# COSMO-REA6 Start help

**(for application within UNIX/Linux)**

The following document shall facilitate the use of COSMO-REA6 data fields. The original format of the hourly COSMO-REA6 fields is the grib-format (DWD grib1).

To make it easier to work with the reanalysis data, single parameters, for example temperature or zonal and meridional wind components, are saved in separate files. The variables are saved by day or month, depending on whether it is a 2D or 3D variable. These monthly or daily files are packed with `bzip2` and their size amounts to 1GB and 0.1GB, respectively. After decompression the files come with the extension *.grb and can be handled with the `CDOs`, `eccodes`, or `wgrib`, as is explained in the following.

More information about constant fields like the land sea mask, orography or distribution of soil types are saved in COSMO_REA6_CONST_withOUTsponge.grb. This file also contains the longitude and latitude information for the regular geographical system. COSMO-REA6 is calculated on a rotated longitude-latitude grid. For a comparison of reanalysis data with station observations one has to use the non-rotated coordinates named **RLAT** and **RLON** in the constant grib (or nc) file. In addition, the variable HHL defines the heights over sea-level for the 41 model half-levels. The 3D variables are provided at the model main level, so that the corresponding height over sea level can be computed by the arithmetic mean of the two neighboring model half-level. For example, the lowest layer is restricted by HHL[41] and HHL[40], the level height computed via arithmetic mean amounts to 10m above sea level over the sea. At the sea the six lowest model level heights are 10m, 35m, 69m, 116m, 178m, and 258m. Over land, these values vary with topography.

More information on the COSMO model grid and heights are saved in
http://www.cosmo-model.org/content/model/documentation/core/cosmo_userguide_5.04.pdf

## CDO (Climate data operators)

The `CDOs` provide many operators for selection, modification, and statistical computation of grib and netcdf data.

Download: https://code.mpimet.mpg.de/projects/cdo/files
Documentation: https://code.mpimet.mpg.de/projects/cdo/embedded/cdo_refcard.pdf

### Get Information

It is possible to get information about the grid and the variables within a data file:

```
cdo griddes infile
cdo sinfon infile
```

## Extraction of Subarea

Subarea of the model domain with specific corners in the southwest (lon1,lat1) and northeast (lon2,lat2):

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

## Compute wind speed

The wind speed can be calculated from the horizontal wind components U and V. The calculation of the scalar wind speed can be performed on the rotated as well as on the geographical grid.

```
cdo chname,U,ws -sqrt -add -sqr -selname,U infile1 -sqr -
selname,vwnd infile2 outfile
```

## Select timesteps

Here, days 3, 4, and 5 of the monthly input file are selected:

```
cdo selday,3,4,5 infile outfile
```

## Convert from grib to netCDF format considering the grid

The prerequisite of any conversion is that the metadata variable *timeRangeIndicator* is set equal to zero. This is necessary because otherwise the CDOs would misinterpret other metadata settings. For most of the available parameters (instantaneous values), this is already the default setting. For all averaged, accumulated, and MinMax values, the *timeRangeIndicator* needs to be manually set to zero:

```
grib_set -s timeRangeIndicator=0 infile.grb outfile.grb
```

There are two principally different ways of handling the subsequent conversion into netCDF format.

**1: Projection onto a regular lon-lat grid:**

Here, the original REA6 file is projected onto a regular grid by means of a user specified grid definition file. The grib format is retained.

```
cdo remapcon,outgrid.txt infile.grb outfile.remap.grb
```

In the following example, a conversion into the netCDF 3 format is performed:

```
cdo –f nc remapcon,outgrid.txt infile.grb outfile.nc
```

Another possibility is to convert into the netCDF 4 format that additionally performs a compression of the data. That compression can be as efficient as that of the grib format:

```
cdo –f nc4 –z zip=9 –remapcon,outgrid.txt infile.grb
outfile.nc4
```

The text file `outgrid.txt` contains the properties of the new grid. The example below shows a non-rotated lon-lat grid with 500x500 grid cells, a resolution of about 12km, and the corner in the southwest has the coordinates 30°W and 30°N.

outgrid.txt:

gridtype = lonlat
xsize = 500
ysize = 500
xfirst = -30
yfirst = 30
xinc = 0.11
yinc = 0.11

**2a: Transform to original back-rotated COSMO grid: scalar variables**

This conversion method uses the file griddes_REA6.txt in order to preserve the original COSMO grid structure. The coordinates are changed from the native COSMO grid (rotated pole) to geographical coordinates, i.e., from regular X(lon(x),lat(y)) to irregular X(lon_1(x,y),lat_1(x,y)) with x=1,..,848 and y=1,..,824. For **scalar variables,** the new lon lat values correspond to the effective geographical locations whereas the old lon lat values are valid in rotated coordinates only.

```
cdo -f nc4 -z zip=9 -copy -setgrid,griddes_REA6.txt in.grb out.nc4
```

**2b: Transform to original back-rotated COSMO grid: vector variables**

Vector variables such as near surface wind (U_10M, V_10_M) require additional treatment in order to transform from rotated to geographical coordinates. The file `in.grb` contains both components:

```
cdo -rotuvb,var33,var34 in.grb rotuvb.grb
```

Additionally, wind on model levels (U, V) requires consideration of the staggered COSMO grid:

```
cdo -rotuvb,var33,var34 -uvDestag,var33,var34 in.grb rotuvb.grb
```

The new coordinates are finally set as outlined above (or remap to a regular grid from here):

```
cdo -f nc4 -z zip=9 -copy -setgrid,griddes_REA6.txt rotuvb.grb
out.nc4
```

Now, one can work with the output netCDF file `out.nc4` to some degree. However, with the above instruction the CDOs add in many cases an additional dimension to the data variable, which describes the height of the variable. Also, a *height* and *grid_mapping_1* variable are added, and the name of the data variable is incorrect. The following commands help to clean up the metadata of the netCDF file and delete the additional dimension.

The following commands are based on the NCO package and executables that are provided by it (source code and documentation: http://nco.sourceforge.net/).

The first command renames the generic variable name. Here, the generic variable name is *var11* and the data variable name *T_2M*; **choose appropriately**:

```
ncrename -h -O -v var11,T_2M out.nc4
```

The next command deletes the variable *grid_mapping_1*:

```
ncks -C -h -O -x -v grid_mapping_1 out.nc4 out.nc4
```

This command edits attributes of a variable, in this case the data variable, **adopt appropriately**:

```
ncatted -h -O -a table,T_2M,d,,
              -h -O -a grid_mapping,T_2M,d,,
              -h -O -a coordinates,T_2M,d,,
              -h -O -a standard_name,T_2M,c,c,"air_temperature"
              -h -O -a long_name,T_2M,c,c,"2m air temperature"
              -h -O -a units,T_2M,c,c,"K" out.nc4
```

In many cases, the CDOs add a *height* dimension and variable. This presumably happens if the level variable in the grib-file metadata is set unequal to zero. The following two commands delete the dimension and variable *height*. **Note**: in order to delete the dimension, the complete file needs to be read into memory; hence, a huge amount of memory is needed:

```
ncwa -a height out.nc4 new.nc4

ncks -C -h -O -x -v height new.nc4 out.nc4

rm –fv new.nc4
```

The last step is to correct the global variable *history*, **adopt appropriately**:

```
DATE=$(LANG=en_us_88591; date)
ncatted -h -O -a history,global,d,,
           -O -a history,global,c,c,"$DATE: COSMO-REA6 2m
temperature data converted to netCDF" out.nc4
```

## Compute wind direction

The wind direction itself is a scalar, however, it is calculated from the wind component vectors. Hence, the wind direction needs to be calculated on the grid for which it is intended. Usually, the wind direction is used on the geographical grid. Since the wind components of REA6 are provided on the rotated grid, they need to be rotated back onto the geographical grid first (see above). It should be pointed out here once again that the 2D variables U_10M and V_10M are already *destaggered.* In order to calculate the wind direction from the wind components, the following commands can be applied:

```
cdo atan2 U_10M V_10M winddir
cdo -divc,3.14159265359 winddir o
cdo mulc,180. o winddir
cdo addc,180. winddir o
cdo setvar,winddir o winddir
rm –f o
```

## Grib Api und ecCodes

ecCodes is a software tool, developed by ECMWF (European Centre for Medium range Weather Forecast). It is the next development step of Grip Api and can be used for decoding data in grib and bufr format.

Download and documentation:
https://software.ecmwf.int/wiki/display/ECC/ecCodes+Home

**Examples**:

- Converting grib to netcdf file:

```
grib_to_netcdf –o output.nc input.grib
```

- Listing data input and further information like date, level, name of variables, gridtype, etc.

```
grib_ls input.grb
```

- Finding the next point to a specific longitude-latitude (here: 50.5N, 12.34E). This operator can be used for non-rotated grids only. For application to COSMO-REA6 fields, first a projection via CDO to a non-rotated grid has to be performed.

```
grib_ls –l 50.5,12.34,1 input.grb
```

- Selecting first three timesteps and saving to new file:

```
grib_copy –w count=1/2/3 input.grb output.grb
```

## Wgrib

Wgrib is an old software, however, it is still used.

Download and documentation: http://www.cpc.ncep.noaa.gov/products/wesley/wgrib.html

**Examples:**

- Short listing of inventory (name of variables and record number):

```
wgrib –s gribfile
```

- Converting the complete file to ASCII format:

```
wgrib -d all -text -o outfile.txt gribfile
```

- Selection of a specific variable, here number 20:

```
wgrib –V –d 20 gribfile
```

## Python

For further processing of COSMO-REA6 data with Python, the following library for handling grib data can be used:

```
import pygrib
```

https://pypi.python.org/pypi/pygrib

## R

An easy way to processing REA6 data with R is to convert the grib files into netcdf format first, because available R packages exist which allow reading and handling netcdf files:

```
library(ncdf4)
```

or

```
library(RNetCDF)
```

In order to read grib (1 and 2) files directly, especially the package `gribr` is recommended:

```
library(gribr)
```

However, this package is not available with CRAN but needs to be downloaded from GITHUB and then manually installed. The technical prerequisite is an installation of the ecCodes (see above) because the `gribr` package uses its functionality.

There are more packages for reading and handling grib data in R, for example the following:

```
library(rgdal)
```

## Matlab

Download and documentation of routines for Matlab netcdf/grib reader:

http://www.mathworks.com/matlabcentral/fileexchange/21579-netcdf-grib-reader

## IDL

Download and documentation of routines for IDL:

http://www.harrisgeospatial.com/docs/GRIB_Routines.html