

2022.06.30 Ivanova Kseniya

How to install, compile, configure and run ICON

See Icon wiki page and tutorial here :

[ICONIconModel - MODELS - C2SM Wiki \(ethz.ch\)](https://icon-models.github.io/IconModel/)

[ICON Model Tutorial 2020 \(dwd.de\)](https://www.dwd.de/EN/Research/Modelling/Modelling_Tools/icon/icon_model_tutorial_2020/icon_model_tutorial_2020_node.js.html)

DWD icontools are a repository on the C2SM github: <https://github.com/C2SM/icontools>

First, one clone Icon repository recursively (with all externals submodels, such as , for example, icon-art) from gitHub using SSH protocol:

```
git clone git@github.com:C2SM/icon.git
```

Execute

```
git submodule update --init
```

ICON needs to be configured before compilation. There is a configure-script for each compiler and/or machines located in [config](#) folder in the downloaded locally ICON source code. On daint I configure by running the following in the terminal in the icon source code folder:

```
./config/cscs/daint.cpu.pgi-20.1.1-eccodes
```

Often I got problems with eccodes root, you have to be sure that inside the configure file the provided link exists. For example, if in configure file you provide the following

```
ECCODES_ROOT='/project/g110/spack-install/daint/ec-  
codes/2.19.0/pgi/6btxe4laukde6xhzonqnnzczp6tech7b'
```

Check that the directory

```
'/project/g110/spack-install/daint/eccodes/2.19.0/pgi/6btxe4laukde6xhzonqnnzczp6tech7b/share/ec-  
codes/samples'
```

really exists and containing file GRIB2.tmpl.

After configuring, **Makefile** will be created. Compile with

```
make -j8
```

The **exe** file will be created in the /bin directory in the icon source code folder.

After that, if all steps were successful, you can run icon or icon-art through processing chain using the following command with preparing data

```
python run_chain.py icon-art-BRM 2018-12-21 0 24 -f -j prepare_data icon
```

or without preparing data :

```
python run_chain.py icon-art-BRM 2018-12-21 0 24 -f -j icon
```

All namelists one can find in the run file icon_runjob.cfg . One can change the output namelist as you wish (grib2 output format or netCDF, both are possible) and add/remove output variables as you wish in (**ml_varlist**).

Grib2 output:

In order to write grb output, in configure file we need additional flag

```
EXTRA_CONFIG_ARGS+= ' --disable-mpi-checks --enable-grib2 --without-external-cdi --enable-art'
```

This flag is also enable ART model from externals.

Here are some modifications I did to run Icon with processing chain:

I commented in [prepare_data.py](#)

the following lines in order to avoid problems with lateral boundary conditions :

```
# tools.copy_file(cfg.lateral_boundary_grid,  
#               cfg.lateral_boundary_grid_scratch,  
#               output_log=True)
```

I also changed a bit the file [config.py](#) in /users/kivanova/processing-chain/cases/icon-art-BRM :

-created additional input root with meteoswiss data grid , dictionary etc

```
input_root2 = '/users/kivanova/icon1.config'
```

```
input_root = '/store/empa/em05/input_iconart_processing_chain_example/'
```

Icontool directory I used from Michael Steiner

```
icontools_dir = '/scratch/snx3000/msteiner/spack-stages/daint/spack-stage-icontools-master-
t524rnfa5sfyn4rbvarypyzwae4jg46d/spack-src/icontools'
```

Also, I used the following :

```
radiation_grid_filename = os.path.join(input_root2,
                                       "ICON-1E_DOM01.parent.nc")
dynamics_grid_filename = os.path.join(input_root2, "ICON-1E_DOM01.nc")
map_file_latbc = os.path.join(input_root_grid, "map_file.latbc")
extpar_filename = os.path.join(
    input_root2, "external_parameter_mch_ICON_1E_R19B08_DOM1.nc")
lateral_boundary_grid = "/scratch/snx3000/kivanova/processing_chain/icon-art-
BRM/2018122100_0_24/icon/input/grid/lateral_boundary.grid.grid.nc"
```

And

ART settings-----

```
input_root_tracers = os.path.join(input_root, 'XML')
chemtracer_xml_filename = os.path.join(input_root_tracers,
                                       'tracers_BRM_pntsrc.xml')
pntSrc_xml_filename = os.path.join(input_root_tracers, 'pntSrc_BRM.xml')
art_input_folder = os.path.join(input_root, 'ART')
```

SIMULATION

=====

ICON -----

Executable

```
icon_bin = "/scratch/snx3000/kivanova/icon_src/bin/icon"
#os.path.join(exe_dir, "icon-kit-art_20211018")
```

Namelists and slurm runscript templates

```
icon_runjob = os.path.join(case_dir, 'icon_runjob.cfg')
```

XML files one can find here :

/store/empa/em05/input_iconart_processing_chain_example/XML

In tracers_BRM_pntsrc.xml I changed the units :

mol mol-1 to kg kg-1

The tags "mol_weight" and "unit" are not required for passive tracers. And if they are provided, they have no influence on the tracer or output.

All tracers are treated as mass mixing ratios in the model. Passive tracers are also written out as MMR in the output. These 2 tags are used for chemical tracers in a conversion-routine such that the output is then in VMR, but passive tracers are skipped in this conversion-routine and are written out as MMR.

At the beginning of icon run script icon_runjob.cfg after SBATCH lines I added radiation input files coping it one by one

Link radiation input files

In -sf {cfg.art_input_folder}/runctrl_examples/photo_ctrl/* .

In -sf {cfg.art_input_folder}/runctrl_examples/init_ctrl/* .

In -sf /users/kivanova/icon1.config/ecrad_data/*

In -sf /users/kivanova/icon1.config/ecrad_data/socrates_droplet_scattering_rrtm.nc

In -sf /users/kivanova/icon1.config/ecrad_data/fu_ice_scattering_rrtm.nc

In -sf /users/kivanova/icon1.config/ecrad_data/mcica_gamma.nc

In -sf /users/kivanova/icon1.config/ecrad_data/RADRRTM

In -sf /users/kivanova/icon1.config/ecrad_data/RADSRTM

In -sf /users/kivanova/icon1.config/ecrad_data/slingo_droplet_scattering_rrtm.nc

In -sf /users/kivanova/icon1.config/ecrad_data/yi_ice_scattering_rrtm.nc

In -sf /users/kivanova/icon1.config/ecrad_data/baran_ice_scattering_rrtm.nc

In -sf /users/kivanova/icon1.config/ecrad_data/es_droplet_scattering_rrtm.nc

Fieldextra

In fieldextra it is mandatory to remap variable hsurf since we use tag grid from this variable

&Process in_field = "HSURF", **use_tag="GRID" /**

&Process

```
in_file = "_tmp_u<DDHH>0000"
out_file = "IconPntSr<DDHH>0000", out_type = "GRIB2"
in_regrid_target = "GRID"
tstart = 0, tstop = 24, tincr = 1
/
```

I also changed the levels to 80, as in Meteoswiss

```
levmax = 80
```

and in icon_runjob.cfg in /users/kivanova/processing-chain/cases/icon-art-BRM

```
num_lev = 80      ! number of full levels (atm.) for each domain
```

To run new date with icon art we have to change XML file pntSrc_BRM.xml, config.py in processing chain (output_dir = os.path.join(work_root, casename, '2018041500_0_24', 'icon', 'output') and lateral_boundary_grid = "/scratch/snx3000/kivanova/processing_chain/icon-art-BRM/2018041500_0_24/icon/input/grid/lateral_boundary.grid.grid.nc"), in icon_runjob.cfg name of outputfiles

for Flexpart files we have to change AVAILABLE, COMMAND.

Plot Icon unstructured mesh

To plot Icon unstructured, one can use **Paraview with Reader plugin CDIReader**

(available on ddm06 and on Daint), **psyplot with python**

```
import psyplot.project as psy
```

```
fN = '/project/ivme/MCH-1/icon-art-BRM/icon_output/ICON-ART-OEM_AllVarUnstr_DOM01_00130000.nc'
```

```
ncf = nc.Dataset(fN)
```

```
print(ncf)
```

```
print(ncf.variables.keys())
```

```
ds = xr.open_dataset(fN)
```

```
print(ds)
```

```
print(ds.t_2m.CDI_grid_type)
psy.rcParams['plotter.maps.xgrid'] = False
psy.rcParams['plotter.maps.ygrid'] = False
mpl.rcParams['figure.figsize'] = [10., 8.]
BRM = psy.plot.mapplot(fN, name="testtr12", load=True, maskleq=0,
                      levels=[80],
                      clabel="% (long_name)s",
                      title="% (time)s", cmap='Blues',
                      post="import cartopy.feature as cf; self.ax.add_feature(cf.BORDERS, color='red')",
                      post_timing="replot",
                      enable_post=True)
BRM.update(projection="moll")
BRM.show()
```

Or PolyCollection with python

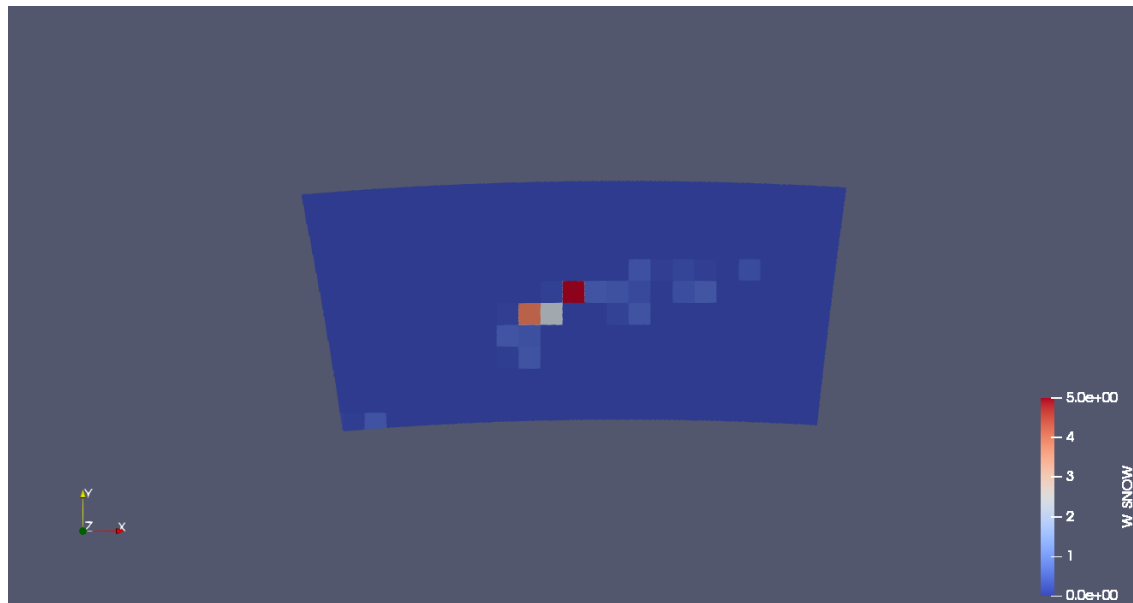
```
from matplotlib.collections import PolyCollection
```

Icontools, problems:

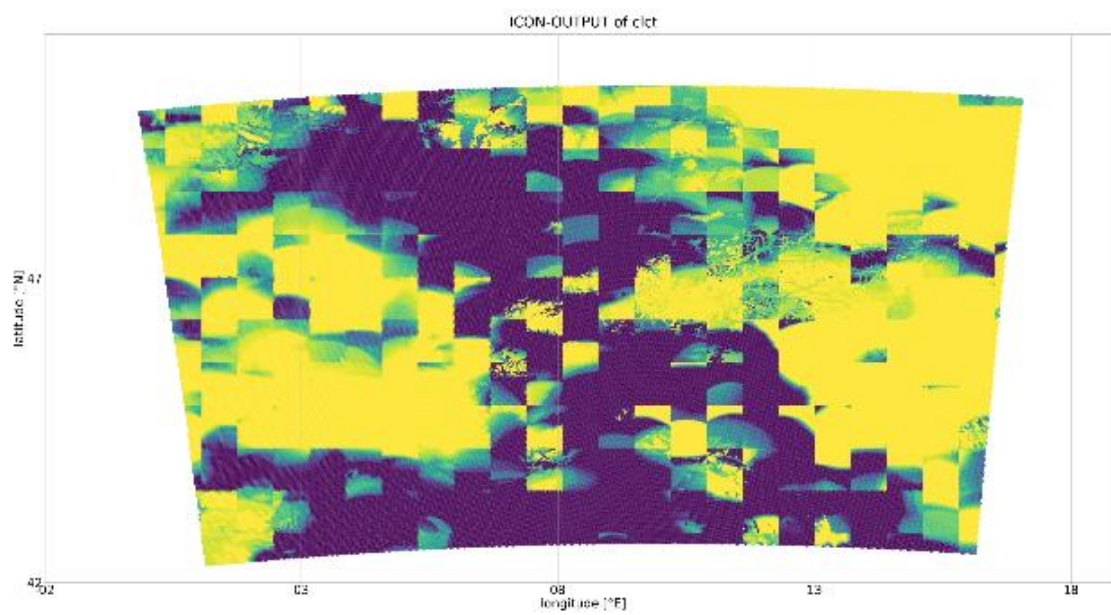
Interpolation method 3 (RBF) with scaling factor 0.01 produce errors after running icon-art with processing-chain (100hPA level is not found).

Before running Icon, we have to prepare initial conditions, i.e. remap it. Before we were doing it with Icontools, which is produced the strange chess-like pattern for almost all ic variables, we also had unrealistic `w_snow` variable, this snow was not melting with time and that influenced on surface fluxes in Flexpart simulations. That is why we replaced the icontools by CDO for remapping step for IC and BCs.

Below one can see the illustration of chess-like problems in IC and unrealistic values of `W_snow`.



Chess-like pattern in IC remapped with Icontools:



Icon and Icontools with spack

To install **icontools** with spack we need to run just 3 command (it didn't work for me on Daint with icon from CSCS, but it worked for David Oschner with Icon from DWD):

```
module load cray-python

source /project/g110/spack/user/daint/spack/share/spack/setup-env.sh

spack install -v icontools@c2sm-master%gcc slave=daint
```

to install icon with spack we need git clone icon source code and run the following command

```
touch file_for_spack
spack dev-build -u build icon@dev-build%nvhpc \
    icon_target=cpu \
    +art +eccodes\
    config_dir=/users/kivanova/icon
rm file_for_spack
touch file_for_spack
spack dev-build -u build icon@dev-build%pgi \
    icon_target=cpu \
    +art +eccodes\
    config_dir=/users/kivanova/icon

rm file_for_spack
```

Icontools -> CDO for preprocessing step

Before running Icon, we have to prepare initial conditions, i.e. remap it. Before we were doing it with Icontools, which is produced the strange chess-like pattern for almost all ic variables, we also had unrealistic `w_snow` variable, this snow was not melting with time and that influenced on surface fluxes in Flexpart simulations. That is why we replaced the icontools by CDO for remapping step for IC and BCs.

Below one can the script for remapping ICs with CDO:



```
datafilename={cfg.input_root_meteo}/{cfg.meteo_prefix}{cfg.inidate_yyyymmddhh}{cfg.meteo_suffix}
datafile="{{datafilename#*/}}" # get filename without path
outdatafile="{{datafile%.*}}" # get filename without suffix

cdo -t ecmwf -f nc copy {cfg.input_root_meteo}/${datafile}} tmp1.nc
cdo setpartabn,/users/kivanova/new_lbc_processing_chain/cases/icon-art-BRM-CD0ic/mypartab,convert tmp1.nc tmp2.nc

cdo selname,LSM tmp2.nc input_FR_LAND.nc
ncrename -h -v LSM,FR_LAND input_FR_LAND.nc
ls -l {cfg.extpar_filename_scratch}
cdo selname,FR_LAND {cfg.extpar_filename_scratch} output_FR_LAND.nc
ncecat -O -u time output_FR_LAND.nc output_FR_LAND.nc # add time dimension otherwise ICON stops
ncks -h -A -v time input_FR_LAND.nc output_FR_LAND.nc # give time a value to avoid CDO warnings
cdo -L setctomiss,0. -ltc,0.5 input_FR_LAND.nc input_ocean_area.nc
cdo -L setctomiss,0. -gec,0.5 input_FR_LAND.nc input_land_area.nc
cdo -L setctomiss,0. -ltc,1. output_FR_LAND.nc output_ocean_area.nc
cdo -L setctomiss,0. -gtc,0. output_FR_LAND.nc output_land_area.nc
cdo -L setrtoc2,0.5,1.0,1.0 output_FR_LAND.nc output_lsm.nc
rm input_FR_LAND.nc output_FR_LAND.nc

# create file with ICON grid information for CDO
cdo -s selgrid,2 {cfg.dynamics_grid_filename} triangular-grid.nc

# remap land area only variables (ocean points are assumed to be undefined in the input data)
cdo setmisstodis -selname,SMIL1,SMIL2,SMIL3,SMIL4,STL1,STL2,STL3,STL4,W_SNOW,T_SNOW tmp2.nc tmp11.nc
cdo remapdis,triangular-grid.nc tmp11.nc tmp12.nc
# cdo -s div tmp12.nc output_land_area.nc tmp_output_l.nc
mv tmp12.nc tmp_output_l.nc
rm tmp1*.nc

## remap land and ocean area differently for variables
# ocean part
cdo -s selname,SKT tmp2.nc tmp_input_ls.nc
cdo -s div tmp_input_ls.nc input_ocean_area.nc tmp1s1.nc
cdo -s setmisstodis tmp1s1.nc tmp1s2.nc
cdo -s remapdis,triangular-grid.nc tmp1s2.nc tmp1s3.nc
cdo -s div tmp1s3.nc output_ocean_area.nc tmp_ocean_part.nc
rm tmp1s*.nc output_ocean_area.nc input_ocean_area.nc

# land part
cdo -s div tmp_input_ls.nc input_land_area.nc tmp1s1.nc
cdo -s setmisstodis tmp1s1.nc tmp1s2.nc
cdo -s remapdis,triangular-grid.nc tmp1s2.nc tmp1s3.nc
cdo -s div tmp1s3.nc output_land_area.nc tmp_land_part.nc
rm tmp1s*.nc output_land_area.nc input_land_area.nc

# merge remapped land and ocean part
cdo -s ifthenelse output_lsm.nc tmp_land_part.nc tmp_ocean_part.nc tmp_output_ls.nc
```



```
# remap the rest
ncks -h -O -x -v T_SNOW,STL1,STL2,STL3,STL4,SMIL1,SMIL2,SMIL3,SMIL4,SKT,LSM tmp2.nc tmp_input_rest.nc
cdo -s remapdis,triangular-grid.nc tmp_input_rest.nc ifs_ini.nc

# remap the snow
ncks -h -O -x -v W_SNOW tmp2.nc tmp_input_snow.nc
cdo -s remapdis,triangular-grid.nc tmp_input_snow.nc ifs_ini.nc

# merge remapped files plus land sea mask from EXTPAR
ncks -h -A tmp_output_l.nc ifs_ini.nc
ncks -h -A tmp_output_ls.nc ifs_ini.nc
ncks -h -A output_lsm.nc ifs_ini.nc
rm -f tmp_output_l.nc tmp_output_ls.nc tmp_input_ls.nc tmp_input_rest.nc output_lsm.nc

# attribute modifications
ncatted -h -a coordinates,FR_LAND,o,c,"clon clat" ifs_ini.nc

# renamings
ncrename -h -v FR_LAND,LSM ifs_ini.nc
ncrename -h -d cell,ncells ifs_ini.nc
ncrename -h -d nv,vertices ifs_ini.nc

cdo expr,"PS=exp(LNPS)" ifs_ini.nc PS.nc
cdo merge PS.nc ifs_ini.nc out_test1.nc
cdo selvar,W_SNOW out_test1.nc wsnow.nc
cdo selvar,topography_c {cfg.extpar_filename_scratch} topo.nc
cdo merge wsnow.nc topo.nc merged.nc
# cdo setrtoc,-1.e99,9999,0 out_wsnow0.nc {cfg.icon_input_icbc}/${outdatafile}.nc
ncap2 -s 'where(topography_c<=2000) W_SNOW=0' merged.nc wsnow_topo.nc
cdo selvar,W_SNOW wsnow_topo.nc right_wsnow.nc
cdo replace out_test1.nc right_wsnow.nc {cfg.icon_input_icbc}/${outdatafile}.nc
```

Below one can see the script to remap BCs with CDO.

```
export ECCODES_DEFINITION_PATH=/project/gli0/spack-install/daint/eccodes/2.19.0/pgl/g5wilyap5zgpzfwuamx6g47zrk12uk4/share/eccodes/definitions
echo $ECCODES_DEFINITION_PATH

export BINARY_DIR=/project/gli0/spack-install/daint/icontools/c2sm-master/gcc/eg76zscn2fwv3fkglbmas63pnqe6dywx/bin

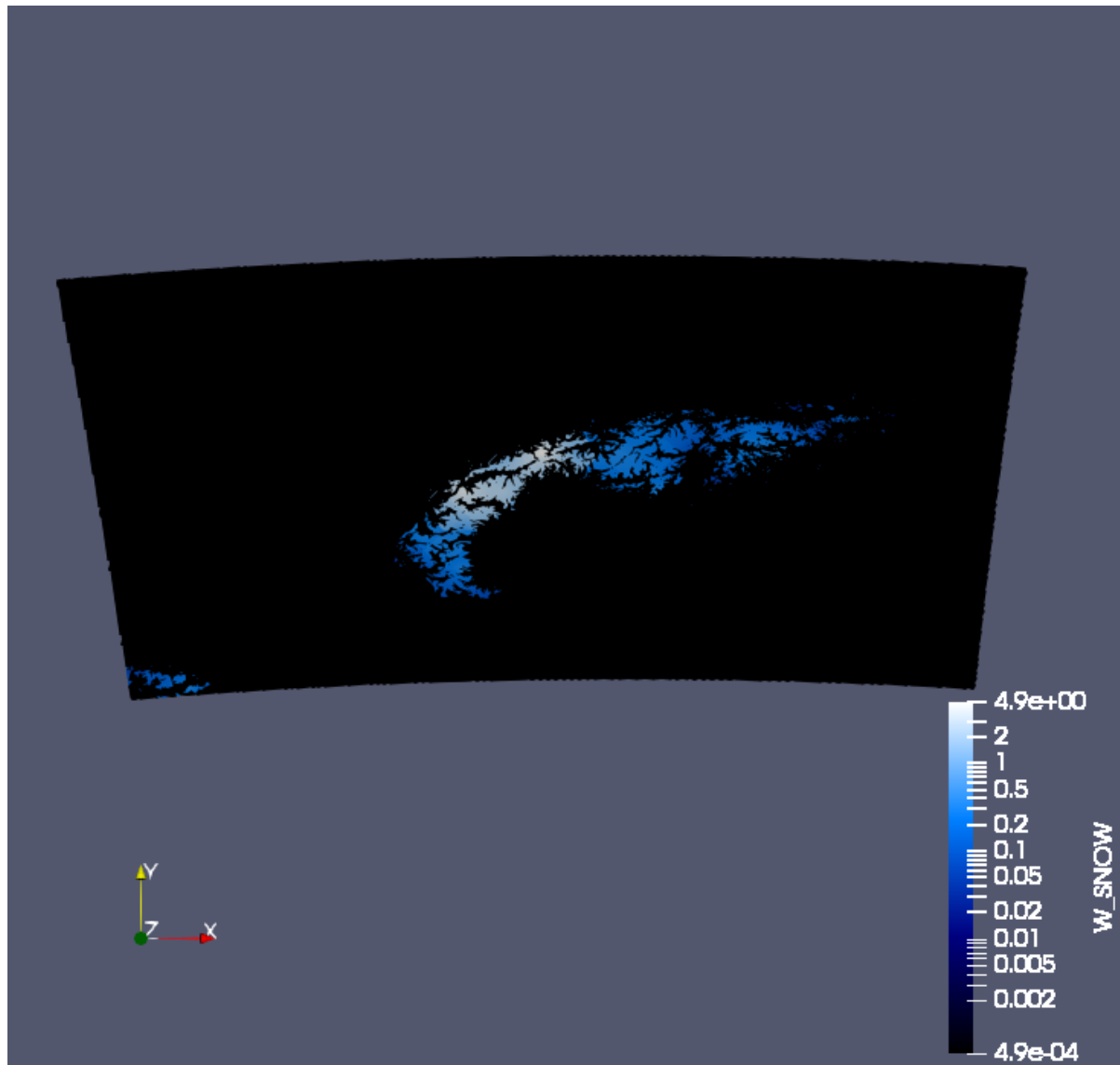
ln -sf /users/kivanova/new_lbc_processing_chain/cases/icon-art-BRM-CD0ic/mypartab

#-----
# Extract boundary data
#-----
#-----
cdo selgrid,2 {cfg.lateral_boundary_grid} triangular-grid_lbc.nc
# loop over file list:
echo "DATAFILELIST is {datafile_list_rest}"
for datafilename in {datafile_list_rest} ; do
    datafile=${datafilename##*/} # get filename without path
    outdatafile=${datafile%.*} # get filename without suffix
    cdo -t ecmwf -f nc copy {cfg.input_root_meteo}/${datafile} tmp1_lbc.nc
    cdo setpartabn,/users/kivanova/new_lbc_processing_chain/cases/icon-art-BRM-CD0ic/mypartab,convert tmp1_lbc.nc tmp2_lbc.nc
    cdo -s remapdis,triangular-grid_lbc.nc -selname,T,U,V,W,LNPS,QV,QI,QR,QC,QS tmp2_lbc.nc {cfg.icon_input_icbc}/${outdatafile}_lbc.nc
    cdo merge {cfg.icon_input_icbc}/${outdatafile}_lbc.nc GEOSP.nc mergedlbc.nc
    mv mergedlbc.nc {cfg.icon_input_icbc}/${outdatafile}_lbc.nc
    ncrename -d cell,ncells {cfg.icon_input_icbc}/${outdatafile}_lbc.nc
    ncrename -d nv,vertices {cfg.icon_input_icbc}/${outdatafile}_lbc.nc
done

#-----
# clean-up
#-----
rm -f nml.log

#-----
exit
```

W_snow



LNPS initial condition remapped with CDO

