CDO User's Guide

Climate Data Operators
Version 0.9.6
April 2005

Uwe Schulzweida
Max-Planck-Institute for Meteorology

Contents

1 Introduction

1.1 Grid description

In some situations it is necessary to give a description of a grid. These situations are

- Changing the grid description (operator: setgrid)
- Horizontal regridding (operator: interpolate, remapbil, remapbic, remapcon, remapdis)
- Generating variables (operator: const, random)

There are several possibilities to define a horizontal grid.

1.1.1 Predefined grids

The following pre-defined grid names can be used:

Global regular grid: `r<LON>x<LAT>`

`r<LON>x<LAT>` defines a global regular grid. The number of the longitutes `<LON>` and the latitudes `<LAT>` can be selected at will. The longitudes starts at 0° with an increment of (360/`<LON>`)°. The latitudes go from south to north with an increment of (180/`<LAT>`)°.

Global gaussian grid: `t<RES>grid`

`t<RES>grid` defines a global gaussian grid. Each valid triangular resolution can be used for `<RES>`. The longitudes starts at 0° with an increment of (360/nlon)°. The gaussian latitudes go from north to south.

1.1.2 Grids from data files

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

1.1.3 SCRIP grids

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

Example of the netCDF header from a MPIOM1 GROB3 grid:

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

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

1.1.4 PINGO grids

PINGO is using a very simple grid description in ASCII format. You can use it to describe regular longitude/latitude or global gaussian grids. All PINGO grid description files are supported. For more information about this format see [PINGO].

Example of a T21 gaussian grid:

```
Grid Description File
(Comments start at non digit characters and end at end of line)
First part: The dimensions.
64 32 = Number of longitudes and latitudes
Second part: The listed longitudes.
2 means equidistant longitudes
0.000000 5.625000 = Most western and second most western longitude
Third part: The listed latitudes.
32 means all 32 latitudes are given in the following list:
 85.761  80.269  74.745  69.213  63.679  58.143  52.607  47.070
 41.532  35.995  30.458  24.920  19.382  13.844   8.307   2.769
 -2.769  -8.307 -13.844 -19.382 -24.920 -30.458 -35.995 -41.532
-47.070 -52.607 -58.143 -63.679 -69.213 -74.745 -80.269 -85.761
```

1.1.5 CDO grids

The CDO grid description is an ASCII formatted file. It is a common grid description for all available grids. The following keywords can be used to describe a grid:

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

Which keywords are necessary depents on the gridtype. The next table gives an overview of the default values or the array size for the different grid types.

gridtype	lonlat	gaussian	curvilinear	cell
gridsize	xsize*ysize	xsize*ysize	xsize*ysize	**ncell**
xsize	**nlon**	**nlon**	**nlon**	gridsize
ysize	**nlat**	**nlat**	**nlat**	gridsize
xvals	xsize	xsize	gridsize	gridsize
yvals	ysize	ysize	gridsize	gridsize
xnpole	0			
ynpole	90			
nvertex	2	2	4	**nv**
xbounds	2*xsize	2*xsize	4*gridsize	nv*gridsize
ybounds	2*ysize	2*ysize	4*gridsize	nv*gridsize

The keywords nvertex, xbounds and ybounds are optional if the area weights are not needed.

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
```

Example of a global regular grid with 60x30 points:

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

Example of a regional rotated lon/lat grid:

```
gridtype = lonlat
xsize    = 81
ysize    = 91
xfirst   =  -19.25
xinc     =    0.5
yfirst   =  -24.75
yinc     =    0.5
xnpole   = -170
ynpole   =   32.5
```

1.2 Time axis

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

1.2.1 Absolute time

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

1.2.2 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 used calendar. CDO supports the standard gregorian, 360 days, 365 days and 366 days calendars. The relative time axis is preferably used by weather forecast models. In netCDF files the relative time axis is to be recognized by the unit of the time: "*time-units* `since` *reference-time*", e.g "`days sincs 1965-9-1 14:30`".

1.2.3 Conversion of the time

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

1.3 Parameter table

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

Example of a 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]
```

1.4 Missing values

All operators can handle missing values. The default missing value for GRIB, SERVICE and EXTRA files is -9e+33. The CDO option '-m <missval>' overwrites the default missing value. In netCDF files the variable attribute '_FillValue' is used as missing value. The operator 'setmissval' can be used to set a new missing value.

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

addition	**b**	**miss**
a	$a+b$	$miss$
miss	$miss$	$miss$

subtraction	**b**	**miss**
a	$a-b$	$miss$
miss	$miss$	$miss$

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

division	**b**	**0**	**miss**
a	a/b	$miss$	$miss$
0	0	$miss$	$miss$
miss	$miss$	$miss$	$miss$

maximum	**b**	**miss**
a	$max(a,b)$	a
miss	b	$miss$

minimum	**b**	**miss**
a	$min(a,b)$	a
miss	b	$miss$

The handling of missing values by the operations ”minimum” and ”maximum” may be surprising, but is turned out that the definition given here is more related to what is 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 statistics functions ignore missing values, treading them as not belonging to the sample, with the side-effect of a reduced sample size.

1.4.1 Mean and average

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

2 Reference manual for all operators

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

$i(t)$	Timestep t of ifile
$i(t, x)$	Element number x of the field of timestep t of `ifile`
$o(t)$	Timestep t of ofile
$o(t, x)$	Element number x of the field of timestep t of `ofile`

2.1 Information

File information

info `ifiles`

Prints simple statistics for each field of a file. For each field the operator print in one line the:

- field number
- date and time
- code
- level
- size of the grid
- number of missing values
- minimum, mean and maximum

The mean value is computed without the use of area weight!

File information

infov `ifiles`

The same as operator info. Using the name instead of the code number to identify the variable.

Print simple map

map `ifiles`

Prints simple statistics and a map for each field of a file. The map will be printed only for fields on a rectangular grid.

Short file information

sinfo `ifile`

Prints short information for each variable of a file. For each variable the operator print in one line the:

- variable number
- institute and source
- code and codetable
- horizontal grid size and number
- vertical grid size and number

Short file information

sinfov `ifile`

The same as operator sinfo. Using the name instead of the code number to identify the variable.

Differences of two files

diff `ifile1 ifile2`

Print statistics over differences of two files. For each pair of fields the operator print in one line the:

- field number
- date and time
- code
- level
- size of the grid
- number of missing values
- occurrence of coefficient pairs with different signs
- occurrence of zero values
- maximum absolute difference of coefficient pairs
- maximum relative difference of non-zero coefficient pairs with equal signs

Differences of two files

diffv `ifile1 ifile2`

The same as operator diff. Using the name instead of the code number to identify the variable.

Number of years

nyear `ifile`

Prints the number of different years.

Number of months

nmon `ifile`

Prints the number of different combinations of years and months.

Number of dates

ndate `ifile`

Prints the number of different dates.

Number of timesteps

ntime `ifile`

Prints the number of timesteps.

Number of codes

ncode `ifile`

Prints the number of different codes.

Number of variables

nvar `ifile`

Prints the number of different variables.

Number of levels

nlevel `ifile`

Prints the number of levels for each variable.

Show years

showyear `ifile`

Prints all different years.

Show months

showmon `ifile`

Prints all different months.

Show dates

showdate `ifile`

Prints all different dates.

Show timesteps

showtime `ifile`

Prints all timesteps.

Show codes

showcode `ifile`

Prints the code of all different variables.

Show variable names

showvar `ifile`

Print all different variable names.

Show levels

showlevel `ifile`

Prints all levels for each variable.

Variable description

vardes `ifile`

Prints a table with a description of all variables. For each variable the operator print in one line the code, name, description and units.

Grid description

griddes `ifile`

Prints the description of all grids in a file.

Vertical coordinate table

vct `ifile`

Prints the vertical coordinate table.

2.2 File operations

Copy files

copy `ifiles ofile`

Copy `ifiles` to `ofile`. Each input file must have the same variables with complete timesteps.

Concatenate files

cat `ifiles ofile`

Concatenate `ifiles` and add the result at the end of `ofile`. Each input file must have the same variables with complete timesteps. The output file musst exist.

Merge files

merge `ifiles ofile`

Merge all variables of `ifiles` to `ofile`. Each input file must have different variables with the same number of timesteps.

Split codes

splitcode `ifile oprefix`

Splits a file into pieces, one for each code.

Split variables

splitvar `ifile oprefix`

Splits a file into pieces, one for each variable.

Split levels

splitlevel `ifile oprefix`

Splits a file into pieces, one for each level.

Split grids

splitgrid `ifile oprefix`

Splits a file into pieces, one for each grid.

Split zaxis

splitzaxis `ifile oprefix`

Splits a file into pieces, one for each zaxis.

Split hours

splithour `ifile oprefix`

Splits a file into pieces, one for each hour.

Split days

splitday `ifile oprefix`

Splits a file into pieces, one for each day.

Split months

splitmon `ifile oprefix`

Splits a file into pieces, one for each month.

Split seasons

splitseas `ifile oprefix`

Splits a file into pieces, one for each season.

Split years

splityear `ifile oprefix`

Splits a file into pieces, one for each year.

Split records

splitrec `ifile oprefix`

Splits a file into pieces, one for each record.

2.3 Formatted I/O

ASCII output

output `ifiles`

Prints all values to standard output.

Integer output

outputint `ifiles`

Prints all values rounded to the nearest intergers to standard output.

SERVICE output

outputsrv `ifiles`

Prints all values to standard output. Each field with a header of 8 integers (SERVICE likely).

EXTRA output

outputext `ifiles`

Prints all values to standard output. Each field with a header of 4 integers (EXTRA likely).

2.4 Generation of variables

Constant variable

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

Generates a constant variable.
Parameter:

const	FLOAT	Constant
grid	STRING	Grid description file or name

Variable with random values

random,*grid* `ofile`

Generates a variable with rectangularly distrubuted random numbers in the interval [0,1].
Parameter:

grid	STRING	Grid description file or name

Duplicate variables

vardup `ifile ofile`

Duplicate all variables.

Multiply variables

varmul,*nmul* `ifile ofile`

Multiply all variables.
Parameter:

nmul	INTEGER	Number of multiplications

2.5 Manipulating the header/field

Set parameter table

setpartab,*table* `ifile ofile`

Sets the parameter table for all variables.
Parameter:

table	STRING	Parameter table file or name

Set code

setcode,*code* `ifile ofile`

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

code	INTEGER	Code number

Set variable name

setvar,*name* `ifile ofile`

Sets the name of the first variable.
Parameter:

name	STRING	Variable name

Set date

setdate,*date* `ifile ofile`

Sets the date in every timestep to the same given value.

Set time

settime,*time* `ifile ofile`

Sets the time in every timestep to the same given value.

Set day

setday,*day* `ifile ofile`

Sets the day in every timestep to the same given value.

Set month

setmon,*month* `ifile ofile`

Sets the month in every timestep to the same given value.

Set year

setyear,*year* `ifile ofile`

Sets the year in every timestep to the same given value.

Set time units

settunits,*units* `ifile ofile`

Sets the time units.
Parameter:

units	STRING	Base units of the time axis (minute, hour, day, month, year).

Set time axis

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

Sets the time axis.
Parameter:

date	INTEGER	Start date (e.g. 19780130)
time	INTEGER	Start time (e.g. 1800)
inc	STRING	Optional increment (e.g. 12hour) [default: 0hour]

Set reference time

setreftime,*date,time* `ifile ofile`

Sets the reference time of an relative time axis.
Parameter:

date	INTEGER	Reference date (e.g. 19780130)
time	INTEGER	Reference time (e.g. 1800)

Shift time steps

shifttime,*sval* `ifile ofile`

Shifts all time steps by the parameter sval.
Parameter:

sval	STRING	Shift value (e.g. -3hour)

Change code

chcode,*ocode,ncode,...* `ifile ofile`

Changes some user given codes to new user given values.
Parameter:

ocode,ncode,...	INTEGER	Pairs of old and new code

Set grid

setgrid,*grid* `ifile ofile`

Sets the grid information for all variables.
Parameter:

grid	STRING	Grid description file or name of the target grid

Set grid type

setgridtype,*gridtype* `ifile ofile`

Sets the grid type for all grids to a user given value.
Parameter:

gridtype	STRING	New grid type (curvilinear or cell)

Set global attribute

setgatt,*attname,attstring* `ifile ofile`

Sets one user defined global text attribute.
Parameter:

attname,attstring	STRING	Name and text of the global attribute

Set global attributes

setgatts,*attfile* `ifile ofile`

Sets user defined global text attributes. The name and text of the global attrubutes are read from a file.
Parameter:

attfile	STRING	File name which contains global attributes

Invert latitude

invertlat `ifile ofile`

Invert the latitude of a field.

Invert longitude

invertlon `ifile ofile`

Invert the longitude of a field.

Invert latitude decription

invertlatdes `ifile ofile`

Invert only the latitude decription of a field.

Invert longitude decription

invertlondes `ifile ofile`

Invert only the longitude decription of a field.

Invert latitude data

invertlatdata `ifile ofile`

Invert only the latitude data of a field.

Invert longitude data

invertlondata `ifile ofile`

Invert only the longitude data of a field.

2.6 Selection

Select codes

selcode,*codes* `ifile ofile`

Selects all fields with a code in a user given list.
Parameter:

codes	INTEGER	Comma separated list of codes

Delete codes

delcode,*codes* `ifile ofile`

Deletes all fields with a code in a user given list.
Parameter:

codes	INTEGER	Comma separated list of codes

Select variables

selvar,*vars* `ifile ofile`

Selects all fields with a variable name in a user given list.
Parameter:

vars	STRING	Comma separated list of variable names

Delete variables

delvar,*vars* `ifile ofile`

Deletes all fields with a variable name in a user given list.
Parameter:

vars	STRING	Comma separated list of variable names

Select levels

sellevel,*levels* `ifile ofile`

Selects all fields with a level in a user given list.
Parameter:

levels	FLOAT	Comma separated list of levels

Select grids

selgrid,*grids* `ifile ofile`

Selects all fields with a grid in a user given list.
Parameter:

grids	INTEGER	Comma separated list of grids

Select zaxis

selzaxis,*zaxis* `ifile ofile`

Selects all fields with a zaxis in a user given list.
Parameter:

zaxis	INTEGER	Comma separated list of zaxis

Select records

selrec,*records* `ifile ofile`

Selects all fields with a record number in a user given list. This operator works only with GRIB, SERVICE and EXTRA data!
Parameter:

records	INTEGER	Comma separated list of records

Select timesteps

seltimestep,*timesteps* `ifile ofile`

Selects all timesteps with a timestep in a user given list.
Parameter:

timesteps	INTEGER	Comma separated list of timesteps

Select times

seltime,*times* `ifile ofile`

Selects all timesteps with a time in a user given list.
Parameter:

times	INTEGER	Comma separated list of times

Select hours

selhour,*hours* `ifile ofile`

Selects all timesteps with a hour in a user given list.
Parameter:

hours	INTEGER	Comma separated list of hours

Select days

selday,*days* `ifile ofile`

Selects all timesteps with a day in a user given list.
Parameter:

days	INTEGER	Comma separated list of days

Select months

selmon,*months* `ifile ofile`

Selects all timesteps with a month in a user given list.
Parameter:

months	INTEGER	Comma separated list of months

Select seasons

selseas,*seasons* `ifile ofile`

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

seasons	STRING	Comma separated list of seasons (DJF, MAM, JJA, SON)

Select years

selyear,*years* `ifile ofile`

Selects all timesteps with a year in a user given list.
Parameter:

years	INTEGER	Comma separated list of years

Select dates

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

Selects all timesteps with a date in a given range.
Parameter:

date1	INTEGER	Start date
date2	INTEGER	End date

Select lon/lat box

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

Selects a longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box.
Parameter:

lon1	FLOAT	Western longitude
lon2	FLOAT	Eastern longitude
lat1	FLOAT	Southern or northern latitude
lat2	FLOAT	Northern or southern latitude

Select index box

selindexbox,*ilon1*,*ilon2*,*ilat1*,*ilat2* `ifile ofile`

Selects an index box. The user has to give the indexes of the edges of the box.
Parameter:

ilon1	INTEGER	Index of first longitude
ilon2	INTEGER	Index of last longitude
ilat1	INTEGER	Index of first latitude
ilat2	INTEGER	Index of last latitude

2.7 Missing values

Set a new missing value

setmissval,*miss* `ifile ofile`

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

Parameter:

miss	FLOAT	New missing value

Set constant to missing value

setctomiss,*c* `ifile ofile`

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

Parameter:

c	FLOAT	Constant

Set missing value to constant

setmisstoc,*c* `ifile ofile`

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

Parameter:

c	FLOAT	Constant

Set range to missing value

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

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

Parameter:

rmin	FLOAT	Lower bound
rmax	FLOAT	Upper bound

2.8 Sorting

Sort by code number

sortcode `ifile ofile`

Sorts all variables by the code number.

Sort by variable name

sortvar `ifile ofile`

Sorts all variables by the name.

Sort by level

sortlevel `ifile ofile`

Sorts the levels of all variables.

Sort over the time

timsort `ifile ofile`

Sorts for every field position the elements in ascending order. After sorting it is

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

2.9 Arithmetic processor

Evaluate expressions

expr,*instr* `ifile ofile`

This Operator arithmetically processes every timestep of ifile. The processing instructions are read from the parameter. Each individual assignment statement must end with a semi-colon.
Parameter:

instr STRING Processing instructions

Evaluate expressions from script file

exprf,*filename* `ifile ofile`

This Operator arithmetically processes every timestep of ifile. Contrary to expr the processing instructions are read from a file.
Parameter:

filename STRING File with processing instructions

2.10 Arithmetic

Add by constant

addc,*c* `ifile ofile`

$o(t, x) = i(t, x) + c$

Parameter:

c	FLOAT	Constant

Subtract by constant

subc,c `ifile ofile`

$o(t, x) = i(t, x) - c$

Parameter:

c	FLOAT	Constant

Multiply by constant

mulc,c `ifile ofile`

$o(t, x) = i(t, x) * c$

Parameter:

c	FLOAT	Constant

Divide by constant

divc,c `ifile ofile`

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

Parameter:

c	FLOAT	Constant

Add two fields

add `ifile1 ifile2 ofile`

$o(t, x) = i_1(t, x) + i_2(t, x)$

Subtract two fields

sub `ifile1 ifile2 ofile`

$o(t, x) = i_1(t, x) - i_2(t, x)$

Multiply two fields

mul `ifile1 ifile2 ofile`

$o(t, x) = i_1(t, x) * i_2(t, x)$

Divide two fields

div `ifile1 ifile2 ofile`

$o(t, x) = i_1(t, x)/i_2(t, x)$

Minimum of two fields

min `ifile1 ifile2 ofile`

$o(t,x) = MIN(i_1(t,x), i_2(t,x))$

Maximum of two fields

max `ifile1 ifile2 ofile`

$o(t,x) = MAX(i_1(t,x), i_2(t,x))$

Add multi-year monthly time averages

ymonadd `ifile1 ifile2 ofile`

Add a time series and a multi-year monthly time averages.

Subtract multi-year monthly time averages

ymonsub `ifile1 ifile2 ofile`

Subtract a time series and a multi-year monthly time averages.

Multiply multi-year monthly time averages

ymonmul `ifile1 ifile2 ofile`

Multiply a time series and a multi-year monthly time averages.

Divide multi-year monthly time averages

ymondiv `ifile1 ifile2 ofile`

Divide a time series and a multi-year monthly time averages.

2.11 Mathematical functions

Square

sqr `ifile ofile`

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

Calculates the value of i(t,x) raised to the power of 2.

Square root

sqrt `ifile ofile`

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

Calculates the non-negative square root of i(t,x).

Exp

exp `ifile ofile`

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

Calculates e (the base of natural logarithms) raised to the power of i(t,x).

Logarithm

log `ifile ofile`

$o(t,x) = log(i(t,x))$

Calculates the natural logarithm of i(t,x).

Logarithm base 10

log10 `ifile ofile`

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

Calculates the base-10 logarithm of i(t,x).

Sine

sin `ifile ofile`

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

Calculates the sine of i(t,x), where i(t,x) is given in radians.

Cosine

cos `ifile ofile`

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

Calculates the cosine of i(t,x), where i(t,x) is given in radians.

Tangent

tan `ifile ofile`

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

Calculates the tangent of i(t,x), where i(t,x) is given in radians.

Arcus sine

asin `ifile ofile`

$o(t,x) = asin(i(t,x))$

Calculates the arcus sine of i(t,x); that is the value whose sine is i(t,x).

Arcus cosine

acos `ifile ofile`

$o(t,x) = acos(i(t,x))$

Calculates the arcus cosine of i(t,x); that is the value whose cosine is i(t,x).

Arcus tangent

atan `ifile ofile`

$o(t,x) = atan(i(t,x))$

Calculates the arcus tangent of i(t,x); that is the value whose tangent is i(t,x).

2.12 Comparisons

Equal

eq `ifile1 ifile2 ofile`

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

Not equal

ne `ifile1 ifile2 ofile`

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

Less equal

le `ifile1 ifile2 ofile`

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

Less then

lt `ifile1 ifile2 ofile`

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

Greater equal

ge `ifile1 ifile2 ofile`

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

Greater then

gt `ifile1 ifile2 ofile`

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

Equal constant

eqc,*c* `ifile ofile`

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

Parameter:

c	FLOAT	Constant

Not equal constant

nec,*c* `ifile ofile`

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

Parameter:

c	FLOAT	Constant

Less equal constant

lec,*c* `ifile ofile`

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

Parameter:

c	FLOAT	Constant

Less then constant

ltc,c `ifile ofile`

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

Parameter:

c FLOAT Constant

Greater equal constant

gec,c `ifile ofile`

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

Parameter:

c FLOAT Constant

Greater then constant

gtc,c `ifile ofile`

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

Parameter:

c FLOAT Constant

2.13 Conditions

A value not equal to zero is treated as "true", zero is treated a "false".

If then

ifthen `ifile1 ifile2 ofile`

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

If not then

ifnotthen `ifile1 ifile2 ofile`

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

If then constant

ifthenc,c `ifile ofile`

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

Parameter:

c FLOAT Constant

If not then constant

ifnotthenc,c `ifile ofile`

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

Parameter:

c FLOAT Constant

2.14 Statistical description of the data

In this program there is the different notion of "mean" and "average" to distinguish two different kinds of treatment of missing values: While computing the mean, only the not missing values are considered to belong to the sample with the side effect of a probably reduced sample size. Computing the average is just adding the sample members and divide the result by the sample size. For example, the mean of 1, 2, miss and 3 is (1+2+3)/3 = 2, whereas the average is (1+2+miss+3)/4 = miss/4 = miss. If there are no missing values in the sample, the average and the mean are identical. In this chapter the abbreviations as in the following table are used:

mean resp. **avg**	$n^{-1} \sum_{i=1}^{n} x_i$
mean resp. **avg** weighted by $\{w_i, i = 1, ..., n\}$	$\left(\sum_{j=1}^{n} w_j \right)^{-1} \sum_{i=1}^{n} w_i \, x_i$
Variance **var**	$n^{-1} \sum_{i=1}^{n} (x_i - \overline{x})^2$
var weighted by $\{w_i, i = 1, ..., n\}$	$\left(\sum_{j=1}^{n} w_j \right)^{-1} \sum_{i=1}^{n} w_i \left(x_i - \left(\sum_{j=1}^{n} w_j \right)^{-1} \sum_{j=1}^{n} w_j \, x_j \right)^2$

Field minimum

fldmin `ifile ofile`

$$o(t,1) = \mathbf{min}\{i(t',x'), t' = t\}$$

Field maximum

fldmax `ifile ofile`

$o(t,1) = \mathbf{max}\{i(t',x'), t' = t\}$

Field sum

fldsum `ifile ofile`

$o(t,1) = \sum_{x} i(t,x)$

Field mean

fldmean `ifile ofile`

$o(t,1) = \mathbf{mean}\{i(t',x'), t' = t\}$

weighted by area weights obtained by the input field.

Field average

fldavg `ifile ofile`

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

weighted by area weights obtained by the input field.

Field standard deviation

fldstd `ifile ofile`

$o(t,1) = \sqrt{\mathbf{var}\{i(t',x'), t' = t\}}$

weighted by area weights obtained by the input field.

Field variance

fldvar `ifile ofile`

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

weighted by area weights obtained by the input field.

Zonal minimum

zonmin `ifile ofile`

For every latitude the minimum over all longitudes is computed.

Zonal maximum

zonmax `ifile ofile`

For every latitude the maximum over all longitudes is computed.

Zonal sum

zonsum `ifile ofile`

For every latitude the sum over all longitudes is computed.

Zonal mean

zonmean `ifile ofile`

For every latitude the mean over all longitudes is computed.

Zonal average

zonavg `ifile ofile`

For every latitude the average over all longitudes is computed.

Zonal standard deviation

zonstd `ifile ofile`

For every latitude the standard deviation over all longitudes is computed.

Zonal variance

zonvar `ifile ofile`

For every latitude the variance over all longitudes is computed.

Meridional minimum

mermin `ifile ofile`

For every longitude the minimum over all latitudes is computed.

Meridional maximum

mermax `ifile ofile`

For every longitude the maximum over all latitudes is computed.

Meridional sum

mersum `ifile ofile`

For every longitude the sum over all latitudes is computed.

Meridional mean

mermean `ifile ofile`

For every longitude the mean over all latitudes is computed.

Meridional average

meravg `ifile ofile`

For every longitude the average over all latitudes is computed.

Meridional standard deviation

merstd `ifile ofile`

For every longitude the standard deviation over all latitudes is computed.

Meridional variance

mervar `ifile ofile`

For every longitude the variance over all latitudes is computed.

Vertical minimum

vertmin `ifile ofile`

For every gridpoint the minimum over all levels is computed.

Vertical maximum

vertmax `ifile ofile`

For every gridpoint the maximum over all levels is computed.

Vertical sum

vertsum `ifile ofile`

For every gridpoint the sum over all levels is computed.

Vertical mean

vertmean `ifile ofile`

For every gridpoint the mean over all levels is computed.

Vertical average

vertavg `ifile ofile`

For every gridpoint the average over all levels is computed.

Vertical standard deviation

vertstd `ifile ofile`

For every gridpoint the standard deviation over all levels is computed.

Time minimum

timmin `ifile ofile`

$o(1,x) = \mathbf{min}\{i(t',x'), x' = x\}$

Time maximum

timmax `ifile ofile`

$o(1,x) = \mathbf{max}\{i(t',x'), x' = x\}$

Time sum

timsum `ifile ofile`

$o(1,x) = \sum_{t=1}^{n} i(t',x)$

Time mean

timmean `ifile ofile`

$o(1,x) = \mathbf{mean}\{i(t',x'), x' = x\}$

Time average

timavg `ifile ofile`

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

Time standard deviation

timstd `ifile ofile`

$o(1,x) = \sqrt{\mathbf{var}\{i(t',x'), x' = x\}}$

Hourly minimum

hourmin `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same hour, it is

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

Hourly maximum

hourmax `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same hour, it is

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

Hourly sum

hoursum `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same hour, it is

$o(t,x) = \sum_{t=1}^{n} i(t',x)$

Hourly mean

hourmean `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same day, it is

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

Hourly average

houravg `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same day, it is

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

Hourly standard deviation

hourstd `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same hour, it is

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

Daily minimum

daymin `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same day, it is

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

Daily maximum

daymax `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same day, it is

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

Daily sum

daysum `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same month, it is

$o(t, x) = \sum_{t=1}^{n} i(t', x)$

Daily mean

daymean `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same day, it is

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

Daily average

dayavg `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same day, it is

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

Daily standard deviation

daystd `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same day, it is

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

Monthly minimum

monmin `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same month, it is

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

Monthly maximum

monmax `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same month, it is

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

Monthly sum

monsum `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same month, it is

$o(t, x) = \sum_{t=1}^{n} i(t', x)$

Monthly mean

monmean `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same month, it is

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

Monthly average

monavg `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same month, it is

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

Monthly standard deviation

monstd `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same month, it is
$o(t, x) = \sqrt{\mathbf{var}\{i(t', x), t_1 < t' \le t_n\}}$

Yearly minimum

yearmin `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same year, it is
$o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \le t_n\}$

Yearly maximum

yearmax `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same year, it is
$o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \le t_n\}$

Yearly sum

yearsum `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same year, it is
$o(t, x) = \sum_{t=1}^{n} i(t', x)$

Yearly mean

yearmean `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same year, it is
$o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \le t_n\}$

Yearly average

yearavg `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same year, it is
$o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \le t_n\}$

Yearly standard deviation

yearstd `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of field of the same year, it is
$o(t, x) = \sqrt{\mathbf{var}\{i(t', x), t_1 < t' \le t_n\}}$

Seasonally minimum

seasmin `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year and season, where december belongs to the northern hemispheric winter of the next year, it is

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

Be careful about the first and the last timestep, they may be incorrect DJF values.

Seasonally maximum

seasmax `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year and season, where december belongs to the northern hemispheric winter of the next year, it is

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

Be careful about the first and the last timestep, they may be incorrect DJF values.

Seasonally sum

seassum `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year and season, where december belongs to the northern hemispheric winter of the next year, it is

$o(t, x) = \sum_{t=1}^{n} i(t', x)$

Be careful about the first and the last timestep, they may be incorrect DJF values.

Seasonally mean

seasmean `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year and season, where december belongs to the northern hemispheric winter of the next year, it is

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

Be careful about the first and the last timestep, they may be incorrect DJF values.

Seasonally average

seasavg `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year and season, where december belongs to the northern hemispheric winter of the next year, it is

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

Be careful about the first and the last timestep, they may be incorrect DJF values.

Seasonally standard deviation

seasstd `ifile ofile`

For every adjacent sequence $t_1, ..., t_n$ of timesteps of the same year and season, where december belongs to the northern hemispheric winter of the next year, it is

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

Be careful about the first and the last timestep, they may be incorrect DJF values.

Multi-year daily minimum

ydaymin `ifile ofile`

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

Multi-year daily maximum

ydaymax `ifile ofile`

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

Multi-year daily mean

ydaymean `ifile ofile`

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

Multi-year daily average

ydayavg `ifile ofile`

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

Multi-year daily standard deviation

ydaystd `ifile ofile`

$$\begin{aligned} o(001,x) &= \sqrt{\mathbf{var}\{i(t,x), \mathrm{day}(i(t)) = 001\}} \\ &\vdots \\ o(366,x) &= \sqrt{\mathbf{var}\{i(t,x), \mathrm{day}(i(t)) = 366\}} \end{aligned}$$

Multi-year monthly minimum

ymonmin `ifile ofile`

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

Multi-year monthly maximum

ymonmax `ifile ofile`

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

Multi-year monthly mean

ymonmean `ifile ofile`

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

Multi-year monthly average

ymonavg `ifile ofile`

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

Multi-year monthly standard deviation

ymonstd `ifile ofile`

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

Multi-year seasonally minimum

yseasmin `ifile ofile`

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

Multi-year seasonally maximum

yseasmax `ifile ofile`

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

Multi-year seasonally mean

yseasmean `ifile ofile`

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

Multi-year seasonally average

yseasavg `ifile ofile`

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

Multi-year seasonally standard deviation

yseasstd `ifile ofile`

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

Running minimum

runmin,*nts* `ifile ofile`

$$o(t,x) = \textbf{min}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}$$

Parameter:

nts INTEGER Number of timesteps

Running maximum

runmax,*nts* `ifile ofile`

$$o(t,x) = \textbf{max}\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}$$

Parameter:

nts INTEGER Number of timesteps

Running sum

runsum,*nts* `ifile ofile`

$o(t,x) = \sum\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}$

Parameter:

nts INTEGER Number of timesteps

Running mean

runmean,*nts* `ifile ofile`

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

Parameter:

nts INTEGER Number of timesteps

Running average

runavg,*nts* `ifile ofile`

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

Parameter:

nts INTEGER Number of timesteps

Running standard deviation

runstd,*nts* `ifile ofile`

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

Parameter:

nts INTEGER Number of timesteps

2.15 Regression

Detrend

detrend `ifile ofile`

Every time series in `ifile` is linearly detrended. For every field element x only those timesteps

t belong to the sample $S(x)$, which have $i(t,x) \neq \text{miss}$. With

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

and

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

it is

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

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

Trend

trend `ifile ofile1 ofile2`

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

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

and

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

Thus the estimation for a is stored in `ofile1` and that for b is stored in `ofile2`. To subtract the trend from the data see operator subtrend.

Subtract trend

subtrend `ifile1 ifile2 ifile3 ofile`

This operator is for subtracting a trend computed by the operator trend.
It is

$$o(t,x) = i_1(t,x) - (i_2(1,x) + i_3(1,x) \cdot t)$$

2.16 Interpolation

Conservative remapping

remapcon,*grid* `ifile ofile`

SCRIP first order conservative remapping.
Parameter:

grid	STRING	Grid description file or name of the target grid

Environment:

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

Bilinear interpolation

remapbil,*grid* `ifile ofile`

SCRIP bilinear interpolation (only rectangular grids).
Parameter:

grid `STRING` Grid description file or name of the target grid

Bicubic interpolation

remapbic,*grid* `ifile ofile`

SCRIP bicubic interpolation (only rectangular grids).
Parameter:

grid `STRING` Grid description file or name of the target grid

Distance-weighted averaging

remapdis,*grid* `ifile ofile`

SCRIP distance-weighted average of the four nearest neighbor values.
Parameter:

grid `STRING` Grid description file or name of the target grid

Interpolate

interpolate,*grid* `ifile ofile`

Grid interpolation from PINGO.
Parameter:

grid `STRING` Grid description file or name of the target grid

Grid interpolation

intgrid,*grid* `ifile ofile`

Linear grid interpolation.
Parameter:

grid `STRING` Grid description file or name of the target grid

Point interpolation

intpoint,*long*,*lat* `ifile ofile`

Linear point interpolation.
Parameter:

long	FLOAT	Longitude of the point
lat	FLOAT	Latitude of the point

Model to pressure level interpolation

ml2pl,*levels* `ifile ofile`

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

levels	FLOAT	Pressure levels in pascal

Environment:

EXTRAPOLATE

If set to 1 extrapolate missing values.

Model to height level interpolation

ml2hl,*levels* `ifile ofile`

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

levels	FLOAT	Height levels in meter (max level: 65535 m)

Environment:

EXTRAPOLATE

If set to 1 extrapolate missing values.

Time interpolation

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

Linear interpolation between time steps.
Parameter:

date	INTEGER	Start date (e.g. 19780130)
time	INTEGER	Start time (e.g. 1800)
inc	STRING	Optional increment (e.g. 12hour) [default: 0hour]

Year interpolation

intyear,*years* `ifile1 ifile2 ofile`

Linear interpolation between two years.
Parameter:

years INTEGER Comma separated list of years

2.17 Spectral transformation

Spectral to gridpoint

sp2gp `ifile ofile`

Convert all spectral fields to gaussian gridpoint.

Gridpoint to spectral

gp2sp `ifile ofile`

Convert all gaussian gridpoint fields to spectral.

Spectral to spectral

sp2sp,*trunc* `ifile ofile`

Change truncation of all spectral fields. The operator performs downward conversion by cutting the resolution. Upward conversions are achieved by filling in zeros.
Parameter:

trunc INTEGER New spectral resolution

Cut spectral wave number

spcut,*wnums* `ifile ofile`

Set the user defined wave numbers to zero.
Parameter:

wnums INTEGER Comma separated list of wave numbers

2.18 Other

GrADS data descriptor file

gradsdes `ifile`

Creates a GrADS data descriptor file. Supported file formats are GRIB, SERVICE and EXTRA. For GRIB files the GrADS index file is also generated. For SERVICE and EXTRA files the grid musst be specified with the CDO option '-g <grid>'. This operator takes `ifile` in order to create filenames for the descriptor (ifile.ctl) and the index (ifile.gmp) file.

Mass stream function

mastrfu `ifile ofile`

Compute the mass stream function (ECHAM code272). The input field must be a zonal mean of v-velocity (code132) on pressure levels.

Bibliography

[GRIB] "Guide to WMO Binary Code Form GRIB 1",
John Stackpole, 1994,
http://www.wmo.ch/web/www/WDM/Guides/Guide-binary.html

[netCDF] "NetCDF User's Guide",
Russ Rew, Glenn Davis, Steve Emmerson, Harvey Davies, 2005,
http://www.unidata.ucar.edu/packages/netcdf/index.html

[PINGO] "The PINGO package",
Juergen Waszkewitz, Peter Lenzen, Nathan Gillet, 1996,
http://www.mad.zmaw.de/Pingo/post/down/BigPingo.pdf

[SCRIP] "A User's Guide for SCRIP",
Philip W. Jones, 1998,
http://climate.lanl.gov/Software/SCRIP/SCRIPusers.pdf

Operator catalog

Information

info	File information
infov	File information
map	Print simple map
sinfo	Short file information
sinfov	Short file information
diff	Differences of two files
diffv	Differences of two files
nyear	Number of years
nmon	Number of months
ndate	Number of dates
ntime	Number of timesteps
ncode	Number of codes
nvar	Number of variables
nlevel	Number of levels
showyear	Show years
showmon	Show months
showdate	Show dates
showtime	Show timesteps
showcode	Show codes
showvar	Show variable names
showlevel	Show levels
vardes	Variable description
griddes	Grid description
vct	Vertical coordinate table

File operations

copy	Copy files
cat	Concatenate files
merge	Merge files
splitcode	Split codes
splitvar	Split variables
splitlevel	Split levels
splitgrid	Split grids
splitzaxis	Split zaxis
splithour	Split hours
splitday	Split days
splitmon	Split months
splitseas	Split seasons

splityear	Split years
splitrec	Split records

Formatted I/O

output	ASCII output
outputint	Integer output
outputsrv	SERVICE output
outputext	EXTRA output

Generation of variables

const	Constant variable
random	Variable with random values
vardup	Duplicate variables
varmul	Multiply variables

Manipulating the header/field

setpartab	Set parameter table
setcode	Set code
setvar	Set variable name
setdate	Set date
settime	Set time
setday	Set day
setmon	Set month
setyear	Set year
settunits	Set time units
settaxis	Set time axis
setreftime	Set reference time
shifttime	Shift time steps
chcode	Change code
setgrid	Set grid
setgridtype	Set grid type
setgatt	Set global attribute
setgatts	Set global attributes
invertlat	Invert latitude
invertlon	Invert longitude
invertlatdes	Invert latitude decription
invertlondes	Invert longitude decription
invertlatdata	Invert latitude data
invertlondata	Invert longitude data

Selection

selcode	Select codes
delcode	Delete codes
selvar	Select variables
delvar	Delete variables
sellevel	Select levels
selgrid	Select grids
selzaxis	Select zaxis
selrec	Select records
seltimestep	Select timesteps
seltime	Select times
selhour	Select hours
selday	Select days
selmon	Select months
selseas	Select seasons
selyear	Select years
seldate	Select dates
sellonlatbox	Select lon/lat box
selindexbox	Select index box

Missing values

setmissval	Set a new missing value
setctomiss	Set constant to missing value
setmisstoc	Set missing value to constant
setrtomiss	Set range to missing value

Sorting

sortcode	Sort by code number
sortvar	Sort by variable name
sortlevel	Sort by level
timsort	Sort over the time

Arithmetic processor

expr	Evaluate expressions
exprf	Evaluate expressions from script file

Arithmetic

addc	Add by constant
subc	Subtract by constant
mulc	Multiply by constant
divc	Divide by constant

add	Add two fields
sub	Subtract two fields
mul	Multiply two fields
div	Divide two fields
min	Minimum of two fields
max	Maximum of two fields
ymonadd	Add multi-year monthly time averages
ymonsub	Subtract multi-year monthly time averages
ymonmul	Multiply multi-year monthly time averages
ymondiv	Divide multi-year monthly time averages

Mathematical functions

sqr	Square
sqrt	Square root
exp	Exp
log	Logarithm
log10	Logarithm base 10
sin	Sine
cos	Cosine
tan	Tangent
asin	Arcus sine
acos	Arcus cosine
atan	Arcus tangent

Comparisons

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

Conditions

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

Statistical description of the data

fldmin	Field minimum
fldmax	Field maximum
fldsum	Field sum
fldmean	Field mean
fldavg	Field average
fldstd	Field standard deviation
fldvar	Field variance
zonmin	Zonal minimum
zonmax	Zonal maximum
zonsum	Zonal sum
zonmean	Zonal mean
zonavg	Zonal average
zonstd	Zonal standard deviation
zonvar	Zonal variance
mermin	Meridional minimum
mermax	Meridional maximum
mersum	Meridional sum
mermean	Meridional mean
meravg	Meridional average
merstd	Meridional standard deviation
mervar	Meridional variance
vertmin	Vertical minimum
vertmax	Vertical maximum
vertsum	Vertical sum
vertmean	Vertical mean
vertavg	Vertical average
vertstd	Vertical standard deviation
timmin	Time minimum
timmax	Time maximum
timsum	Time sum
timmean	Time mean
timavg	Time average
timstd	Time standard deviation
hourmin	Hourly minimum
hourmax	Hourly maximum
hoursum	Hourly sum
hourmean	Hourly mean
houravg	Hourly average
hourstd	Hourly standard deviation
daymin	Daily minimum
daymax	Daily maximum
daysum	Daily sum
daymean	Daily mean
dayavg	Daily average
daystd	Daily standard deviation

monmin	Monthly minimum
monmax	Monthly maximum
monsum	Monthly sum
monmean	Monthly mean
monavg	Monthly average
monstd	Monthly standard deviation
yearmin	Yearly minimum
yearmax	Yearly maximum
yearsum	Yearly sum
yearmean	Yearly mean
yearavg	Yearly average
yearstd	Yearly standard deviation
seasmin	Seasonally minimum
seasmax	Seasonally maximum
seassum	Seasonally sum
seasmean	Seasonally mean
seasavg	Seasonally average
seasstd	Seasonally standard deviation
ydaymin	Multi-year daily minimum
ydaymax	Multi-year daily maximum
ydaymean	Multi-year daily mean
ydayavg	Multi-year daily average
ydaystd	Multi-year daily standard deviation
ymonmin	Multi-year monthly minimum
ymonmax	Multi-year monthly maximum
ymonmean	Multi-year monthly mean
ymonavg	Multi-year monthly average
ymonstd	Multi-year monthly standard deviation
yseasmin	Multi-year seasonally minimum
yseasmax	Multi-year seasonally maximum
yseasmean	Multi-year seasonally mean
yseasavg	Multi-year seasonally average
yseasstd	Multi-year seasonally standard deviation
runmin	Running minimum
runmax	Running maximum
runsum	Running sum
runmean	Running mean
runavg	Running average
runstd	Running standard deviation

Regression

detrend	Detrend
trend	Trend
subtrend	Subtract trend

Interpolation

remapcon	Conservative remapping
remapbil	Bilinear interpolation
remapbic	Bicubic interpolation
remapdis	Distance-weighted averaging
interpolate	Interpolate
intgrid	Grid interpolation
intpoint	Point interpolation
ml2pl	Model to pressure level interpolation
ml2hl	Model to height level interpolation
inttime	Time interpolation
intyear	Year interpolation

Spectral transformation

sp2gp	Spectral to gridpoint
gp2sp	Gridpoint to spectral
sp2sp	Spectral to spectral
spcut	Cut spectral wave number

Other

gradsdes	GrADS data descriptor file
mastrfu	Mass stream function

Operator index

N

O

R

S

T

V

Y

Z