DYAMOND

DYAMOND stands for The DYnamics of the Atmospheric general circulation Modeled On Non-hydrostatic Domains, and it was the first ever intercomparison of global-storm-resolving models. Nine models participated, with a great number of simulations being performed with ICON. This page collects some information regarding accessing the output as well as tips and tricks for post-processing native output from different models.

Search

Search only in this Namespaces below. For a global search use the field in the upper right corner. More tipps: how_to_use_the_wikisearch

Accessing DYAMOND output

Nine models participated in the DYAMOND comparison of global storm-resolving models. An overview of the models is provided as part of the ESiWACE DYAMOND website. Because the simulations were so computationally demanding, and because we wanted to gain experience with different workflows, groups were allowed to submit whatever output they wished in whatever format they wished. Model output that was uploaded to DKRZ is saved under

```
/work/ka1081/DYAMOND
```

the raw output (which for many models is on their native grid) will eventually be archived to tape in a way that will be documented at that time. Explore the DYAMOND directory to discover models, and model runs, contributing to output.

For the analysis a workflow was developed where two-dimensional model fields (*x*-*y*) of interest were selected, remapped to a common lon-lat grid, and merged across all their time-steps. These remapped and merged files over all time-steps then served the basis for subsequent aggregation, for instance into time, or zonal, means. Post processed output and some tools used in the postprocessing is standardized and saved in the PostProc directory:

```
/work/ka1081/DYAMOND/PostProc
```

Tips for processing individual models

Examples of the workflow used to arrive at the post-processed output are given below and help users get a start in performing their own processing of the native output.

NCAR MPAS Output

MPAS uses a Voroni unstructured grid, and also writes its output in netcdf5 which may not be readable for many libraries. To process these data with the CDOs the following syntax was developed, whereby it only functions with version 1.9.7 or greater of the CDOs. The 'mpas:' syntax after the 'setgrid' command alerts CDO to the MPAS variable naming for the grid meta-data.

```
GW='/work/ka1081/DYAMOND/PostProc/GridsAndWeights'
cdo -P 8 -f nc4 remap,$GW/0.10_grid.nc,$GW/MPAS-3.75km_0.10_grid_wghts.nc  -
setgrid,mpas:$GW/MPAS-3.75km_gridinfo.nc -selname,vert_int_qc
/work/ka1081/DYAMOND/MPAS-3.75km_Dec2018new/diag.2016-08-01_03.00.00.nc
MPAS-3.75km_vert_int_qc_0.10deg.nc
```

The setgrid command associates the MPAS cells with geographic coordinates, i.e., cell longitude, latitude, boundaries and areas. For many of the models with unstructured grids the grid is disassociated to save space. For this processing we created and used light-weight versions of the MPAS full grid-info files (MPAS-3.75km_gridinfo.nc or MPAS-7.5km_gridinfo.nc) as for these grid files the minimal amount of information required by CDO was extracted and rewritten in the netcdf2 standard.

To extract a subsetted area of temperature and pressure fields the 3D output of the 3.75 km runs, the following command sequence was used:

```
GW='/work/ka1081/DYAMOND/PostProc/GridsAndWeights'
ifile = '/work/ka1081/DYAMOND/MPAS-3.75km-
scaleAwareTiedtke/history.2016-09-09_00.00.00.nc'
ofile = 'MPAS-3.75km.2016-09-09_00.00.00.nc'
cdo -P 2 setattribute,*@axis="txz" -selname,temperature,pressure $ifile
temp.nc
cdo -P 2 sellonlatbox,0,360,-10,10 -setgrid,mpas:$GW/MPAS-3.75km_gridinfo.nc
temp.nc $ofile
rm temp.nc
```

The processing of the 3D files from MPAS was very slow, with about 30 min for the above command sequence. It was also memory intensive and frequently crashed due to insufficient memory on shared nodes. We were able to process four files concurrently by running interactively on a separately allocated post-processing node (made available by `salloc –partition=prepost –account=xxxxxx -N 1')

GFDL FV3 Output

For the Cubed-sphere as used at GFDL the CDO processing requires that a grid be created and stitched together, as the output is distributed on the six base tiles of the cubed sphere. This is done as follows:

```
for N in {1..6}; do
    cdo import_fv3grid $GW/grid_spec.tile$N.nc FV3-3.3km_gridspec.tile$N.nc
done
cdo collgrid FV3-3.3km_gridspec.tile?.nc FV3-3.25km_gridspec.nc
```

Subsequently, to process variables this grid definition needs to be applied, but also the six-tiles have to be collected for each quantity. For instance to read the first times tep of the integrated cloud water the syntax would be as follows:

```
cdo -P 8 seltimestep,1 -setgrid,$GW/FV3-3.3km_gridspec.nc -
collgrid,gridtype=unstructured $DataDir/intql_15min.tile?.nc
FV3-3.3km_intql_15min_001.nc
```

These basic steps can be used to remap the data, and once it exists on a regular grid it can be processed in the familiar ways. For reference, a complete script which also includes steps for remapping and averaging the FV3-3.3km data is provided in this example.

proc_FV3.ksh

```
#! /bin/bash
#SBATCH --job-name=proc_fv3    # Specify job name
#SBATCH --partition=shared     # Specify partition name
#SBATCH --ntasks=5             # Specify max. number of tasks to be
invoked
#SBATCH --mem-per-cpu=5120     # Specify real memory required per CPU
in MegaBytes
#SBATCH --time=08:00:00        # Set a limit on the total run time
#SBATCH --mail-type=FAIL       # Notify user by email in case of job
failure
#SBATCH --account=bb1072       # Charge resources on this project
account
#SBATCH --output=my_job.o%j    # File name for standard output
#SBATCH --error=my_job.e%j     # File name for standard error output

# Author: Daniel Klocke
# Date: Aug 2018
# Purpose: process FV3 2D DYAMOND output to obtain time averaged fields
on 0.1x0.1deg lat lon grid
#
echo "Executing proc_FV3.sh" `date`
source /sw/rhel6-x64/etc/profile.mistral
module swap cdo cdo/1.9.5-magicsxx-gcc64

## choose what you want to do. Files need to be remapped first.
Remapping takes long.
do_weights='no'
do_remap='yes'
do_timeave='no'
do_zonmean='no'
do_ztmean='no'

## choose variables to processe.
#
## List of FV3 2D fields:
#                          cape, flds, cin, flus, flut, fsds, fsdt, fsus
#                          fsut, h500, intqg, intqi, lhflx, u10m, pr,
intql,
#                          u200, ps, intqr, intqs, ustrs, intqv, v10m,
rh500,
#                         v200, rh700, rh850, shflx, t2m, vstrs, ts
declare -a var_list=("ps") #"cape" "flds" "cin" "flus" "flut" "fsds"
"fsdt" "fsus" "fsut" "intqg" "intqi" "lhflx" "u10m" "pr" "intql"
"intqr" "intqs" "intqv" "v10m" "shflx" "t2m" "ustrs" "vstrs" "ts")

## Directories
tmp_dir='/scratch/m/m218027'
out_dir='/work/ka1081/Hackathon/GrossStats'
fv3_dir='/work/ka1081/DYAMOND/FV3-3.25km'

## Creat grid for original FV3 grid to latlon
if [ $do_weights == 'yes' ]
then
    echo 'creating remapping weights from FV3 cubed spere to 0.1x01.deg
lat lon grid'
    for N in 1 2 3 4 5 6; do
      cdo import_fv3grid
/work/ka1081/DYAMOND/FV3-3.25km/2016082100/grid_spec.tile$N.nc
${tmp_dir}/FV3-3.25km/gridspec.tile$N.nc
    done
    cdo collgrid ${tmp_dir}/FV3-3.25km/gridspec.tile?.nc
${tmp_dir}/FV3-3.25km/gridspec.nc
    cdo -P 12 genycon,${out_dir}/0.10_grid.nc -
setgrid,${tmp_dir}/FV3-3.25km/gridspec.nc -
collgrid,gridtype=unstructured
${fv3_dir}/2016080100/lhflx_15min.tile?.nc
${tmp_dir}/FV3-3.25km/fv3_weights_to_0.10deg.nc
fi


## remapping
if [ $do_remap == 'yes' ]
then
    echo 'remapping FV3 fields to 0.1x01.deg lat lon grid'
    cd ${tmp_dir}
    for var in "${var_list[@]}"; do
    echo 'remapping ' ${var}
    cdo -P 12
remap,${out_dir}/0.10_grid.nc,${tmp_dir}/FV3-3.25km/fv3_weights_to_0.10
deg.nc -setgrid,${tmp_dir}/FV3-3.25km/gridspec.nc -
collgrid,gridtype=unstructured
${fv3_dir}/2016080100/${var}_15min.tile?.nc ${tmp_dir}/FV3_${var}_1.nc
    cdo -P 12
remap,${out_dir}/0.10_grid.nc,${tmp_dir}/FV3-3.25km/fv3_weights_to_0.10
deg.nc -setgrid,${tmp_dir}/FV3-3.25km/gridspec.nc -
collgrid,gridtype=unstructured
${fv3_dir}/2016081100/${var}_15min.tile?.nc ${tmp_dir}/FV3_${var}_2.nc
    cdo -P 12
remap,${out_dir}/0.10_grid.nc,${tmp_dir}/FV3-3.25km/fv3_weights_to_0.10
deg.nc -setgrid,${tmp_dir}/FV3-3.25km/gridspec.nc -
```

```
collgrid,gridtype=unstructured
${fv3_dir}/2016082100/${var}_15min.tile?.nc ${tmp_dir}/FV3_${var}_3.nc
    cdo -P 12
remap,${out_dir}/0.10_grid.nc,${tmp_dir}/FV3-3.25km/fv3_weights_to_0.10
deg.nc -setgrid,${tmp_dir}/FV3-3.25km/gridspec.nc -
collgrid,gridtype=unstructured
${fv3_dir}/2016083100/${var}_15min.tile?.nc ${tmp_dir}/FV3_${var}_4.nc
    cdo cat ${tmp_dir}/FV3_${var}_?.nc
${out_dir}/AllTimeStepsOn0.1degGrid/fv3_${var}_0.10deg.nc
done
fi

## time means
if [ $do_timeave == 'yes' ]
then
    echo 'time averaging remapped FV3 fields'
    cd ${out_dir}/AllTimeStepsOn0.1degGrid

    for var in "${var_list[@]}"; do
      if [ ! -f fv3_${var}_0.10deg.nc ]; then
        echo 'file fv3_'"${var}"'_0.10deg.nc does not exist'
        echo 'remap first!'
        exit
      fi
      cdo timmean fv3_${var}_0.10deg.nc
../40DayTimeAveragesOn0.1degGrid/fv3_0d_40d_${var}_0.10deg.nc
      cdo -timmean -seltimestep,1/960 fv3_${var}_0.10deg.nc
../10DayTimeAveragesOn0.1degGrid/fv3_0d_10d_${var}_0.10deg.nc
      cdo -timmean -seltimestep,961/1920 fv3_${var}_0.10deg.nc
../10DayTimeAveragesOn0.1degGrid/fv3_10d_20d_${var}_0.10deg.nc
      cdo -timmean -seltimestep,1921/2880 fv3_${var}_0.10deg.nc
../10DayTimeAveragesOn0.1degGrid/fv3_20d_30d_${var}_0.10deg.nc
      cdo -timmean -seltimestep,2881/3840 fv3_${var}_0.10deg.nc
../10DayTimeAveragesOn0.1degGrid/fv3_30d_40d_${var}_0.10deg.nc
    done
fi

## zonal means
if [ $do_zonmean == 'yes' ]
then
    echo 'zonally averaging remapped FV3 fields'
    cd ${out_dir}/AllTimeStepsOn0.1degGrid

    for var in "${var_list[@]}"; do
      if [ ! -f fv3_${var}_0.10deg.nc ]; then
        echo 'file fv3_'"${var}"'_0.10deg.nc does not exist'
        echo 'remap first!'
        exit
      fi
      echo 'processing ' ${var}
      cdo -f nc4 -k auto zonmean fv3_${var}_0.10deg.nc
```

```
../ZonMeanOn0.1degGrid/fv3_zonmean_${var}_0.10deg.nc
    done
fi

## zonal time means
if [ $do_ztmean == 'yes' ]
then
    echo 'time averaging zonal averaged remapped FV3 fields'
    cd ${out_dir}/ZonMeanOn0.1degGrid

    for var in "${var_list[@]}"; do
      if [ ! -f fv3_zonmean_${var}_0.10deg.nc ]; then
        echo 'file fv3_zonmean_'"${var}"'_0.10deg.nc does not exist'
        echo 'remap and zonally average first!'
        exit
      fi
      echo 'processing ' ${var}
      cdo timmean fv3_zonmean_${var}_0.10deg.nc
fv3_zonmean_0d_40d_${var}_0.10deg.nc
      cdo timmean -seltimestep,1/960 fv3_zonmean_${var}_0.10deg.nc
fv3_zonmean_0d_10d_${var}_0.10deg.nc
      cdo timmean -seltimestep,961/1920 fv3_zonmean_${var}_0.10deg.nc
fv3_zonmean_10d_20d_${var}_0.10deg.nc
      cdo timmean -seltimestep,1921/2880 fv3_zonmean_${var}_0.10deg.nc
fv3_zonmean_20d_30d_${var}_0.10deg.nc
      cdo timmean -seltimestep,2881/3840 fv3_zonmean_${var}_0.10deg.nc
fv3_zonmean_30d_40d_${var}_0.10deg.nc
    done
fi

exit
```

Arpege NH

Processing Arpege required a number of steps, as the model output was attuned to the operational culture of the operational (Weather Centers). The first step was to run fortran code made available by CNRM (Ludovic Auger) to preprocess the data from MeteoFrance GRIB to ECMWF GRIB:

```
/work/ka1081/DYAMOND/PostProc/Utilities/gribmf2ecmwf $fin $tmp1
```

This creates a Grid file, but it is not readable by the present version of CDO. To work on these GRIB files use the ecCodes package (version 2.7.0) developed and distributed by ECWMF. A build is provided in

```
/work/ka1081/DYAMOND/PostProc/Utilities/eccodes-2.7.0-Source/build/bin/
```

Some useful tools are grib_ls and grib_dump, where grib_ls will give you a variable listing of the file it operates on. Here using grib_ls to inspect a 2D file (first processed with gribmf2ecmwf) will identify the variable in position 5 as LSWP (large-scale water precipitation). This variable can be extracted into a CDO parseable file using the ecCodes grib_copy utility:

```
/work/ka1081/DYAMOND/PostProc/Utilities/eccodes-2.7.0-
Source/build/bin/grib_copy -w count=5 $tmp1 $tmp1.grb
```

This file ($tmp1.grb) can then be operated on like every other grib file; remapping requires the weights and desired grid, but follows like for other variables, with the grid files and weights specified as shown here.

```
cdo -f nc4 remap,$GW/0.10_grid.nc,$GW/ARPEGE-2.5km_0.10_grid_wghts.nc -
setgrid,$GW/griddes.arpege1 -setgridtype,regular $tmp1.grb fout.nc
```

Tips for processing 3D output

ICON

Processing of 3D ICON output is relatively simple because CDOs are designed to work for ICON output. It is generally a good idea to start with ICON before converting or working with the 3D output of the other models.

If only a certain level is of relevance, this can be extracted with, e.g.,

```
cdo -P 12 sellevidx,30
/work/ka1081/DYAMOND/ICON-5km/nwp_R2B09_lkm1006_atm_3d_t_ml_20160801T000000Z
.grb t_20160801T000000Z.grb
```

If more levels are needed, 30 can be replaced with a list (30 → 30,31,32 etc.)

Remapping works just like for 2D variables, i.e.,

```
cdo -P 12 -f nc4 -remap,0.20_grid.nc,ICON_R2B10_0.20_grid_wghts.nc
t_20160801T000000Z.grb t_20160801T000000Z.nc
```

For 3D variables, vertical interpolation is often required. The most accurate command for this is cdo intlevel3d. Please refer to the cdo manual for a description. intlevel3d lets you interpolate from any 3D field, for example, geometric height 'zg' to specified 3D geometric heights. The command cdo ap2hl could also be used to interpolate to height levels, but these would be pressure heights (assuming a constant scale height).

Some other commands that can be useful for 3D data are: merging variables into one file with cdo merge and splitting files into individual timesteps with cdo splithour. This can help reduce the amount of data to what is really needed.

Interpolation can require a large amount of memory. Sometimes it is necessary to make use of mistral's high-memory nodes. This is done by specifying in the batch script header:

```
#SBATCH --partition=compute2,compute2,prepost,gpu
```

```
#SBATCH --exclusive
```

submit the script with:

```
sbatch --mem=250G -N 1 --exclusive script.sh
```

IMPORTANT: only do this as a last resort. There are very few graphics nodes and they are needed by others. If you absolutely need to use these nodes, only use 1 or 2 at the same time (otherwise you will receive a phone call 😐).

IFS

Converting 3D IFS data takes a little patience. Keep calm and carry out these 5 steps in this order:

1) select N levels (here N = 1)

```
/pf/zmaw/m214003/local/bin/cdo --eccodes -sellevidx,21
/work/ka1081/DYAMOND/ECMWF-4km/gg_uv_mars_out_ml_vor_div_sh.48.grib
file_1.grb
```

2) set numberOfVerticalCoordinateValues to N

```
~m218027/eccodes-2.7.0-Source/build/bin/grib_set -s
numberOfVerticalCoordinateValues=1 file_1.grb file_2.grb
```

3) set grib edition to 1

```
~m218027/eccodes-2.7.0-Source/build/bin/grib_set -s editionNumber=1
file_2.grb file_3.grb
```

4) convert to NetCDF

```
/pf/zmaw/m214003/local/bin/cdo -P 12 -R -f nc copy file_3.grb file_4.nc
```

5) remap

```
/pf/zmaw/m214003/local/bin/cdo -
remap,0.20_grid.nc,ECMWF-4km_0.20_grid_wghts.nc file_4.nc file_5.nc
```

The IFS level heights can be found here: https://www.ecmwf.int/en/forecasts/documentation-and-support/137-model-levels. If the parameter b is zero, then the level is at constant height.

More tricks

Vertical interpolation of 3D data for models other than ICON is problematic. Variable names, attribute names, coordinate axes, and so on, are different or they can simply have different names. This doesn't mean CDO cannot be used. If you managed to interpolate an ICON file, then try to make the other files look like ICON by renaming/adding/deleting attributes, variables, axes ecetera (easy with nco). You may also have to invert the vertical axis (possible with cdo). If nothing helps, you could always do something rather ugly: Assuming you have an ICON file and a file from a different model, both re-gridded to the same horizontal grid. Then you could use your favorite software to overwrite some fields of the ICON file with the respective fields of the other model. This can be helpful, especially if you have other software that needs to read the files but is designed for ICON files.

Watch out for offsets and scale factors. ICON data do not use them, but for at least one other model (NICAM) you have to multiply and add some constant values to arrive at the true value. They are contained in the output files.