CDO - advanced data operations

Ralf Müller
Uwe Schulzweida
Luis Kornblueh
Karl-Hermann Wieners
Oliver Heidmann

MPI Met

29. September 2015

Max-Planck-Institut
für Meteorologie

Overview

Operations

Hundreds of operators for selection, comparison, arithmetic functions, statistical analysis, regression, interpolation, meta data processing, compression, plotting, ...

Supported file formats

netCDF3/4, GRIB1, GRIB2 (grib_api), MPIMET: SERVICE, EXTRA and IEG including multiple output precisions

Supported Platforms

- POSIX Compatiple: AIX, Super-UX, Linux, BSD
- Windows: 32bit (mingw32, limited functionality), 64bit (cygwin, full functionality)

Homepage

https://code.zmaw.de/projects/cdo

Institut
logie

Main Feature: One Rule to Combine them all!

Operator Chaining

Operators can be combined with '-' on the command line
$\longrightarrow$ running in parallel

```
cdo -f nc -setunit,'m/s' \
  -setname,velocity \
  -sqrt \
    -add \
       -mul -selname,u $ifile -selname,u $ifile \
       -mul -selname,v $ifile -selname,v $ifile \
   $ofile
```

Main Feature: One Rule to Combine them all!

Operator Chaining

Operators can be combined with '-' on the command line
$\longrightarrow$ running in parallel

```
cdo -f nc -setunit,'m/s' \
  -setname,velocity \
  -sqrt \
    -add \
      -mul -selname,u $ifile -selname,u $ifile \
      -mul -selname,v $ifile -selname,v $ifile \
   $ofile
```

```
cdo \
  -div \
    -addc,273.15 -select,name=temp $ifile0 \
    -mul \
      -gtc,1035.0 -selname,rho $ifile1 \
      -ltc,1038.0 -selname,rho $ifile1 \
   $ofile
```

Main Feature: One Rule to ... let them share something?

Shared Memory Parallelisation

- Smallest IO unit is a *record*: one horizontal field - like a GRIB record
- Output stream of right operator is input stream of left operator
- data read/write is synchronized with pthread

Max-Planck-Institut
für Meteorologie

Main Feature: One Rule to ... let them share something?

Shared Memory Parallelisation

- Smallest IO unit is a *record*: one horizontal field - like a GRIB record
- Output stream of right operator is input stream of left operator
- data read/write is synchronized with pthread

What's the benefit?

- Huge files can be processed as long as a single record fits into memory
- No need for temporary files
- Users can write their own operations based on existing ones
- Other parallelisation techniques can be use on top or below: File splitting, OpenMP

Max-Planck-Institut
für Meteorologie

Highlights: Usefull options - Part I

Get help

-h [operator]	get help for given operator or module
-V	information about the CDO binary

Highlights: Usefull options - Part I

Get help

-h [operator]	get help for given operator or module
-V	information about the CDO binary

Set output format

-f grb/grb2/nc/nc2/nc4/nc4c/srv/ext/ieg

```
Climate Data Operators version 1.6.9 (http://mpimet.mpg.de/cdo)
Compiled: by ram on luthien (x86_64-unknown-linux-gnu) Jun 26 2015 14:42:31
Compiler: gcc -g -O3 -std=gnu99 -Wall -fopenmp -march=native
 version: gcc (GCC) 5.1.0
Features: PTHREADS OpenMP4 NC4/HDF5 OPeNDAP SZ Z UDUNITS2 PROJ.4 FFTW3 AVX2
Libraries: proj/4.91
Filetypes: srv ext ieg grb grb2 nc nc2 nc4 nc4c
     CDI library version : 1.6.9 of Jun 26 2015 14:42:11
 CGRIBEX library version : 1.7.2 of Apr 22 2015 13:44:04
GRIB_API library version : 1.13.1
  netCDF library version : 4.3.3.1 of Mar 12 2015 14:13:12 $
    HDF5 library version : 1.8.14
 SERVICE library version : 1.3.2 of Jun 26 2015 14:42:09
   EXTRA library version : 1.3.2 of Jun 26 2015 14:42:14
     IEG library version : 1.3.3 of Jun 26 2015 14:42:14
    FILE library version : 1.8.2 of Jun 26 2015 14:42:13
```

Max-Planck-Institut für Meteorologie

Highlights: Usefull options - Part II

Run multiuple OpenMP threads

`-P <threads>`

OpenMP is mostly used in horizontal interpolation, ensemble analysis, filtering and eofs

Set netcdf header size

`--hdr_pad <numberOfBytes>`

If the memory dedicated to data definitions is large enough, meta information can be changed *without* rewriting the data. *[netcdf only]*

Set output precision

`-b <numberOfBits>`

Possible values are `I8/I16/I32/F32/F64` for nc/nc2/nc4/nc4c

`P1 - P24` for grb/grb2

Max-Planck-Institut
für Meteorologie

Highlights - GRIB2 decoding

Use the *copy* operator and desired output type

```
cdo -f nc copy input.grb2 output.nc
```

```
File format : GRIB2
 -1 : Institut Ttype     Levels Points Dtype : Parameter name
  1 : DWD      instant         1  65160  P16  : prmsl
  2 : DWD      accum           1  65160  P16  : sshf
 ...
 21 : DWD      instant         1  65160  P16  : NCRAIN
```

```
File format : netCDF
 -1 : Institut Ttype     Levels Points Dtype : Parameter name
  1 : DWD      instant         1  65160  F32  : prmsl
  2 : DWD      instant         1  65160  F32  : sshf
 ...
 21 : DWD      instant         1  65160  F32  : NCRAIN
```

... but

·Institut
ologie

Highlights - GRIB2 decoding

Use the *copy* operator and desired output type

```
cdo -f nc copy input.grb2 output.nc
```

```
File format : GRIB2
 -1 : Institut Ttype     Levels Points Dtype : Parameter name
  1 : DWD      instant         1  65160  P16  : prmsl
  2 : DWD      accum           1  65160  P16  : sshf
 ...
 21 : DWD      instant         1  65160  P16  : NCRAIN
```

```
File format : netCDF
 -1 : Institut Ttype     Levels Points Dtype : Parameter name
  1 : DWD      instant         1  65160  F32  : prmsl
  2 : DWD      instant         1  65160  F32  : sshf
 ...
 21 : DWD      instant         1  65160  F32  : NCRAIN
```

... but

... results depend on the grib_api library installation

·Institut
ologie

Highlights - GRIB2 encoding

Back to the initial format

```
cdo -f grb2 output.nc FromGrib2ToNcToGrib2.grb2
```

```
File format : GRIB2
 -1 : Institut Ttype     Levels Points Dtype : Parameter name
  1 : DWD      instant         1  65160  F32  : prmsl
  2 : DWD      instant         1  65160  F32  : SHFL_S
 ...
 21 : DWD      instant         1  65160  F32  : NCRAIN
```

Compare original and transformed grib2 files ... slighty perfect

```
-1 : Institut Ttype     Levels Points Dtype : Parameter ID
 1 : DWD      instant         1  65160  F32  : 1.3.0
 2 : DWD      instant         1  65160  F32  : 11.0.0
21 : DWD      instant         1  65160  F32  : 216.1.0
```

```
 -1 : Institut Ttype     Levels Points Dtype : Parameter ID
  1 : DWD      instant         1  65160  P16  : 1.3.0
  2 : DWD      accum           1  65160  P16  : 11.0.0
 21 : DWD      instant         1  65160  P16  : 216.1.0
```

Highlights - fine tuned data conversion

How to convert meta data of variables in a single step

setpartabn and *setpartabp* allow meta data transformations based on a fortran namelist syntax:

```
&parameter
   name              = topo
   out_name          = topography
   standard_name     = surface_height
   units             = "cm"
/
```

Other transformation keys are: long_name, missing_value, type, valid_min, factor, delete, convert, ...

CDO call looks like

```
cdo setpartabn,<tableFile>[,convert] <ifile> <ofile>
```

Unitconvertsion is done with UDUNITS2. Parameter tables of existing files can be created with the *partab* operator.

Highlights - formulars with *expr*

Provide formulars as string on the command line:

```
cdo -expr,'T=T+271.15' tempInK.nc tempInC.nc
```

Support for math.h and Array functions

sin, cos, tanh, sqrt,log, exp, asin, gamma, min, max, sum, avg, mean, std, var

Max-Planck-Institut für Meteorologie

Highlights - formulars with *expr*

Provide formulars as string on the command line:

```
cdo -expr,'T=T+271.15' tempInK.nc tempInC.nc
```

Support for math.h and Array functions

sin, cos, tanh, sqrt,log, exp, asin, gamma, min, max, sum, avg, mean, std, var

Possible replacement for the initial example: Absolute Velocity computation

```
cdo \
  -setname,velocity \
  -setunit,'m/s' \
  -expr,'vel=sqrt(u*u+v*v)' $ifile \
   $ofile
```

Borrowed from NCO's *ncap*.

Max-Planck-Institut
für Meteorologie

Highlights - more complex expressions

Mask valued expressions

== ,!= ,< ,<= ,>= ,> ,<=> ,&& ,|| ,?: (ternary operator)

```
cdo -f nc \
  -setmisstonn \
  -sellonlatbox,-12,10,40,62 \
  -aexpr,'P=1013.25*exp(-1.602769777072154*log((exp(topo
      /10000.0)*213.15+75.0)/288.15));T=213.0+75.0*exp((-1)*
      topo/10000.0)-273.15;' \
  -expr,'topo=((topo>=0.0))?topo:(topo/0.0)'   \
  -remapbic,r1440x720 \
  -topo surfTemp_if.nc
```

expr vs. *aexpr*

aexpr performas a copy on all input fields to the output stream and appends the computaion results to it. *expr* writes comuted fields only

Max-Planck-Institut
für Meteorologie

Highlights - more complex expressions

Mask valued expressions

== ,!= ,< ,<= ,>= ,> ,<=> ,&& ,|| ,?: (ternary operator)

```
cdo -f nc \
  -setmisstonn \
  -sellonlatbox,-12,10,40,62 \
  -aexpr,'P=1013.25*exp(-1.602769777072154*log((exp(topo
      /10000.0)*213.15+75.0)/288.15));T=213.0+75.0*exp((-1)*
      topo/10000.0)-273.15;' \
  -expr,'topo=((topo>=0.0))?topo:(topo/0.0)'   \
  -remapbic,r1440x720 \
  -topo surfTemp_if.nc
```

expr vs. *aexpr*

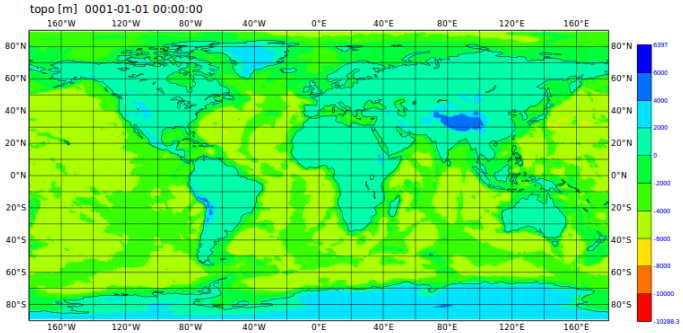
aexpr performas a copy on all input fields to the output stream and appends the computaion results to it. *expr* writes comuted fields only

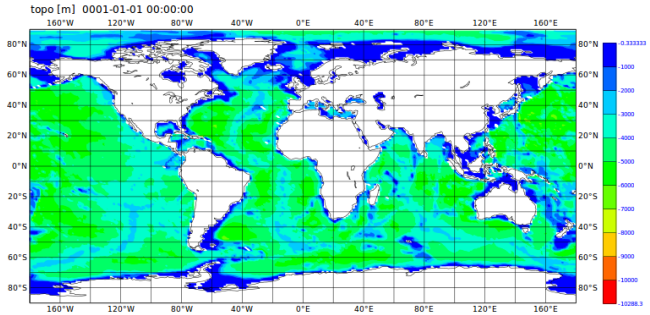
And what if formulars are getting lengthy?

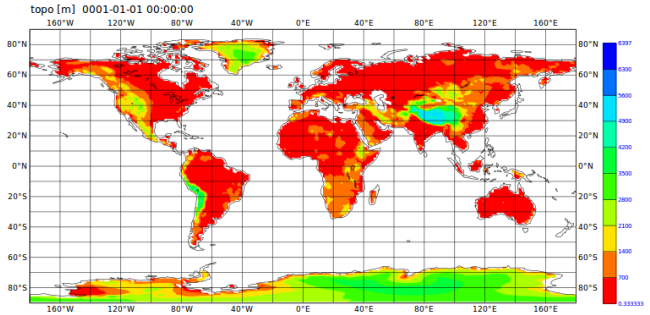
exprf and *aexprf* accept textfile names as arguments from where the formulars will be read in

·Institut
ɔlogie

Highlights: built-in topography with *topo* operator

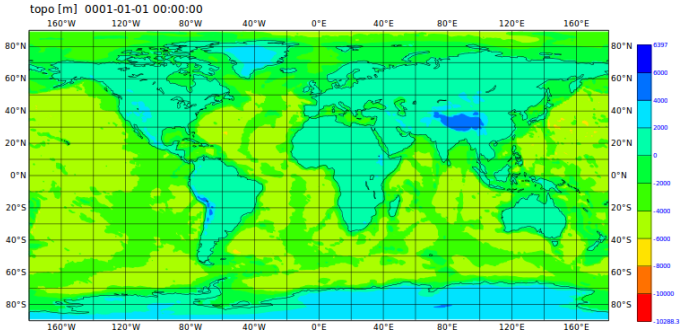


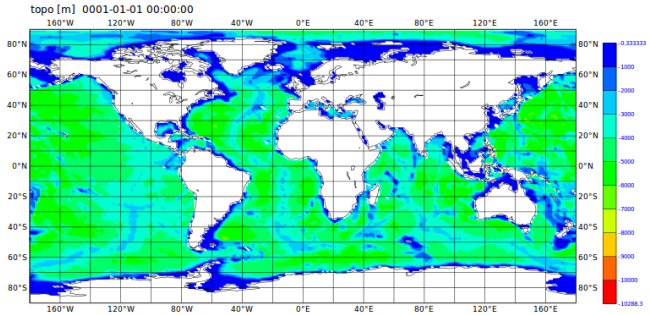


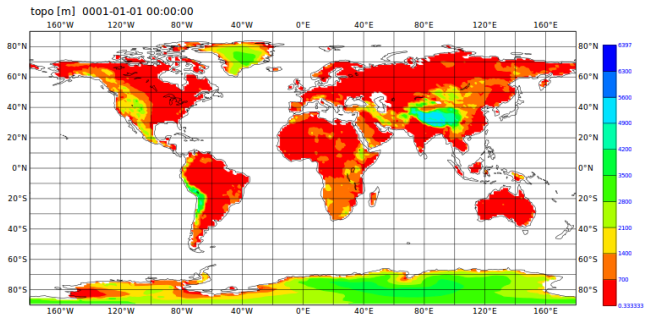


Max-Planck-Institut für Meteorologie

Highlights: built-in topography with *topo* operator



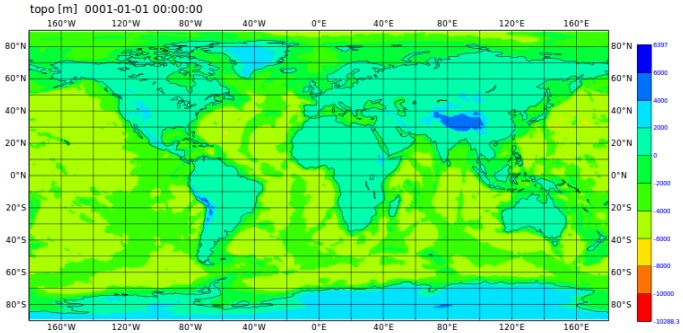




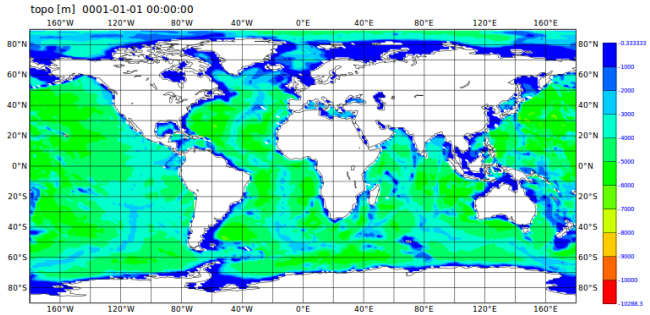

```
cdo -topo topo.grb
```

Max-Planck-Institut für Meteorologie

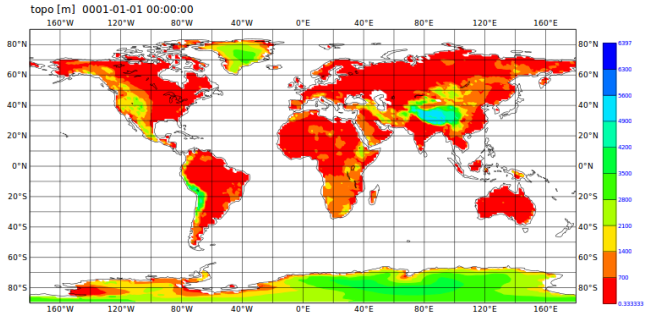
Highlights: built-in topography with *topo* operator




```
cdo -topo topo.grb
```




```
cdo -setrtomiss,0,10000
      -topo topo_ocean.grb
```



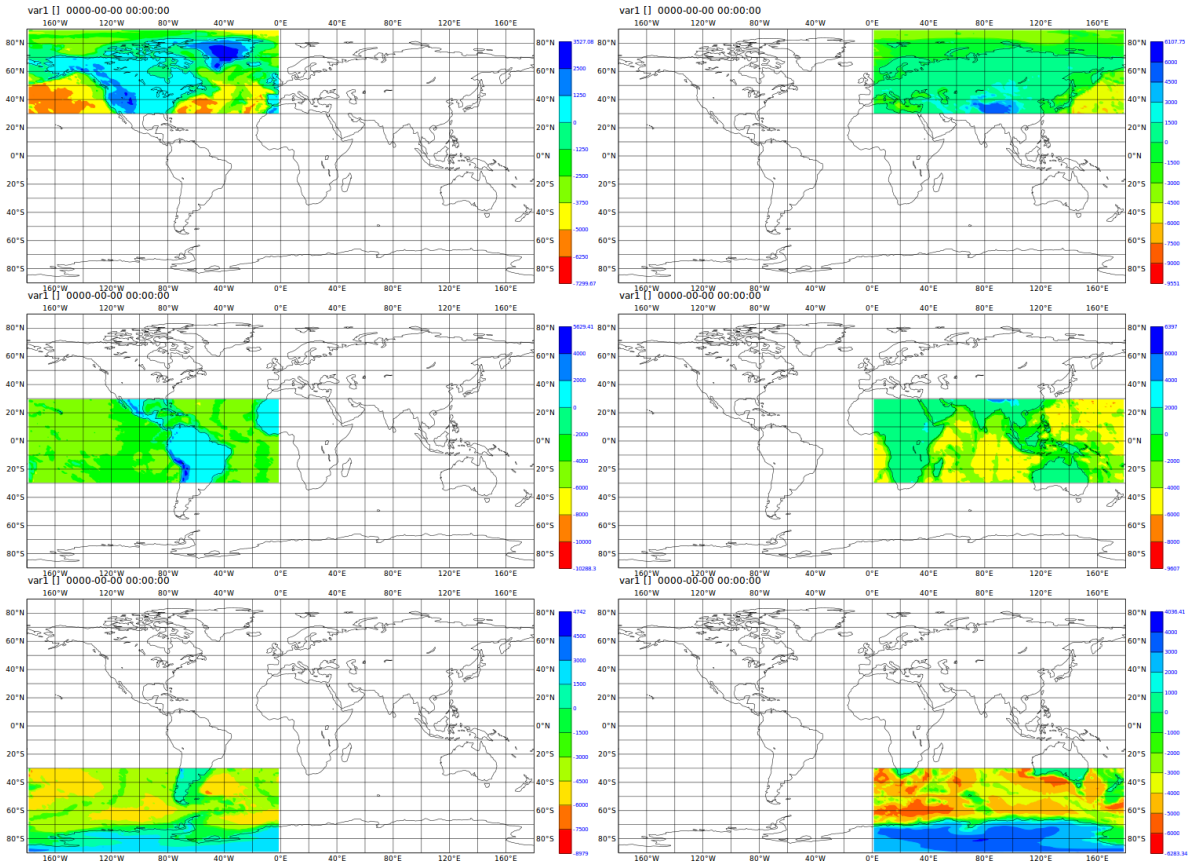

```
cdo -setrtimiss,-20000,0
      -topo topo_land.grb
```

Max-Planck-Institut für Meteorologie

Highlights: Split the grid with *distgrid - collgrid*

Break your regular grid into $n \times m$ parts

cdo -distgrid,2,3 -topo topo_splitted



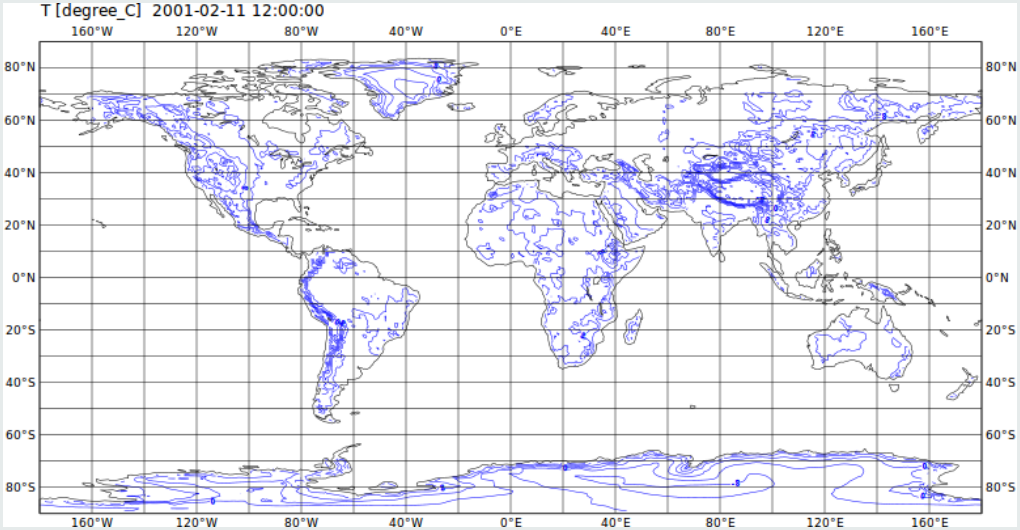
Put your pieces together with

cdo -collgrid topo_splitted*grb collectedtopo.grb

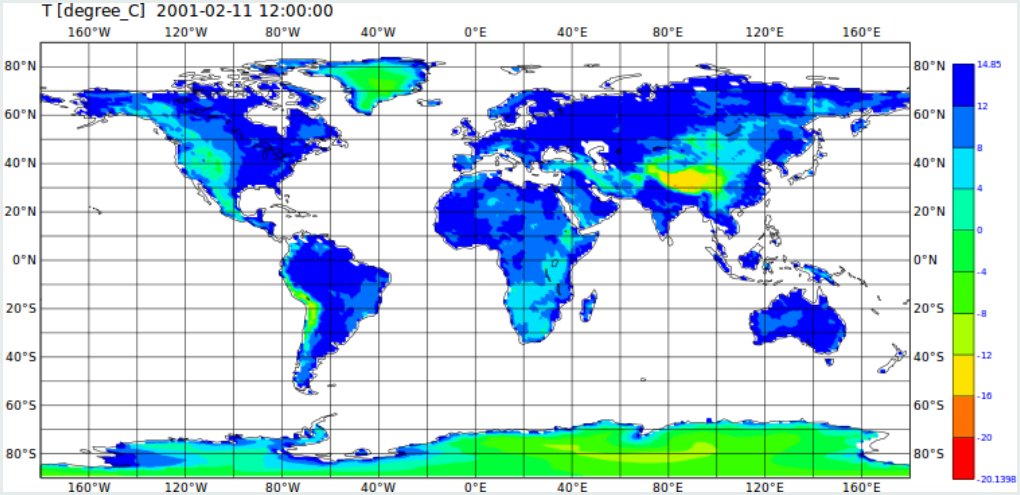
Highlights: Magics++ for plotting ... Watch out PIXAR!

Possible plot types

- *contour*



- *shaded*

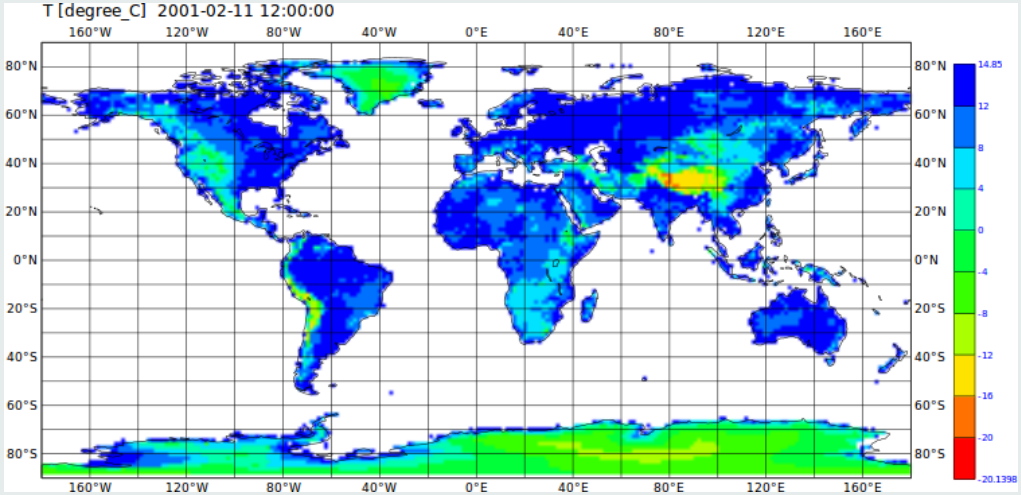


·Institut
ɔlogie

Highlights: Magics++ for plotting ... Watch out PIXAR!

Possible plot types

- coloured cells: *grfill*



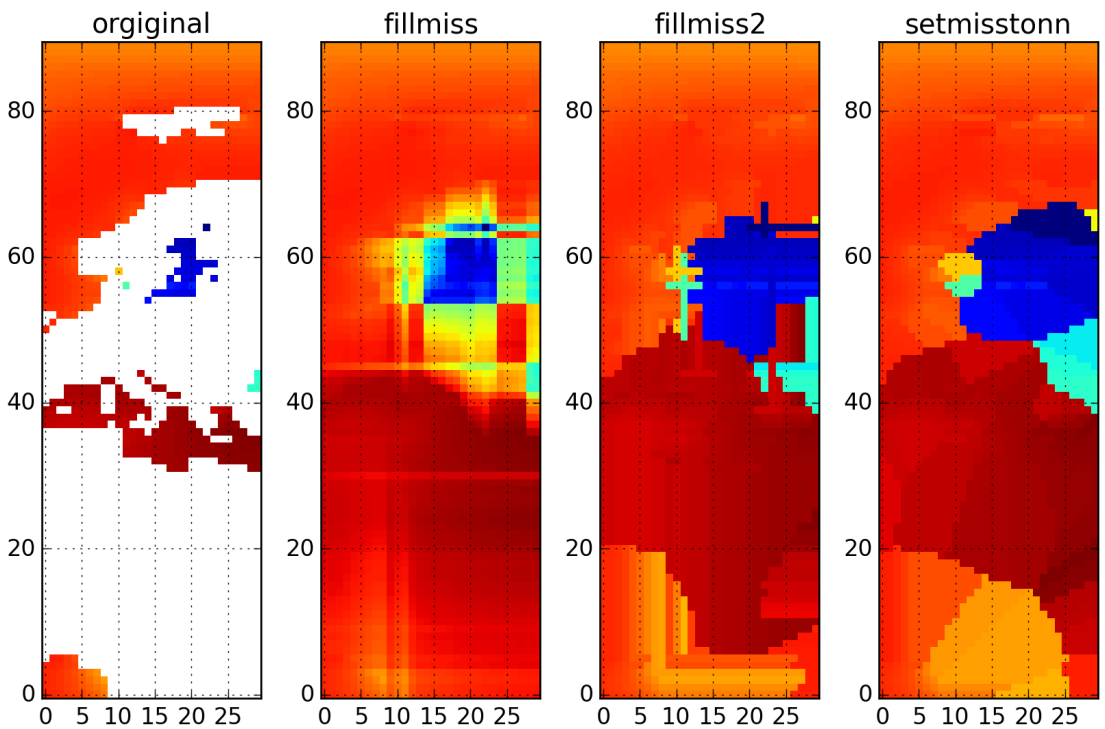
- more: line plots, vectors, animations, output formats: png,svg,ps,pdf,...

Max-Planck-Institut für Meteorologie

Fill missing values

How to overwrite missing data with something reasonable

Model initial data for ocean salinity is on low resolution, usually 1deg.



For higher resolution runs, a simple interpolation could lead to wrong values in the baltic see. Nearest-neighbor interpolation does the trick.

Max-Planck-Institut für Meteorologie

Play the wildcard ... with files

Problem

How to keep the chaining of operators working, when their number of input streams is abitrary? - Polish notation only works for operators with fixed arity

Might not be a problem for operators like *info* or *copy*, but concatenation (*cat*) and merging (*merge*/*mergetime*) would create large temporary data

Max-Planck-Institut für Meteorologie

Play the wildcard ... with files

Problem

How to keep the chaining of operators working, when their number of input streams is abitrary? - Polish notation only works for operators with fixed arity

Might not be a problem for operators like *info* or *copy*, but concatenation (*cat*) and merging (*merge*/*mergetime*) would create large temporary data

... let CDO do the wildcard evaluation

Given single quoted wildcard as input stream, CDO evaluates it into a fixed length list

```
cdo -timmean -cat 'exp004_201?_global.nc*'
```

Max-Planck-Institut für Meteorologie

Play the wildcard ... with variables

Problem

How to select collections of data without explicitly given names or parameters

Max-Planck-Institut
für Meteorologie

Play the wildcard ... with variables

Problem

How to select collections of data without explicitly given names or parameters

... use *select*

CDO's *select* operator accepts wildcards for the 'name' and 'param' key

```
cdo -select,'name=s*' $ifile $ofile
cdo -select,'param=1.?.0' $ifile $ofile
```

Max-Planck-Institut
für Meteorologie

Scripting with Ruby/Python

cdo.{rb,py}

- is a *smart* caller of a CDO binary (with all the pros and cons)
- doesn't need to be re-installed for a new CDO version
- directly bridges your data to the scientific package in Ruby/Python

Max-Planck-Institut
für Meteorologie

Scripting with Ruby/Python

cdo.{rb,py}

- is a *smart* caller of a CDO binary (with all the pros and cons)
- doesn't need to be re-installed for a new CDO version
- directly bridges your data to the scientific package in Ruby/Python
- isn't a shared library, which keeps everything in memory
- doesn't allow write access to files via the numpy or masked arrays

Max-Planck-Institut
für Meteorologie

Scripting with Ruby/Python

cdo.{rb,py}

- is a *smart* caller of a CDO binary (with all the pros and cons)
- doesn't need to be re-installed for a new CDO version
- directly bridges your data to the scientific package in Ruby/Python
- isn't a shared library, which keeps everything in memory
- doesn't allow write access to files via the numpy or masked arrays

homepage: https://code.zmaw.de/projects/cdo/wiki/Cdo{rbpy}
or directly join development at
https://github.com/Try2Code/cdo-bindings

Max-Planck-Institut
für Meteorologie

Usage: Basic - Python 2.7/3.x

Interface examples

```
from cdo import *
cdo = Cdo()

# concatenate list of files, relative time axis
cdo.cat(input = ' '.join(ofiles),
        output = ofile,
        options = '-r')
# vertical interpolation
cdo.intlevel(100,200,500,1000,
             input='Temperatures_L199.grb',
             output='TempOnTargetLevels.grb')
# perform zonal mean after interpolation in nc4 classic
    format
cdo.zonmean(input = "-remapbil,r1400x720 "+myData,
            output = zonmeanFile,
            options = '-P 8 -f nc4c')
```

Max-Planck-Institut
für Meteorologie

Usage: Advanced

return numpy and masked arrays

```
cdo.div(input='salinity.nc landSeaMask.nc',
    returnArray='S')
cdo.copy(input='-div salinity.grb landSeaMask.grb',
    returnMaArray='S', options='-f nc')
```

get cdf handles

```
cdf   = cdo.fldmin(:input => ifile ,:returnCdf => true)
tData = cdf.variables['T'][:]
```

conditional output: no execution if output file is present

```
cdo.forceOutput = False #or
cdo.operator(....., force=False)
```

-Institut
für Meteorologie

Usage: Parallelism with Python

Beyond the shell

```
def grepYear(ifiles,year):
  yearFiles = []
  for ifile in ifiles:
    if (year in cdo.showyear(input = ifile).split()):
      yearFiles.append(ifile)
  cdo.cat(input = ' '.join(yearFiles),
          output = yearFile)
```

```
pool      = multiprocessing.Pool(8)
yearFiles = []
for year,files in filesOfYears.iteritems():
    yearFile = pool.apply_async(grepYear, [files,str(year)])
    yearFiles.append([year,yearFile,yearMeanFile])

pool.close()
pool.join()
```

Max-Planck-Institut
für Meteorologie

CDO's Future

Our Plans

- C++ rewrite to get more recent features
- make operators available to models - online processing will get more and more imported with rising resolution
- plugin system
- additional parallelisation techniques: OpenACC, MPI

Max-Planck-Institut für Meteorologie

CDO's Future

Our Plans

- C++ rewrite to get more recent features
- make operators available to models - online processing will get more and more imported with rising resolution
- plugin system
- additional parallelisation techniques: OpenACC, MPI

What feature do YOU need most?

Max-Planck-Institut für Meteorologie

Don't drink and Derive

$$
\begin{aligned}
a &= b \\
a^2 &= ab \\
2a^2 &= a^2 + ab \\
2a^2 - 2ab &= a^2 - ab \\
2a(a-b) &= a(a-b) \\
2a &= a \\
2 &= 1
\end{aligned}
$$

Max-Planck-Institut
für Meteorologie