ICON Holocene Simulations

This documentation helps you to:

1. run a spin up simulation forced with Holocene conditions from ECHAM runs
2. run a high-resolution nesting simulation starting from restart files from the spin up simulation

Basic Preparation

To start a ICON simulation with Holocene conditions you have to download the ICON binaries for ICON-NWP, especially *icon-2.5.0-rc-orbit*. A detailed description on how to start with ICON can be found in the ICON quick start guide.

```
git clone --recursive git@git.mpimet.mpimet.de:icon-les icon-2.5.0-rc-orbit
cd icon-2.5.0-rc-orbit

patch -p1 < patch-async_latbc_finalize.txt
cd externals/jsbach
patch -p1 ../../patch-jsbach_orbit.txt
cd ../..

./configure --with-fortran=intel17 --with-openmp
./build_command
```

The last step is to create the runscripts. To do so, you need the name of an experimental descriptor file in ./run. In case of the holocene simulations these are:

```
exp.holocene_spinup
exp.holocene_nested
```

In your icon-directory execute:

```
./make_runscripts holocene_spinup
./make_runscripts holocene_nested
```

Running the Holocene spin up simulation

In the following it will be explained how to start the 30-year spin up simulation or in general - how to force ICON with ECHAM initial and boundary data.

To run a ICON simulation you need:

- Initial file
- grid file
- External parameters
- boundary data if your simulation is limited in its area

Grids and external parameters

The grid file(s) can be created either with the DWD online grid generator. The grids from this source were unfortunately connected to problems preforming nesting simulations. The better way is to create the grids with the grid generator you can compile on mistral. How to download the GridGenerator can be found in the ICON quick start guide. The setup to create the grids for the Holocene Simulations:

```
set_spr_optimization_grids
base_file_name="Global_Icos_<resolution>.nc"
max_resolution=40000
create_icosahedron_grids

#creat first patch
# cut_lonlat_rectangle_grid Global_Icos_0039km.nc nest_patch.nc -14.5 4
75.5 44
cut_lonlat_rectangle_grid Global_Icos_0039km.nc nest_patch.nc 14.5 4  85
54

#==============================================================================
===
#  create nests
  rm holocene_*_nested.nc
  cp nest_patch.nc holocene_40km_nested.nc

cut=$lonlat_rectangle_condition
inner=$inner_cells_condition
inner_cells_depth=13

  no_of_patches=5
  patchNameList=(     holocene_40km_nested.nc     holocene_20km_nested.nc
holocene_10km_nested.nc     holocene_5km_nested.nc
holocene_2_5km_nested.nc    )
  patchIDList=(           1              2              3
4                5    )
  patchParentIDList=(     0              1              2
3                4    )
  patchShapeList=(      none           $inner         $inner
$cut             $inner  )
  patchCenterXList=(    none           14.5           14.5           8
8    )
  patchCenterYList=(    none             4              4
20               20    )
  rectangleXradiousList=(none           70.5           68
45               42    )
  rectangleYradiousList=(none           39             36.5
20               18    )
   patchOptimization=(   .false.       .false.         .false.
.false.          .false. )
```

```
  boundary_indexing_depth=12

  set_hdcp2_optimization_grids
  create_hierarchical_grids
```

The external parameters need to be created seperately. For this you can ask Daniel Klocke from DWD.

Prepare the boundary data

To prepare the boundary data, you need to install the DWD icontools. These are needed to remap the boundary data to the *boundary frame grid*. Check also the ICON quick start guide on this under “ICON tools”.

```
git clone git@git.mpimet.mpg.de:dwd_icon_tools.git
```

There is a detailed documentation on the DWD ICON tools which you can find in your icontools directory:

```
dwd_icon_tools/doc/icontools doc.pdf.
```

The boundary data are necessary to run limited area simulations. In our case they are red 6-hourly from ICON. We want to use **ECHAM data** to create the boundary files.

In the List of Simulations you find the paths to all conducted holocene simulations and their output. We use the **slo0021a** run. ICON needs the boundary data files for each time step seperately. The following steps help you to prepare the boundary data:

1. convert grib to nc and convert echam code numbers to variable names

```
cdo -f nc -t echam copy infile file1
```

2. convert spectral to grid-point grid: needed because ECHAM is a spectral grid model

```
cdo sp2gp file1 file2
```

3. use the afterburner to get the vertical velocity from u and v

```
cdo after file2 file3 <<EON
CODE=131,132,135 TYPE=20
EON
```

4. extract only the needed variables for the boundary data

```
ncks -v
hyai,hyam,hybi,hybm,lev,t,tsw,aps,q,xi,xl,tsurf,geosp,var131,var132,var135
file3 file4
```

5. step in between (not necessary) and rename the variables: you can do this step directly in the

script for mapping the boundary data to the boundary frame grid

```
cdo chname,var131,U -chname,var132,V -chname,var135,OMEGA -chname,tsw,SST -
chname,aps,PS -chname,q,QV -chname,t,T -chname,xi,QI -chname,xl,QC -
chname,geosp,GEOSP file4 file5
```

6. because there were problems with the time axis in ICON do:

```
cdo -a setcalendar,standard file5 file6
ncatted -a units,time,m,c,day as %Y%m%d.%f file6 file7
```

With the following script you can split the yearly files into one-time step nc-files and seperate the needed variables:

```
import numpy as np
import netCDF4 as nc4
from   netCDF4 import Dataset
import datetime
import os


MainFolder="/work/mh0731/m300674/ICON/boundary_data/original_data/only_one/
"
for (path, dirs, files) in os.walk(MainFolder):
    for ncfile in files:
        if ncfile[-3:]=='.nc':
            ncfile=os.path.join(path,ncfile)
            ncfile=Dataset(ncfile, 'r', 'NETCDF4')
            lon_in = ncfile.variables['lon'][:]
            lat_in = ncfile.variables['lat'][:]
            time_in = ncfile.variables['time'][:]
            lev_in = ncfile.variables['lev'][:]
            T_in  = ncfile.variables['T'][:]     # Air Temperature
            QV_in  = ncfile.variables['QV'][:]     # specific humidity
            GEOSP_in = ncfile.variables['GEOSP'][:] # surface geopotential
            QI_in = ncfile.variables['QI'][:]   # cloud ice
            QC_in = ncfile.variables['QC'][:]   # cloud water
            U_in = ncfile.variables['U'][:]
            V_in = ncfile.variables['V'][:]
            OMEGA_in = ncfile.variables['OMEGA'][:]
            PS_in = ncfile.variables['PS'][:]
            SST_in = ncfile.variables['SST'][:]
            ncfile.close()

   # ----- CREATE 6-HOURLY FILES ---------------------------

            for t in range(0,len(time_in)):

               number_dec = time_in[t]-int(time_in[t])
               number_int = int(time_in[t])
```

```
                print(number_dec)

                if number_dec < 0.5:
                    res = str(number_int) + "_0"+ str(int(number_dec * 24))
                else:
                    res = str(number_int) + "_" + str(int(number_dec * 24))

                print('t',t,time_in[t])

                f_out =
nc4.Dataset('/work/mh0731/m300674/ICON/boundary_data/splitted_boundary_data
/boundary_conditions_%s.nc' % res,'w', format='NETCDF4_CLASSIC')
                print('boundary_conditions_%s.nc' % res)

                f_out.createDimension('lon', len(lon_in))
                f_out.createDimension('lat', len(lat_in))
                f_out.createDimension('time', None)
                f_out.createDimension('lev',len(lev_in))

                 #------ from first file ----------------------------------
                lat = f_out.createVariable('lat', np.float,'lat')
                lat[:] = lat_in

                lon = f_out.createVariable('lon', np.float,'lon')
                lon[:] = lon_in

                time = f_out.createVariable('time',np.double,'time')
                time[0] = time_in[t]

                lev = f_out.createVariable('lev',np.double,'lev')
                lev[:] = lev_in

                T =
f_out.createVariable('T',np.float,('time','lev','lat','lon'))
                T[0,:,:,:] = T_in[t,:,:,:]

                PS =
f_out.createVariable('PS',np.float,('time','lat','lon'))
                PS[0,:,:] = PS_in[t,:,:]

                QV =
f_out.createVariable('QV',np.float,('time','lev','lat','lon'))
                QV[0,:,:,:] = QV_in[t,:,:,:]

                U =
f_out.createVariable('U',np.float,('time','lev','lat','lon'))
                U[0,:,:,:] = U_in[t,:,:,:]

                V =
f_out.createVariable('V',np.float,('time','lev','lat','lon'))
                V[0,:,:,:] = V_in[t,:,:,:]
```

```
                OMEGA =
f_out.createVariable('OMEGA',np.float,('time','lev','lat','lon'))
                OMEGA[0,:,:,:] = OMEGA_in[t,:,:,:]

                QI =
f_out.createVariable('QI',np.float,('time','lev','lat','lon'))
                QI[0,:,:,:] = QI_in[t,:,:,:]

                QC =
f_out.createVariable('QC',np.float,('time','lev','lat','lon'))
                QC[0,:,:,:] = QC_in[t,:,:,:]

                GEOSP =
f_out.createVariable('GEOSP',np.float,('time','lat','lon'))
                GEOSP[0,:,:] = GEOSP_in[t,:,:]

                SST =
f_out.createVariable('SST',np.float,('time','lat','lon'))
                SST[0,:,:] = SST_in[t,:,:]

                #----local attributes ---------
                lon.units = 'degrees_east'
                lat.units = 'degrees_north'
                T.long_name = 'Air Temperature'
                T.units = 'K'
                PS.long_name = 'Surface pressure'
                PS.units = 'Pa'
                QV.long_name = 'Specific humidity'
                QV.units = 'kg/kg'
                U.long_name = 'horizontal wind component u'
                U.units = 'm/s'
                V.long_name = 'horizontal wind component v'
                V.units = 'm/s'
                OMEGA.long_name = 'vertical wind velocity'
                OMEGA.units = 'Pa/s'
                QI.long_name = 'cloud ice'
                QI.units = 'kg/kg'
                QC.long_name = 'cloud water'
                QC.units = 'kg/kg'
                GEOSP.longname = 'Surface geopotential'
                GEOSP.units = 'm2 s-2'
                SST.longname = 'Sea Surface Temperature'
                SST.units = 'K'

                #-------------------------------------------------------------
---------

                f_out.close()
```

The variables which are needed are : **U, V, W, PS, T, SST, GEOSP, QC, QI, QV** Because of the high time resolution the boundary data are needed, it is useful to minimize the file size. Therefore we map the boundary data on a boundary frame grid. To do so, we use the DWD ICON tools. Here is a setup of the script: The script can be found under the following path:
`/work/mh0731/m300674/ICON/boundary_data/xce_remap_lbc_ECHAM`

In all of the scripts you have to adjust the paths to your own directories and you can adjust variable names, attributes etc.

Prepare the initial file

The initial file for the Holocene simulation consists of data from the ECHAM holocene Data, IFS data and some variables which are set to zero. You can use your first boundary file, **before** you remapped it onto the boundary frame. Because there are no variables for soil moisture in the ECHAM data (as they are needed in ICON) we use IFS data for the soil initialization. Variables you need to extract from an IFS file: **SMIL1, SMIL2, SMIL3, SMIL4, STL1, STL2, STL3, STL4, SKT, CI, LSM**. Variables you can set to zero are mainly ice and snow variables, which can be neglected in our domain: **W_SNOW, T_SNOW, W_I. RHO_SNOW**

Do not forget to adjust the time axis to be identical in all files

Running the high resolution nesting simulation

You already prepared the grids and external parameters, as well as the boundary data. The latter are needed for the coarsest resolution only. The higher resolution domains are forced by the coarser ones respectively. First, you should check when the soil moisture equilibrates and chose a date to start the nesting simulation after that point. The first-guess file need to be remapped to all of your domains.

The nested simulation will be initialized from first guess variables. You can write out these variables using **“group:dwd_fg_atm_vars”,“group:dwd_fg_sfc_vars”** in your output_nml.

You have to write out only the one instant time step from which you want to start your nesting simulation

Create the runscript for the nesting simulation mentioned above. With this you can start the nested Holocene simulations easily.