Hands on exercises with ORCHIDEE OFF-LINE

Revised for training session 2014-11-06 - 2014-11-07

Josefine Ghattas, IPSL

The goal of this exercise is to learn how to install, compile and launch a basic test case with ORCHIDEE in off-line mode. Exercice 3-4 are using libIGCM. All exercises can be done at curie/TGCC, Ada/IDRIS or obelix/LSCE. All commands needed for the basic exercises are listed in the text.

Today at Ada: use training account

For you that have login at IDRIS or do not have a login, todays training session will be done at temporary acounts. These acounts have specific priority in the queue to avoid to much waiting time. Access this acount by the menu apearing while clicking with the mouse.

For the exercises with libIGCM, using the account for the traing session there is no access to the archive machine ergon. Therfor you need to specify in each config.card to use the workdir at Ada by setting in the UserChoicies section:

```
ARCHIVE=$WORKDIR
```

Today at curie: use specific project id

Connect to curie from the Ada account for the training session, using following:

```
ssh -X yourlogin@curie-ccrt.ccc.cea.fr
```

For curie during the training sessions in november 2014, always use the project id tgcc0052 instead of your normal project id. For exercises with libIGCM the choice is done while launching ins_job. This script will ask you to enter the project id which will then be set in the heading of the jobs. Special resources are allocated using this project number. With this project number it is not possible to submit in test queue.

Today at obelix: use specific queue

Connect to obelix using:

```
ssh -X -t yourlogin@idefix1.extra.cea.fr ssh obelix
```

A part of obelix has been reserved for the training session. Use the queue **train** for all exercices today to have your jobs running faster.

For the exersices using libIGCM, before creating the jobs, after installing modipsl and extracting the configuration (after ./model config_name) change the AA_ files in modipsl/libIGCM to take this queue name. In the AA_* files in modipsl/libIGCM the suffix lxiv8 corresonds to obelix. Change in all AA_ files the line:

```
#-Q- lxiv8 #PBS -q medium
```

into

```
#-Q- lxiv8 #PBS -q train
```

This has to be done in: AA_job, AA_rebuild_fromWorkdir, AA_create_ts, AA_create_se, AA_atlas_LMDZ, AA_atlas_ORCHIDEE, AA_monitoring, AA_pack_debug, AA_pack_output, AA_pack_restart

1 Install and compile

Download first **modipsl** and explore what is inside. modipsl contains some tools in the directory **util**. In util, scripts are found for : extraction (*model, mod.def*), creation of makefiles (*ins_make, AA_make.gdef*), creation of job (*ins_job*) and some more. modipsl is also a empty file tree that will receive the models and tools.

Start this exercise by logging in to obelix and extract modipsl in a new directory :

```
mkdir MYFIRSTTEST ; cd MYFIRSTTEST
svn co http://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl
cd modipsl/util
ls
```

The script **model** is used to download a specific predefined configuration with the model sources and tools needed. The script uses the file **mod.def** that contains specifications for each configuration predefined. Use **./model -h** to see all existing configurations and **./model -h** *config_name* for information about a specific configuration. Same information can be found in the file mod.def on a specific syntax. Download a configuration using **./model** *config_name.*

For ORCHIDEE, there are different predefined configurations available. Set *config_name* to one of following :

- ORCHIDEE_trunk : the latest revision of the trunk ORCHIDEE for off-line use. Note that the latest revision is not always stable and changes often.
- ORCHIDEE_TAG : the tag ORCHIEE_1_9_6 for off-line use.
- ORCHIDEE_SVN_AR5 : off-line version used for reference CMIP5 simulations.
- LMDZOR_v5 : this is a configuration using LMDZ5(atmospheric model) and ORCHIDEE version AR5.
- LMDZOR_v5.2 : same as LMDZOR_v5 but with a recent revision number on the trunk ORCHIDEE.
- IPSLCM5_v5 : coupled configuration containing NEMO(ocean model), LMDZ and ORCHIDEE version AR5. Different versions of IPSLCM5_v5 exist also.

Note : All _v5 configurations contain ORCHIDEE version CMIP5.

In this exercise we will use the configuration ORCHIDEE_trunk. Open mod.def and look at lines begging with ORCHIDEE_trunk and then extract as follow :

```
vi mod.def               # Explore lines beggining with ORCHIDEE_trunk
./model ORCHIDEE_trunk
```

Now explore the directories in modipsl. You will find all source code for ORCHIDEE in directory modipsl/modeles. You also find the directory IOIPSL which is a fortran library that is linked to ORCHIDEE for input/output issues. In directory modipsl/config/ORCHIDEE_OL you find scripts to run ORCHIDEE using libIGCM. libIGCM is a tool developed at IPSL to run coupled and off-line simulations. Specific training session about libIGCM are given by the Plat-forme groupe at IPSL.

The makefiles were created automatically in the end of the script model, done by the script **ins_make**. ins_make will detect at which machine you are working on and create adapted makefiles. By default ins_make recognize the following machines : curie at TGCC, ada at IDRIS and obelix at LSCE. ins_make can also be re-launched manually. For example this is needed if you move the modipsl directory or if you want to create makefiles for another target machine. The main makefile is found in modipsl/config/ORCHIDEE_OL directory.

Now compile the model :

```
cd ../config/ORCHIDEE_OL
gmake
```

When the compiling is finished you will find the executables orchidee_ol, teststomate and forcesoil in modipsl/bin.

Note 1 : Compile using older version

The source code has been slightly re-organized since revision 1042 on ORCHIDEE/trunk version. Now the programs and subroutines for drivers are found in a new directory modipsl/modeles/ORCHIEE/src_driver. For older revisions on the trunk, tags and some old branches the source code is divided into directory modipsl/modeles/ORCHIDEE and modipsl/modeles/ORCHIDEE_OL. In directory ORCHIDEE_OL program and subroutines for the drivers are found. In modipsl/modeles/ORCHIDEE_OL the main Makefile is found.

If in your ORCHIDEE directory no src_driver exist, then you are in the old structure. In this case, compile as follow :

```
cd modipsl/modeles/ORCHIDEE_OL
gmake orchidee_ol
gmake teststomate
gmake forcesoil
```

Note 2 : Compile at a local PC - not needed for obelix

It is possible to install ORCHIDEE at a local linux PC or laptop. We recommend to use the compiler gfortran (or ifort). First you need to install netcdf and compile it with gfortran. Netcdf must be compiled with the same compiler as you will use to compile the model. The file modipsl/util/AA_make.gdef contains compile options for different target platforms. Modify the section begging with gfortran at least for the path to your netcdf library. Also change the path to your netcdf library in ORCHIDEE/arch/arch-gfortran.path. Re-generate the main makefile using ./ins_make -t gfortran and compile as normal.

2 Test simulations

We will now do some test simulations. This will be done interactively to easier understand what is happening. To run the model you need at least the following files in the run directory :

- orchidee_ol : ORCHIDEE executable
- run.def : parameter text file
- forcing_file.nc : climate forcing variables (this file can have another name, in that case it should be indicated in run.def file)
- PFTmap.nc : vegetation map
- soils_param.nc : initialization of soil parameters
- reftemp.nc : initialization of temperature

2.1 First regional run

Create a new directory where to run the model and copy or link the ORCHIDEE executable :

```
cd MYFIRSTTEST; mkdir RUN1; cd RUN1
ln -s ../modipsl/bin/orchidee_ol .
```

Create the parameter file by saving the following lines into a file named run.def :

```
TIME_LENGTH=31D
STOMATE_OK_STOMATE= y
LIMIT_WEST =  -20.
LIMIT_EAST =  0.
LIMIT_NORTH =  20.
LIMIT_SOUTH =  0.
```

Copy or link the netcdf files needed into your run directory. The basedir depends on the machine. You can use export if your shell is bash (for shell tcsh: replace export by set):

```
# At obelix :
export basedir=/home/orchideeshare/igcmg/IGCM/
# At Ada:
export basedir=/workgpfs/rech/psl/rpsl035/IGCM
# At curie:
export basedir=/ccc/work/cont003/dsm/p86ipsl/IGCM

ln -s $basedir/BC/OOL/OL2/NCC/ncc_for_1982.nc forcing_file.nc
ln -s $basedir/BC/SRF/OL2/PFTmap_1850to2005_AR5_LUHa.rc2/PFTmap_IPCC_1982.nc PFTmap.nc
ln -s $basedir/INIT/SRF/OL2/soils_param.nc .
ln -s $basedir/BC/SRF/OL2/reftemp.nc .
```

Or if you work on a local PC you must instead download these files :

```
wget http://dods.extra.cea.fr/work/p86ipsl/IGCM/BC/OOL/OL2/NCC/ncc_for_1982.nc forcing_file.nc
wget http://dods.extra.cea.fr/work/p86ipsl/IGCM/BC/SRF/OL2/...
      ...PFTmap_1850to2005_AR5_LUHa.rc2/PFTmap_IPCC_1982.nc PFTmap.nc
wget http://dods.extra.cea.fr/work/p86ipsl/IGCM/INIT/SRF/OL2/soils_param.nc
wget http://dods.extra.cea.fr/work/p86ipsl/IGCM/BC/SRF/OL2/reftemp.nc
```

You can use ncdump to see what is in the netcdf files. For example :

```
ncdump -h forcing_file.nc
```

Now launch the model :

```
./orchidee_ol      # or ./orchidee_ol > out_exec
```

When the model finished correctly, following log message is found in the output text file out_orchidee_0000:

```
 END of dim2_driver
```

2.2 Description of some parameters in run.def

The file run.def contains parameters to run the model. A line beginning with a # is a comment. For all parameters not set in run.def default values in the model will be used.

The model period in each execution is given by the parameter **TIME_LENGTH**. In this test case TIME_LENGTH is 31 days. It is possible to run one year by putting **TIME_LENGTH=1Y**. It is not possible to run less than one day.

LIMIT_EAST, LIMIT_WEST, LIMIT_NORTH and **LIMIT_SOUTH** are the horizontal domain to be modelize in degrees. The default values correspond to the domain of the forcing file. The difference between EAST and WEST, and NORTH and SOUTH must be at least one degree. The model will stop if the domain does not cover any land points with error message :

```
FATAL ERROR FROM ROUTINE dim2_driver
 --> number of land points error.
 -->  is zero !
 --> stop driver
```

Parameter **STOMATE_OK_STOMATE** activates coupling to the stomate in ORCHIDEE.

2.3 Relaunch in the same directory

If you want to relaunch the model in the same directory, then you need to delete restart and output files previously created by the model. Do this now and re-run the model :

```
rm driver_rest_out.nc sechiba_rest_out.nc stomate_rest_out.nc
rm sechiba_history_0000.nc stomate_history_0000.nc
./orchidee_ol
```

2.4 Continue a simulation using restart files

The model writes restart files in the end of each execution period, after a TIME_LENGTH. These files contain all state variables needed to continue a simulation without loosing information. To continue the simulation you need to rename the restart files produced in previous run and activate reading of these files in run.def with parameters SECHIBA_restart_in, STOMATE_RESTART_FILEIN and RESTART_FILEIN. Save the output files sechiba_history.nc and stomate_history.nc for later analyses.

Add in run.def :

```
SECHIBA_restart_in=sechiba_rest_in.nc
STOMATE_RESTART_FILEIN=stomate_rest_in.nc
RESTART_FILEIN=driver_rest_in.nc
```

Rename restart files :

```
mv driver_rest_out.nc  driver_rest_in.nc
mv sechiba_rest_out.nc sechiba_rest_in.nc
mv stomate_rest_out.nc stomate_rest_in.nc
```

Save output :

```
mv sechiba_history_0000.nc sechiba_history_month01.nc
mv stomate_history_0000.nc stomate_history_month01.nc
```

Relaunch the model :

```
./orchidee_ol > out_exec
```

Note that for longer simulations a script library called **libIGCM** is used to chain the executions without manually copy or move of files. libIGCM is a powerful tool but it needs more time to learn to use.

2.5 Add more output

Diagnostic output are written into *history* files produced by IOIPSL. In this default simulation the files sechiba_history.nc and stomate_history.nc are produced. sechiba_history.nc is a file with daily mean values for some variables. It is possible to change the output level by adding parameters in run.def file. For example it is possible to produce a third output file named *sechiba_out_2.nc* with high frequent diagnostics every 30min by adding **SECHIBA_HISTFILE2=y** in run.def. Try this and relaunch the model. Add also **SECHIBA_HISTLEVEL=11** to have more variables in sechiba_history.nc file.

Add in run.def:
(Do not add the comments or add them on seperate lines beiginning with #):

```
SECHIBA_HISTLEVEL=11          # add more variables in sechiba_history.nc
SECHIBA_HISTFILE2=y           # active sechiba_out_2.nc
WRITE_STEP2= 10800.0          # write frequency in seconds for sechiba_out_2.nc
STOMATE_HIST_DT= -1.          # write freq in sec or if -1 monthly, for stomate_history.nc
STOMATE_HISTLEVEL= 10         # add more variables in stomate_history.nc
STOMATE_IPCC_HIST_DT= -1.     # activate stomate_ipcc_history.nc with monthly write frequency
```

Clean up as above or continue with re-naming restart files

```
./orchidee_ol
```

Notice that it took longer time to run the model with more output.

2.6 Visualization with ferret

Here is a small example to visualize sechiba_history.nc using ferret.

```
> ferret
use sechiba_history_0000.nc # read file
sh d                        # list content in file
shade CONTFRAC              # 2D plot of a variable
go land                     # add contour of continents
shade TEMP_SOL[l=1]         # 2D plot of TEMP_SOL for first time step
shade TEMP_SOL[l=@ave]      # 2D plot of TEMP_SOL average over all time steps
shade SWDOWN[i=@ave]        # zonal plot
plot SWDOWN[i=@ave,j=@ave]  # plot mean value over time
quit
```

2.7 Test in parallel mode

If ORCHIDEE is compiled with MPI and using the preprocessing key CPP_PARA, then the model can be run on 1 or several processes MPI. This is the default cas while compiling at obelix, curie and ada.

You will now launch a global test run on 4 MPI processes. Prepare a new run directory as in the first exercise but do not put any regional limits in run.def (remove the parameters LIMIT_). Then create a file Job_orchidee. This file will be different for different machines. Follow one of the sub sections depending on your machine.

2.7.1 At obelix

Create Job_orchidee with following lines :

```
######################
## OBELIX      LSCE ##
######################
#PBS -N test
#PBS -m a
#PBS -j oe
#### For the training session use -q train, otherwise use -q mediump
#PBS -q train
#PBS -o Script_Output
#PBS -S /bin/ksh
#PBS -l nodes=1:ppn=4

cd $PBS_O_WORKDIR
date
time mpirun ./orchidee_ol
date
```

Submit the job to the queue with the command qsub. Check with the qstat and use the command qcat to see the progress of the job. Do following

```
> qsub Job_orchidee       # submit the job
> qstat -u login          # check the job's running status
> qcat job_id |more       # check the progress of the job. job_id is given by qstat
```

2.7.2 At curie

Create Job_orchidee with following lines :

```
#!/bin/ksh
######################
## CURIE   TGCC/CEA ##
######################
#MSUB -r test            # name of the job
#MSUB -o Script_Output    # name of output file for standard messages
#MSUB -e Script_Output    # name of output file for error messages
#MSUB -eo
#MSUB -n 4               # Request numbre of cores
#MSUB -T 1800            # Time limit in seconds
#MSUB -Q test            # Queue test, priority acces
#MSUB -q standard
#MSUB -A tgcc0052        # Set your project id
```

```
BATCH_NUM_PROC_TOT=$BRIDGE_MSUB_NPROC
set +x

cd ${BRIDGE_MSUB_PWD}
date
/usr/bin/time ccc_mprun -n 4 ./orchidee_ol
date
```

Submit the job to the queue with the command ccc_msub. Check with the ccc_mstat. Do following

```
> ccc_msub Job_orchidee       # submit the job
> ccc_mstat -u login          # check the job's running status
```

2.7.3 At Ada

Create Job_orchidee with following lines :

```
#!/bin/ksh
# ######################
# ##  ADA       IDRIS ##
# ######################
# Job name
# @ job_name = SECHSTOM
# Job type
# @ job_type = parallel
# Standard output file name
# @ output = Script_Output_SECHSTOM.000001
# Error output file name
# @ error = Script_Output_SECHSTOM.000001
# Total number of tasks
# @ total_tasks = 4
# @ environment = "BATCH_NUM_PROC_TOT=4"
# Maximum CPU time per task hh:mm:ss
# @ wall_clock_limit = 1:00:00
#### Only for this training cours use class=cours
# @ class = cours
# End of the header options
# @ queue

# Temporary due to memory problems on mpi version
export MP_EUILIBPATH=/smplocal/lib/ibmhpc/pe12012/ppe.pami/gnu/lib64/pami64
date
/usr/bin/time poe ./orchidee_ol
date
```

Submit the job to the queue with the command llsubmit. Check with the llq. Do following

```
> llsubmit Job_orchidee       # submit the job
> llq | grep login            # check the job's running status
```

2.8 Paralllel output

A parallel job will write to several text files, on per process, out_orchidee_0000, out_orchidee_0001... and the output netcdf files will be written in local domain sechiba_history_000.nc, sechiba_history_0001.nc etc. A reconstruction of the output netcdf files to the total domain is necessary. This is done with the *rebuild* tool developped at IPSL as an extension of IOIPSL.

rebuild is installed at the different machines here:

```
# at obelix:
/home/users/igcmg/rebuild/bin/
# at curie:
/ccc/cont003/home/dsm/p86ipsl/rebuild/src_X64_CURIE/modipsl_v2_2_2_netcdf4.2/bin
# at Ada:
/linkhome/rech/psl/rpsl035/bin
```

You can add the path to your $PATH environement variable. Use correct syntax depending on your login shell (echo $SHELL):

```
# SHELL tcsh
setenv PATH add_here_new_path:$PATH

# SHELL bash
export PATH="add_here_new_path:$PATH"
```

Do the rebuild as follow :

```
rebuild -o sechiba_history.nc sechiba_history_00*
```

2.9 Check reproductibility of results

When running on 1 or several processes, the results should be the same. Create a new directory and make the same test but on 1 processes. Check that restart files and output files are the same. You can use cdo to check that netcdf files are identical.

```
> cdo -diffv file1.nc file2.nc
```

Check also the time gain due to the change between 1 or 4 processes. In the ideal case, if ORCHIDEE would be perfectly scalable the job should run 4 times faster on 4 processes than on 1. This is not the case but at least you should notice a significant gain in time while increasing the number of processes.

3 Simulations using libIGCM

The following exercises will now use the ORCHIDEE_OL configuration with libIGCM.

There are some differences between ORCHIDEE_OL configuration and the coupled configurations such as LMDZOR_v5 par example. In the configuration ORCHIDEE_OL it is not needed to create the submit directory. Instead different predefined experiment directories already exist. They can be copied and used directly. These directories are OOL_SEC_STO, OOL_SEC, FORCESOIL, TESTSTOMATE and SPINUP_ANALYTIC. They follow the standard rules described in the training and documentation for libIGCM. The DRIVER directory do not exist but the "comp.driver" files are found in the COMP directory. SPINUP and ENSEMBLE directories contains experiences that are more complicated and are not taught in the course.

In the ORCHIDEE_OL configuration all parameters set in PARAM/run.def parameter file will be used. Even if the drivers treat the same parameter, what is set in run.def will not be changed. This is a big difference from coupled configurations such as LMDZOR_v5. In the _v5 configurations, all variables marked as AUTO in run.def or the other parameter files can not be changed. The driver will change these parameters depending on the choices set in the "comp.card" files.

We will here work with the SPINUP_ANALYTIC experience and the OOL_SEC_STO experiment.

3.1 SPINUP_ANALYTIC experiment

You'll now set up a spinup simulation. Copy the SPINUP_ANALYTIC directory for this exercise. Look into the config.card. The variables CyclicBegin and CyclicEnd describes the years to loop over. Setting these 2 variables in config.card make available the variable CyclicYear which can be used in the comp.card. In COMP/orchidee_ol.card the variable CyclicYear is used to get the forcing file. For this exercise loop over 10years. To do so, change to CyclicEnd=1910.
Limit the region to a point, in run.def:

```
LIMIT_WEST = -60.
LIMIT_EAST = -58.
LIMIT_NORTH = -8.
LIMIT_SOUTH = -10.
```

Set up the simulation to run over 6 forcing periodes (60years in total). In config.card, set DateEnd=1960-12-31. Set SpaceName=TEST to deactivate pack post treatement if running at curie/ada. Set RebuildFrequency=5Y, TimeSeriesFrequency=20Y and SeasonalFrequency=NONE. Set JobNumProcTot=1. Change in orchidee_ol.card to use the CRU-NCEP v5.3 twodeg for the forcing file. Look at the shared accout to see the location exact of these files.

Create the job and launch the simulation.

- Where is the share account? Where are the CRU-NCEP v5.3 stored?
- In orchidee_ol.card you can use the variable year or CyclicYear. Which are the differces?
- The variable SPINUP_PERIOD is calculated by the stomate.driver and set in run.def. Where can you see the run.def file that was used during simulation? What is the SPINUP_PERIOD?

When the time-series have been done, you can have a look at the evolution of the carbon pools. Go to the output directory

```
cd .../IGCM_OUT/..../JobName/SBG/Analyse/TS_YE
ferret
```

```
use JobName_19010101_19301231_1Y_CARBON_ACTIVE.nc
use JobName_19010101_19301231_1Y_CARBON_SLOW.nc
use JobName_19010101_19301231_1Y_CARBON_PASSIVE.nc

set v ul; plot CARBON_ACTIVE[k=@ave,i=@ave,j=@ave,d=1]
set v ur; plot CARBON_SLOW[k=@ave,i=@ave,j=@ave,d=2]
set v ll; plot CARBON_PASSIVE[k=@ave,i=@ave,j=@ave,d=3]
```

3.2 OOL_SEC_STO experiment

Set up an experiment with sechiba and stomate. Change to use CRU-NCEP v5.3 forcing files on 2 degree resolution. Limit to a 30 by 30 degree domain of your choice in run.def. Run the simulation for 10 years, using 1Y period lenght.

For this exercise start by coping the OOL_SEC_STO directory to a new directory. Choose a name for the directory to be the same as the JobName in config.card.

After you launched the job, use the RunChecker.job in libIGCM to check the progression of the simulation.

Answer following questions:

- Where are the output stored?
- How can you change the place for the ARCHIVE directory? Note that this is recommended only at obelix.
- Which output files are produced?
- Did all post-treatement job finish?
- Where are the time-series stored?

More indications and answers to questions

Exercise 3.1 with SPINUP_ANALYTIC

See here all commands needed for exercise 3.1:

```
cd modipsl/config/ORCHIDEE_OL
cp -r SPINUP_ANALYTIC TestSpin
cd TestSpin

# Modify following variables in the config.card
JobName=MySpin
SpaceName=TEST
DateEnd=1960-12-31
CyclicEnd=1910
JobNumProcTot=1
RebuildFrequency=5Y
TimeSeriesFrequency=10Y
SeasonalFrequency=NONE

# Modify COMP/orchidee_ol.card
# Change the path to the forcing file. The new path is
.../CRU-NCEP/v5.3/twodeg/cruncep_twodeg_${CyclicYear}.nc

# Modify PARAM/run.def : Add following
LIMIT_WEST = -60.
LIMIT_EAST = -58.
LIMIT_NORTH = -8.
LIMIT_SOUTH = -10.

# Create the job
../../../util/ins_job

# Change in the heading of main job Job_MySpin to have specific queue for training session only:
   # PBS -q train
# Set PeriodNb=30
```

Answers to questions in exercise 3.1:

- *Where is the share account? Where are the CRU-NCEP v5.3 stored?*
  At obelix:

  ```
  /home/orchideeshare/igcmg/IGCM
  /home/orchideeshare/igcmg/IGCM/BC/OOL/OL2/CRU-NCEP/v5.3
  ```

- *In orchidee_ol.card you can use the variable year or CyclicYear. Which are the differces?*
  The variable year goes from DateBegin and DateEnd whears CyclicYear loops from CyclicBegin to CyclicEnd over and over again.

- *The variable SPINUP_PERIOD is calculated by the stomate.driver and set in run.def. Where can you see the run.def file that was used during simulation? What is the SPINUP_PERIOD?*
  /home/scratch01/login/IGCM_OUT/OL2/TEST/ExperimentName/JobName/OOL/Debug/*run.def
  SPINUP_PERIOD= 10

Exercise 3.2 with OOL_SEC_STO

See here all commands needed for exercise 3.2.

```
cd modipsl/config/ORCHIDEE_OL
cp -r OOL_SEC_STO tsecsto
cd tsecsto

vi config.card
  # Modify: JobName=tsecsto, DateBegin, DateEnd, PeriodLength=1Y

# Change forcing file
vi COMP/orchidee_ol.card
  # Modify in BoundaryFiles/List section
    NCC/ncc_for_${year}.nc
       into
    CRU-NCEP/v5.3/twodeg/cruncep_twodeg_${year}.nc

# Change horizontal domain
vi PARAM/run.def
  # Add a smaller domain, for example:
    LIMIT_WEST = -10.
    LIMIT_EAST = 20.
    LIMIT_NORTH = 30.
    LIMIT_SOUTH = 0.

# Create the main job
../../../util/ins_job

# Launch main job
qsub Job_tsecsto
qstat -u login

# Use RunChecker.job
../../../libIGCM/RunChecker.job -u login -p .
```

Answers to questions in exercise 3.2:

- *Where are the output stored?*
  /home/scratch01/yourlogin/IGCM_OUT/TagName/SpaceName/ExperimentName/JobName
- *How can you change the place for the ARCHIVE directory? Note that this is recommended only at obelix.*
  Since tag libIGCM_v2.4, add in config.card, UserChoices section: ARCHIVE=/home/yourdisc/yourlogin
  In older version of libIGCM ARCHIVE was set in the main job of the simulation. It can also be changed directly in libIGCM/libIGCM_sys/libIGCM_sys_obelix.ksh
- *Which output files are produced?*
  sechiba_history.nc : monthly frequency, sechiba_out_2.nc : 3H high frequency
  stomate_history.nc : monthly frequency, stomate_ipcc_history.nc : monthly frequency
- *Did all post-treatement job finish?*
  The output from the ATLAS is not correct. This tool has not been adapted to work on a regional domain... If other problem, check in .../IGCM_OUT/.../JobName/Out/ for script output from post-treatement jobs.
- *Where are the time-series stored?*
  .../IGCM_OUT/.../JobName/SRF/Analyse, .../IGCM_OUT/.../JobName/SBG/Analyse