Hands on exercises with ORCHIDEE OFF-LINE

Revised for training session 2015-10-22 - 2015-10-23

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

During the training session everybody works on temporary training accounts at IDRIS.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, choose Applications/machines/ada.

These account do not have acces to ergon which is the archive machine at IDRIS. Therefore today, specify in each config.card to use a directory in the workdir at Ada by setting in the UserChoicies section:

```
ARCHIVE=$WORKDIR/ERGON
```

Install the IPSL environnement at your account during the first connexion to ada. Do the following:

```
cp ~rpsl035/.bash_login $HOME/.
rm $HOME/.bash_profile
source $HOME/.bash_login
```

In order to have priority in the queue system during the training session add in the beginning of each job:

```
# @ class = cours
```

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 extracting 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.

Download a configuration using **./model** *config_name*.

For these exercises will use the configuration ORCHIDEE_trunk which is an offline set up using the latest version of the trunk ORCHIDEE. 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 and XIOS which are fortran and C libraries 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 Platforme groupe at IPSL.
Go into each dircetory and check which versions have been extracted. You're supposed to find same information as you can see in mod.def. Use svn to know version and revision number.

```
cd ../modeles/ORCHIDEE
svn info
cd ../modeles/IOIPSL
svn info
cd ../../libIGCM
svn info
```

The makefile was 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 : Compile at a local PC

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

2.1 First regional run

Create a new directory outside modipsl 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 = -10.
LIMIT_EAST =  30.
LIMIT_NORTH = 70.
LIMIT_SOUTH = 30.
```

Copy or link the netcdf files from the shared repository IGCM into your run directory. The location of the shared repository IGCM depends on the machine but the content is synchronized between the different repositories. You can use export if your shell is bash (for shell tcsh: replace export by set):

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

ln -s $R_IN/SRF/METEO/CRU-NCEP/v5.3.2/twodeg/cruncep_twodeg_1901.nc forcing_file.nc
ln -s $R_IN/SRF/PFTmap_1850to2005_AR5_LUHa.rc2/PFTmap_IPCC_1982.nc PFTmap.nc
ln -s $R_IN/SRF/soils_param.nc .
```

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

```
wget http://dods.extra.cea.fr/work/p86ipsl/IGCM/SRF/METEO/CRU-NCEP/v5.3.2/twodeg/...
   ...cruncep_twodeg_1901.nc forcing_file.nc
wget http://dods.extra.cea.fr/work/p86ipsl/IGCM/SRF/PFTmap_1850to2005_AR5_LUHa.rc2/...
   ...PFTmap_IPCC_1982.nc PFTmap.nc
wget http://dods.extra.cea.fr/work/p86ipsl/IGCM/SRF/soils_param.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. Default values will be used for all parameters not set in run.def. You can find the list of all parameters and their default values in modipsl/modeles/ORCHIDEE/orchidee.default .

The simulation lenght 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 borders(in degrees) for the horizontal domain to be modelize. The default values correspond to the domain of the forcing file. The difference between EAST-WEST and NORTH-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 **libIGCM** is used to chain the executions without manually copy or move of files. libIGCM is a powerful tool but it is still imporant to know how to run the model without it.

2.5 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
```

2.6 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. Coupled to LMDZ it is also possible to run in hybrid parallelization mode with MPI and OpenMP.

You will now launch a global test run on 32 MPI processes (only use 4 at obelix). Prepare a new run directory as in the first exercise but do not put any regional limits in run.def (remove the LIMIT_ parameters). Write a file Job_orchidee as below. This file will be different for different machines. Follow one of the sub sections depending on your machine.

2.6.1 At Ada

Create Job_orchidee with following lines :

```
#!/bin/ksh
# @ job_name = SECHSTOM
# @ job_type = parallel
# @ output = Script_Output_SECHSTOM.000001
# @ error = Script_Output_SECHSTOM.000001
# @ total_tasks = 32
# @ wall_clock_limit = 1:00:00
# @ class = cours
# @ queue

/usr/bin/time poe ./orchidee_ol
```

Note that class=cours can only be used during the training session. Submit the job to the queue with the command llsubmit. Check its running status with the llq or qq. Do following:

```
> llsubmit Job_orchidee       # submit the job
> llq -u login                # check the job's running status
```

2.6.2 At obelix

Create Job_orchidee with following lines :

```
#PBS -N test
#PBS -m a
#PBS -j oe
#PBS -o Script_Output
#PBS -S /bin/ksh
#PBS -l nodes=1:ppn=4

cd $PBS_O_WORKDIR
time mpirun ./orchidee_ol
```

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
```

2.6.3 At curie

Create Job_orchidee with following lines :

```
#!/bin/ksh
#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 32                # Request numbre of cores
#MSUB -T 1800              # Time limit in seconds
#MSUB -Q test              # Queue test, priority acces
#MSUB -q standard
#MSUB -A genXXX            # Set your project id
BATCH_NUM_PROC_TOT=$BRIDGE_MSUB_NPROC
set +x

cd ${BRIDGE_MSUB_PWD}

/usr/bin/time ccc_mprun -n 4 ./orchidee_ol
```

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 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
```

If you've installed the IPSL environnment (.bash_login at ada) you already have rebuild in your path. Try using it which rebuild.
Do the rebuild as follow :

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

2.8 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 half of the 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 difference in time due to the change of number of processors. 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 Work with XIOS

XIOS is the new component used in IPSL models to produce output. Currently XIOS is extracted when installing ORCHIDEE_trunk but the default compiling and simulation set up is done without XIOS. The source code for XIOS is find in modipsl/modeles/XIOS.

3.1 Compiling with XIOS

Go back to modipsl in the directory where you compiled before. In the main makefile a predefined target with_xios exists which compiles ORCHIDEE together with XIOS. To use ORCHIDEE with XIOS the cpp key XIOS must be defined during compilation. This is done by the setting the argument -xios when lauching makeorchidee_fcm. Before compiling, open the Makefile and look at the line with makeorchidee_fcm.

```
cd modispl/config/ORCHIDEE_OL
vi Makefile
gmake with_xios
```

After compilation you'll find also the executable xios_server.exe in modipsl/bin.

3.2 Running ORCHIDEE with XIOS in attached mode

XIOS uses xml files to configure the output. To run ORCHIDEE offline the follwing files are needed: iodef.xml, context_orchidee.xml, field_def_orchidee.xml and file_def_orchidee.xml. These files are found in the ORCHIDEE/src_xml directory.

Now prepare a new run directory as in the exercise for parallel run mode. Copy the xml files to the run directory.

```
mkdir RUNXIOS; cd RUNXIOS
cp ../modipsl/modeles/ORCHIDEE/src_xml/iodef.xml .
cp ../modipsl/modeles/ORCHIDEE/src_xml/context_orchidee.xml .
cp ../modipsl/modeles/ORCHIDEE/src_xml/field_def_orchidee.xml .
cp ../modipsl/modeles/ORCHIDEE/src_xml/file_def_orchidee.xml .
```

Open the xml files and see that attached mode is activated in iodef.xml and that one_file mode is activated in file_def_orchidee.xml. This means that XIOS is used as an library linked in the orchidee_ol executable. It will produce one file. If multiple_file was set instead of one_file then each sub-domain in ORCHIDEE would have an output file and rebuild would have been necessary.

```
# in iodef.xml
<variable id="using_server" type="boolean">false</variable>

# in file_def_orchidee.xml
<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">
```

Now create run.def file for a global run. Activate XIOS and deactivate IOIPSL(WRITE_STEP=0) ouput by adding in run.def :

```
XIOS_ORCHIDEE_OK=y
WRITE_STEP=0
```

Launch the job as before depending on the machine (ada/curie/obelix):

```
llsubmit Job_orchidee / ccc_msub Job_orchidee / qsub Job_orcihdee
```

3.3 Running ORCHIDEE with XIOS in server mode

A more efficient way to run ORCHIDEE with XIOS is to activate server mode. The ORCHIDEE executable and the XIOS server executable is launched in MPMD mode (Multiple Program Multiple Data). In the exercise before in attached mode, ORCHIDEE was launched in SPMD mode (Single Program Multiple Data).

Prepare a new run directory as the exercise before with XIOS. Add this time also the xios server executable. Change to true in iodef.xml to run in server mode:

```
ln -s ../modipsl/bin/xios_server.exe .
vi iodef.xml
  # Change to : <variable id="using_server" type="boolean">true</variable>
```

Create a new job to launch orchidee_ol executable on 31 MPI processes and xios_server.exe executable on 1 MPI process. When having 1 server it does not matter if the option one_file or multiple_file is used. The server will write to the full domaine.

3.3.1 At Ada

Create Job_orchidee_server with following lines(the headings are the same as before) :

```
#!/bin/ksh
# @ job_name = SECHSTOM
# @ job_type = parallel
# @ output = Script_Output_SECHSTOM.000001
# @ error = Script_Output_SECHSTOM.000001
# @ total_tasks = 32
# @ wall_clock_limit = 1:00:00
# @ class = debug
# @ queue

/usr/bin/time poe -pgmmodel mpmd -cmdfile ./run_file
```

Create also the run_file with the following lines (31 lines with orchidee_ol):

```
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./xios_server.exe
```

Now launch the job:

```
llsubmit Job_orchidee
```

3.4 Add a new output variable in ORCHIDEE

Create an output for the variable resp_hetero_litter calculated in stomate_litter.f90 using XIOS. To do this, add in src_stomate/stomate_litter.f90:

```
! Load XIOS in the beginning of the module
use xios_orchidee
...
! Add send to XIOS in the end of the subroutine littercalc
CALL xios_orchidee_send_field("resp_hetero_litter",resp_hetero_litter)
```

Compile in modipsl/config/ORCHIDEE_OL:

```
cd config/ORCHIDEE_OL
gmake with_xios
```

Create a new run directory as before. The variable has dimension DIMENSION(npts,nvm) in the subroutine. The corresponding axis for nvm is called PFT. Add in field_def_orchidee.xml the definition of the new variable:

```
<field id="resp_hetero_litter" name="RESP_H_LITT" long_name="..." unit="?" axis_ref="PFT"/>
```

Add in file_def_orchidee.xml in the section for the file where you want to add the variable:

```
<field field_ref="resp_hetero_litter" level="1"/>
```

Launch as before. Verify that the variable is in the output file.

3.5 Play with the xml files

Do different small runs where you change output frequency, number of variables in the files, activate or deactivate files, change name of the variables in in the output file... Do these kind of changes in file_def_orchidee.xml.

4 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_v6 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 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. The directory FORCESOIL and TESTSTOMATE also follow the same structure but these experiments are not maintained any more as the corresponding propgrams are not maintained in the source code of ORCHIDEE trunk. SPINUP and ENSEMBLE directories contain experiences that are more complicated and are not taught in the course.

We will here work with the OOL_SEC_STO and SPINUP_ANALYTIC experiments.

4.1 Activate XIOS in OOL_SEC_STO experiment

Stay in the modipsl you installed and used previously. For this exercise using XIOS you need to have compiled using *gmake with_xios* as done in the previous exercise. In the set up of the experment change to have XIOS=y in orchidee_ol.card. The experiment will now switch to use XIOS in attached mode instead of IOIPSL for the output. To activate XIOS in server mode you need also to add the component IOS (Input Output Server) in config.card. The orchidee_ol.driver will modify iodef.xml to set server mode when IOS is added in config.card. Use the file config.card.xios_server in OOL_SEC_STO experiment as an example on how to activate IOS. Now copy the directory OOL_SEC_STO into a test directory and modify the cards:

```
cd modipsl/config/ORCHIDEE_OL
cp -r OOL_SEC_STO TestXios
cd TestXios
vi COMP/orchidee_ol.card       # => set XIOS=y
vi config.card
   # Set JobName, DateEnd, SpaceName=TEST, RebuildFrequency=1D, TimeSeriesFrequency=NONE
   # Add IOS in ListOfComponents and Executable, Add section IOS in the end of config.card

   [ListOfComponents]
   SRF= (sechiba, orchidee)
   SBG= (stomate, orchidee)
   OOL= (orchidee_ol, orchidee)
   IOS= (xios, XIOS)
   [Executable]
   SRF= ("", "")
   SBG= ("", "")
   OOL= (orchidee_ol, orchidee_ol, 31MPI)
   IOS= (xios_server.exe, xios.x, 1MPI)

   ...
   [IOS]
   WriteFrequency=""
   Restart= n
   RestartDate=
   RestartJobName=
   RestartPath=
```

Create the main job and check the header of the file. Launch the job.

```
../../../libIGCM/ins_job
llsubmit Job_TestXios  / ccc_msub Job_TestXios  / qsub Job_TestXios
```

Did it work? Probably not if you did the exercise 3.4. In that case you got the following error message in Debug/TestXios_19820101_19820101_out_execution_error file:

```
0-3:> Error [CObjectFactory::GetObject(const StdString & id)] : In file
'/workgpfs/rech/psl/rpsl500/COURSORCH/MYFIRSTTEST/modipsl/modeles/XIOS/src/
object_factory_impl.hpp', line 79 -> [ id = resp_hetero_litter, U = field ]
object is not referenced !
```

You need to adapte the xml files used by the experment. See in COMP/orchidee_ol.card in section ParametersFiles from where the xml files are copied. The file field_def_orchidee.xml is copied directly from ORCHIDEE/src_xml directory. This files is closly related to the version of the model you use. All addition of variables in the source code needs to be reported in this file. The file file_def_orchidee.xml is stored in the experment directory PARAM/. This file contains the set up of the output variables and files for the experiment. It do not need to be updated if you add a file in the source code. But it needs to be updated if you want to write the new variable. Now add in these 2 files the modifications you did in 3.4, clean and relaunch the job:

```
vi ../../../modeles/ORCHIDEE/src_xml/field_def_orchidee.xml
  # Add:
  <field id="resp_hetero_litter" name="RESP_H_LITT" long_name="..." unit="?" axis_ref="PFT"/>
vi PARAM/file_def_orchidee.xml
  # Add:
  <field field_ref="resp_hetero_litter" level="1"/>
../../../libIGCM/clean_month.job
llsubmit Job_TestXios    /  ccc_msub Job_TestXios   /  qsub Job_TestXios
```

Verify that everything went well and that the new output variable is in the output file.

4.2 SPINUP_ANALYTIC experiment

You'll now set up a spinup simulation using libIGCM. Start by coping the SPINUP_ANALYTIC directory for into a new.

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

Look into the config.card. The variables CyclicBegin and CyclicEnd describes the years to loop over. By setting these 2 variables in config.card makes the variable CyclicYear available. CyclicYear can be used in the comp.card. In this exeperiment 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 1MPI on the line for the executable. Do not use server mode for XIOS. Change in orchidee_ol.card to use the CRU-NCEP v5.3.2 twodeg for the forcing file. Look at the shared accout

to see the location exact of these files (see the R_IN variable in exercise 2.)

Create the job and launch the simulation:

```
../../../libIGCM/ins_job
vi Job_JobName # check header lines
llsubmit Job_JobName  /  ccc_msub Job_JobName  / qsub Job_JobName
```

- Where is the share repository IGCM? Where are the CRU-NCEP v5.3.2 stored?
- Where is the output stored? How can you change the place for the ARCHIVE directory? Note that this is recommended only at obelix.
- 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]
```

More indications and answers to questions

Exercise 4.2 with SPINUP_ANALYTIC

See here all commands needed for exercise 4.2:

```
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
OOL= (orchidee_ol, orchidee_ol, 1MPI)
RebuildFrequency=5Y
TimeSeriesFrequency=10Y
SeasonalFrequency=NONE

# Modify COMP/orchidee_ol.card
# Change the path to the forcing file. The new path is
.../CRU-NCEP/v5.3.2/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
../../../libIGCM/ins_job

# Change in the heading of main job Job_MySpin to have specific queue only for training session a
   # @ class = cours
# Set PeriodNb=30
```

Answers to questions in exercise 4.2:

- *Where is the share account? Where are the CRU-NCEP v5.3.2 stored?*

  ```
  At ada:
  /workgpfs/rech/psl/rpsl035/IGCM
  /workgpfs/rech/psl/rpsl035/IGCM/SRF/METEO/CRU-NCEP/v5.3.2

  At curie:
  /ccc/work/cont003/dsm/p86ipsl/IGCM/
  /ccc/work/cont003/dsm/p86ipsl/IGCM/SRF/METEO/CRU-NCEP/v5.3.2

  At obelix:
  /home/orchideeshare/igcmg/IGCM
  /home/orchideeshare/igcmg/IGCM/SRF/METEO/CRU-NCEP/v5.3.2
  ```

- *Where is the output stored? How can you change the place for the ARCHIVE directory? Note that this is recommended only at obelix.*
  At obelix, the output is stored in /home/scratch01/yourlogin/IGCM_OUT/TagName/SpaceName/ExperimentName/JobName. Change this in config.card, UserChoices section: ARCHIVE=/home/yourdisc/yourlogin. It is only the permanent archive that will be changed. This means that if SpaceName=TEST, the output will still be at scratch01. The archive can also be changed directly in libIGCM/libIGCM_sys/libIGCM_sys_obelix.ksh

- *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