setmiss, 121
setmiss, 136
setmisstoc, 121
setmisstodis, 121
setmisstonn, 121
setmissval, 121
setmon, 106
setname, 104
setparam, 104
Setpartab, 102
setpartabn, 102
setpartabp, 102
setprojparams, 110
setreftime, 106
setrtoc, 341
setrtoc2, 341
setrtomiss, 121
setstdname, 104
settaxis, 106
settbounds, 106
Settime, 106
settime, 106
settunits, 106
setunit, 104
setvals, 341
setvrange, 121
setyear, 106
Setzaxis, 112
setzaxis, 112
shaded, 279
shifttime, 106
shiftx, 114
Shiftxy, 114
shifty, 114
Showattribute, 39
showattribute, 39

showchunkspec, 37
 showcode, 37
 showdate, 37
 showfilter, 37
 showformat, 37
 Showinfo, 37
 showlevel, 37
 showltype, 37
 showmon, 37
 showname, 37
 showstdname, 37
 showtime, 37
 showtimestamp, 37
 showyear, 37
 sin, 133
 Sinfo, 29
 sinfo, 29
 sinfon, 29
 Smooth, 339
 smooth, 339
 smooth9, 339
 sp2gp, 260
 sp2sp, 262
 Speconv, 262
 Spectral, 260
 Split, 56
 splitcode, 56
 Splitdate, 61
 splitdate, 61
 splitday, 58
 splitensemble, 56
 splitgrid, 56
 splithour, 58
 splitlevel, 56
 splitmon, 58
 splitname, 56
 splitparam, 56
 splitseas, 58
 Splitsel, 60
 splitsel, 60
 splittabnum, 56
 Splittime, 58
 splityear, 58
 splityearmon, 58
 splitzaxis, 56
 sqr, 133
 sqrt, 133
 stdatm, 343
 Strbre, 360
 strbre, 360
 Strgal, 361
 strgal, 361
 Strwin, 359
 strwin, 359
 sub, 136
 subc, 135
 subtrend, 232
 Symmetrize, 367

symmetrize, 367

T

tan, 133
 Tee, 44
 tee, 44
 timavg, 183
 Timcor, 226
 timcor, 226
 Timcovar, 227
 timcovar, 227
 Timcumsum, 154
 timcumsum, 154
 Timfillmiss, 124
 timfillmiss, 124
 timmax, 183
 timmaxidx, 183
 timmean, 183
 timmin, 183
 timminidx, 183
 Timpctl, 185
 timpctl, 185
 timrange, 183
 timselavg, 177
 timselmax, 177
 timselmean, 177
 timselmin, 177
 Timselpctl, 179
 timselpctl, 179
 timselrange, 177
 Timselstat, 177
 timselstd, 177
 timselstd1, 177
 timselsum, 177
 timselvar, 177
 timselvar1, 177
 Timsort, 345
 timsort, 345
 Timstat, 183
 timstd, 183
 timstd1, 183
 timsum, 183
 timvar, 183
 timvar1, 183
 timyearmean, 195
 Timyearstat, 195
 topo, 343
 topvalue, 85
 Trend, 231
 trend, 231
 Trendarith, 232

U

Unpack, 46
 unpack, 46
 uv2dv, 264
 uv2dv_cfd, 368
 uv2vr_cfd, 368

uvDestag, 346

V

Vargen, 343
varsavg, 156
varskurt, 156
varsmay, 156
varsmean, 156
varsmedian, 156
varsmin, 156
varspctl, 156
varsrage, 156
varsskew, 156
Varsstat, 156
varsstd, 156
varsstd1, 156
varssum, 156
varsvar, 156
varsvar1, 156
vct, 40
vector, 282
Verifygrid, 365
verifygrid, 365
vertavg, 175
Vertfillmiss, 124
vertfillmiss, 124
Vertintap, 253
Vertintgh, 254
Vertintml, 252
vertmax, 175
vertmean, 175
vertmin, 175
vertrange, 175
Vertstat, 175
vertstd, 175
vertstd1, 175
vertsum, 175
vertvar, 175
vertvar1, 175

W

Wct, 357
wct, 357
Wind, 264
Wind2, 263
WindTrans, 346

X

XSinfo, 31
xsinfo, 31
xsinfop, 31

Y

ydayadd, 142
Ydayarith, 142
ydayavg, 210
ydaydiv, 142
ydaymax, 210

ydaymean, 210
ydaymin, 210
ydaymul, 142
Ydaypctl, 212
ydaypctl, 212
ydayrange, 210
Ydaystat, 210
ydaystd, 210
ydaystd1, 210
ydaysub, 142
ydaysum, 210
ydayvar, 210
ydayvar1, 210
ydrunavg, 219
ydrunmax, 219
ydrunmean, 219
ydrunmin, 219
Ydrunpctl, 222
ydrunpctl, 222
Ydrunstat, 219
ydrunstd, 219
ydrunstd1, 219
ydrunsum, 219
ydrunvar, 219
ydrunvar1, 219
yearadd, 140
Yeararith, 140
yearavg, 197
yeardiv, 140
yearmax, 197
yearmaxidx, 197
yearmean, 197
yearmin, 197
yearminidx, 197
yearmonmean, 196
Yearmonstat, 196
yearmul, 140
Yearpctl, 200
yearpctl, 200
yerrange, 197
Yearstat, 197
yearstd, 197
yearstd1, 197
yearsab, 140
yearsum, 197
yearvar, 197
yearvar1, 197
yhouradd, 141
Yhourarith, 141
yhouravg, 204
yhourdiv, 141
yhourmax, 204
yhourmean, 204
yhourmin, 204
yhourmul, 141
yhourrange, 204
Yhourstat, 204
yhourstd, 204

yhourstd1, 204
yhoursub, 141
yhoursum, 204
yhourvar, 204
yhourvar1, 204
ymonadd, 143
Ymonarith, 143
ymonavg, 213
Ymoncomp, 96
ymondiv, 143
ymoneq, 96
ymonge, 96
ymongt, 96
ymonle, 96
ymonlt, 96
ymonmax, 213
ymonmean, 213
ymonmin, 213
ymonmul, 143
ymonne, 96
Ymonpctl, 215
ymonpctl, 215
ymonrange, 213
Ymonstat, 213
ymonstd, 213
ymonstd1, 213
ymonsub, 143
ymonsum, 213
ymonvar, 213
ymonvar1, 213
yseasadd, 144
Yseasarith, 144
yseasavg, 216
Yseascomp, 97
yseasdiv, 144
yseaseq, 97
yseasge, 97
yseasgt, 97
yseasle, 97
yseaslt, 97
yseasmax, 216
yseasmean, 216
yseasmin, 216
yseasmul, 144
yseasne, 97
Yseaspctl, 218
yseaspctl, 218
yseasrange, 216
Yseasstat, 216
yseasstd, 216
yseasstd1, 216
yseassub, 144
yseassum, 216
yseasvar, 216
yseasvar1, 216
zonavg, 167
zonint, 167
zonkurt, 167
zonmax, 167
zonmean, 167
zonmedian, 167
zonmin, 167
zonpctl, 167
zonrange, 167
zonskew, 167
Zonstat, 167
zonstd, 167
zonstd1, 167
zonsum, 167
zonvar, 167
zonvar1, 167

Z

zaxisdes, 40