

# Hands on exercises with ORCHIDEE OFF-LINE

Revised for training session 2018-11-29 - 2018-11-30  
Josefine Ghattas/IPSL, Fabienne Maignan/LSCE

The goal of these exercises is to learn how to install, compile and launch basic test cases with ORCHIDEE in off-line mode. Exercise 4 uses libIGCM. All exercises can be done at Ada(IDRIS), irene(TGCC) or obelix(LSCE). All commands needed for the basic exercises are listed in the text.

## Before starting

### 0.1 At Ada: today use training account

You can change the keyboard language on the training computer by touching **shift + alt** at the same time. You can also change keyboard with the computers which are not in use to be more comfortable during the training sessions.

During the training session everybody works on temporary training accounts at IDRIS. These accounts have specific priority in the queue to avoid to much waiting time. First connect to **ipcours** using a login **cfor0XX** and the password **IPSLnov18**, then open a terminal and connect to ada via ssh as follow:

```
ssh -Y ada337
```

Note: we advice to change the colors in the terminal window to easier read when using vi. Use the menyboard choose Edit/Preferences.

Install the IPSL-cmc environnement, using the commands below, at your account during the first connexion to ada and add export LL\_RES\_ID. The LL\_RES\_ID is a specific reservation with computing resources available only during the training session and it changes each day. It can not be used later. Do following using the LL\_RES\_ID as written below (can also be known using **llqres**):

```
cd $HOME
cp ~/rps1035/.bash_login .
rm .bash_profile
vi .bash_login
=> add LL_RES_ID in the end of the file. Each day a different value is used

export LL_RES_ID=ada338-ib.203.r      # Thursday 29/11/2018
export LL_RES_ID=ada338-ib.204.r      # Friday 30/11/2018
```

After editing .bash\_login, source the file or deconnecte and reconnecte to ada to take into account the changes.

## 0.2 First time at ada/IDRIS with your own login

When working at ada on your own login you still need to install the IPSL-cmc environnement, as follow:

```
cd $HOME
cp ~/rpsl035/.bash_login .
rm .bash_profile
vi .bash_login
```

Deconnect and reconnect, the .bash\_login will now be sourced by default. Modules needed to compile and run the model are loaded via this file. libIGCM will load the same modules by sourcing the file /smphome/rech/psl/rpsl035/.atlas\_env\_ada\_intel\_2013\_0\_bash which is also sourced from .bash\_login.

## 0.3 First time at irene/TGCC

If you'll start work at irene, then install the IPSL-cmc environnement before the first installation of the model. Do as follow:

```
cp /ccc/cont003/home/igcmg/igcmg/MachineEnvironment/irene/bashrc $HOME/.bashrc
cp /ccc/cont003/home/igcmg/igcmg/MachineEnvironment/irene/bashrc_irene $HOME/.bashrc_irene
```

Deconnect and reconnect, the .bashrc\_irene will now be sourced by default. Modules needed to compile and run the model are loaded by via this file. The modules are specified in the file /ccc/cont003/home/igcmg/igcmg/MachineEnvironment/irene/env\_irene. When running with a script, the same modules need to be loaded. This is done automatically by libIGCM. When you work without libIGCM, you must source the file env\_irene from your job, see job example in the appendix.

## 0.4 First time at obelix/LSCE

At obelix, you need to load following netcdf module before compilation and in your script for running (you can add it to your login environnement, for example in .cshrc depending on the shell used):

```
module load netcdf/4p
```

When running with libIGCM, following file is sourced which contains the above module:

```
/home/orchideeshare/igcmg/MachineEnvironment/obelix/env_atlas_obelix
```

## 0.5 Short guide to use editor vi

In these exercises when it says *vi filename* it means open the file with an editor of your choice. You can use for example **vi** or **emacs**. **vi** is a text editor program which can be used in a terminal window to open and edit ascii files. emacs is also an text editor which can be opened in a separate window. See here some very basic commands to use vi:

```
vi filename      # open the file
/toto           # search for toto in the file. Use n for next occurences.
i              # open insert mode. You can now edit the file
escap          # close insert mode
:w             # save the file
:q            # close the file
:q!          # close the file without saving anything you did since last :w
:wq         # save and close
:syntax off  # to desactivate colors if needed
nG          # go to line n in the file
```

# 1 Install and compile

## 1.1 Install ORCHIDEE trunk for offline use

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:

```
cd $WORKDIR
mkdir TESTOFFLINE; cd TESTOFFLINE
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 you 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 beging with ORCHIDEE\_trunk and then extract as follow:

```
vi mod.def          # Explore lines beginning 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 Plat-forme groupe at IPSL.

Go into each directory 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 ../IOIPSL/src
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 recognizes the following machines: irene at TGCC, ada at IDRIS, obelix at LSCE and ciclad and climserv at IPSL. 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 the 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` and `xios_server.exe` in `modipsl/bin`.

## 1.2 Optional: Install a branch of ORCHIDEE

Do as follow if you would like to install a specific branch instead of the trunk of ORCHIDEE. Note you can only do this for branches where you have read acces. Install first `modipsl` as before. Then open the file `mod.def` and look for the section `ORCHIDEE.trunk`. Change the line containing `ORCHIDEE/trunk` into the name for the branch you will use.

For exemple, to extract `ORCHIDEE-MICT`, do the following:

```
cd $WORKDIR; mkdir TESTMICT ; cd TESTMICT
svn co http://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl
cd modipsl/util
```

In `mod.def`, change the line

```
#-C- ORCHIDEE_trunk trunk/ORCHIDEE HEAD 14 ORCHIDEE modeles
```

into

```
#-C- ORCHIDEE_trunk branches/ORCHIDEE-MICT/ORCHIDEE HEAD 14 ORCHIDEE modeles
```

Finally extract as before:

```
./model ORCHIDEE_trunk
```

Note that `branches/ORCHIDEE-MICT/ORCHIDEE` can be changed to another path on the svn repository, for exemple `branches/ORCHIDEE-CN-CAN/ORCHIDEE` or `branches/ORCHIDEE-SOM/ORCHIDEE`. It can also be the path to a personal version.

You can also set a specific revision number by changing `HEAD` into a revision number.

## 2 Test simulations

We will now do some test simulations using the ORCHIDEE trunk offline installation. 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 the run.def file)
- PFTmap.nc : vegetation map
- woodharvest.nc : woodharvest map
- soils\_param.nc : initialization of soil parameters
- alb\_bg.nc : background albedo from Modis
- iodef.xml, context\_orchidee.xml, context\_input\_orchidee.xml, field\_def\_orchidee.xml, file\_def\_orchidee.xml: parameter files for output and input settings using XIOS
- Optional: routing.nc, floodplains.nc, soils\_param\_usda.nc

### 2.1 First regional run

In this exercise you will set up a simple test case and run in sequential mode without using libIGCM. Create now a new directory outside modips1 to run the model and copy or link the ORCHIDEE executable:

```
cd TESTOFFLINE; mkdir RUN1; cd RUN1
ln -s ../modips1/bin/orchidee_ol .
```

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

```
TIME_LENGTH=31D
NVM=15
PFT_TO_MTC=1,2,3,4,5,6,7,8,9,10,11,12,13,10,10
LIMIT_WEST = -10.
LIMIT_EAST = 30.
LIMIT_NORTH = 70.
LIMIT_SOUTH = 30.
```

Copy xml files from models/ORCHIDEE/src\_xml directory into the run directory.

```
cp ../modips1/modeles/ORCHIDEE/src_xml/* .
```

file\_def\_orchidee.xml describes the output files, frequencies and variable contents. Change all occurrences of `_AUTO_` according to the comments in the beginning of the file. Using libIGCM, these variables marked with `_AUTO_` are modified by the drivers. We suggest for these exercises to set daily output frequency. You can activate all files. Read more about the xml files in the appendix.

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 `tsh` shell: replace `export` by `set`):

```

# At Ada:
export R_IN=/workgpfs/rech/psl/rpsl035/IGCM
# At irene:
export R_IN=/ccc/work/cont003/igcmg/igcmg/IGCM
# At obelix:
set R_IN=/home/orchideeshare/igcmg/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/PFTMAPS/CMIP6/ESA-LUH2v2/historical/15PFT.v1/PFTmap_1901.nc PFTmap.nc
ln -s $R_IN/SRF/WOODHARVEST/LUH2v2/historical/woodharvest_1901.nc woodharvest.nc
ln -s $R_IN/SRF/cartepente2d_15min.nc .
ln -s $R_IN/SRF/SOIL/soils_param.nc .
ln -s $R_IN/SRF/albedo/alb_bg_modisopt_2D_ESA_v2.nc alb_bg.nc
ln -s $R_IN/SRF/reftemp.nc .
ln -s $R_IN/SRF/ROUTING/routing.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 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.nc sechiba_out_2.nc sechiba_history_4dim.nc
rm stomate_history.nc stomate_ipcc_history.nc
rm out_*
./orchidee_ol

```

## 2.3 Continue a simulation using restart files

The model writes restart files in the end of each execution period, after a the time set by `TIME_LENGTH`. These files contain all state variables needed to continue a simulation without losing 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.nc sechiba_history_month01.nc
mv stomate_history.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 manual copy or move of files. libIGCM is a powerful tool but it is important to know how the model works.

## 2.4 Visualization with ferret

Here is a small example to visualize sechiba\_history.nc using ferret.

```
> ferret
use sechiba_history.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
```

## 2.5 Change output levels

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 the output file, etc. Do these kinds of changes in file\_def\_orchidee.xml. Read more about xml files in the appendix.

## 2.6 Add a new output variable in ORCHIDEE

Create a diagnostic output variable 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 modipls/config/ORCHIDEE\_OL:

```
cd config/ORCHIDEE_OL
gmake
```

Create a new run directory as before. The variable has dimension DIMENSION(npts,nvm) in the subroutine. The corresponding axis for nvm has id=nvm. Add in field\_def\_orchidee.xml the definition of the new variable:

```
<field id="resp_hetero_litter" name="RESP_H_LITT" unit="gC/m^2/s" grid_ref="grid_nvm"/>
```

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" grid_ref="grid_nvm_out" level="1"/>
```

Use **svn diff** in modeles/ORCHIDEE to see your modifications in the code. Launch as before. Verify that the variable is in the output file.

It is recommended to modify the field\_def\_orchidee.xml in the model directory ORCHIDEE/src\_xml as well as the file used in the run directory.

### 3 Test in parallel run mode

The default parallelization mode for ORCHIDEE offline configuration is MPI. The model can then be run on 1 or several cores MPI. Coupled to LMDZ, the default parallelization mode is hybrid mode with mixed MPI and OpenMP.

Output diagnostic files are written using XIOS by default. It is also possible to use IOIPSL but this method is not longer maintained. XIOS makes it possible to run either in attached mode or in server mode. When running in attached mode, only the executable `orchidee_ol` is used. XIOS is used as an library in the model. When running in server mode, the executable `xios_server.exe` is launched together with the executable `orchidee_ol`. This is a more complex way to launch the model but it is more efficient when simulating on a big region or with high resolution.

#### 3.1 Running ORCHIDEE with XIOS in attached mode

You will now launch a global test run on 32 MPI processes (only use 8 at obelix) in attached mode. When running in attached mode it is possible to set XIOS to write one global output file (mode `one_file`) or to write partial output files, one per local MPI domain (mode `multiple_file`). When running in `multiple_file` mode, the output files need to be post-processed to contain the full domain. This is done using the tool **rebuild**. This method is not recommended except for debugging or development purpose. We will here use the `one_file` mode.

Prepare a new run directory as in the first exercise but do not put any regional limits in `run.def` (remove the `LIMIT_` parameters). Open the xml files and see that `one_file` mode is activated in `file_def_orchidee.xml`. Also it is not possible to use compression of output when running in attached `one_file` mode. Therefore set `compression=0` in the same file.

```
# in file_def_orchidee.xml
<file_definition type="one_file" ... compression_level="0">
```

In earlier version of XIOS it was necessary to specify in `iodef.xml` if running in attached or in server mode. This is not longer needed as XIOS detects it according to how the model is launched.

Write a file `Job_orchidee` as in the appendix to launch the model in the batch system instead of interactively. The job file will be different for different machines, choose the job for `ada` as in the appendix.

#### 3.2 Check reproducibility of results

When running on 1 or several processors, the results should be the same. Create a new directory and make the same test but on half of the processors. 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.



### 3.3 Optional: 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 are 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 ../modips1/bin/xios_server.exe .
```

Create a new job to launch orchidee\_ol executable on 31 MPI processes and xios\_server.exe executable on 1 MPI process. Note that when using 2 servers (by launching xios\_server.exe on 2 MPI) and option `multiple_file`, then each server will write a partial domain and reconstruction is needed after the run (by using `rebuild`, see further below). 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 domain. See job example in the appendix. Create the `Job_orchidee` file and the run file. Launch the job as before using `lsubmit/ccc_msub/qsub` depending on the machine.

### 3.4 Optional: How to rebuild output if running `multiple_file` mode

Parallel jobs 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 developed at IPSL as an extension of IOIPSL.

`rebuild` is installed at the different machines here:

```
# at obelix:
/home/users/igcmg/rebuild/bin/
# at irene:
/ccc/cont003/home/igcmg/igcmg/Tools/irene/rebuild/modips1_IOIPSL_PLUS_v2_2_4/bin
# at Ada:
/linkhome/rech/psl/rps1035/bin
```

If you've installed the IPSL environment (`.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*
```

## 4 Simulations using libIGCM

The following exercises will now use the ORCHIDEE\_trunk configuration with libIGCM.

There are some differences between ORCHIDEE\_trunk configuration and the coupled configurations such as LMDZOR\_v6. In the configuration ORCHIDEE\_trunk 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\_FG\*, OOL\_SEC and SPINUP\_ANALYTIC\_FG1. 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 programs are not maintained in the source code of ORCHIDEE trunk. SPINUP and ENSEMBLE directories contain experiences that are more complex and are not taught in the course.

We will here work with the OOL\_SEC\_STO\_FG2 and SPINUP\_ANALYTIC\_FG1 experiments.

### Specific system option during the training course

To help the post-treatment job to start run easier, you can set the time limit in the beginning of the job to a lower value for short exercises like we do here. Change in libIGCM/AA\_create\_ts in the section ada to have:

```
#-Q- ada # @ wall_clock_limit = 1:00:00
```

If the create\_ts.job already exists, then remove it. It will be recreated when you run ins.job.

### 4.1 Some parameters are changed by the comp.driver

According to the choices made in config.card and the *comp*.card(*comp* stands for the components: orchidee\_ol.card, sechiba.card, stomate.card), some parameters will be changed in the run.def or in the file\_def\_orchidee.xml. The modifications are done by the *comp*.driver (orchidee\_ol.driver, sechiba.driver, stomate.driver). The parameters who might be modified are always marked as AUTO or AUTOBLOCKER

- AUTO: These parameters can be changed using options in comp.card or config.card. You can also change them directly in the PARAM/run.def or modeles/ORCHIDEE/src\_xml/file\_def\_orchidee.xml, in that case the drivers will not change them again.
- AUTOBLOCKER: The job will stop if you modify these parameters. They are set by the comp.driver mainly using the information from config.card.

For example, in PARAM/run.def:

```
STOMATE_RESTART_FILEIN = _AUTOBLOCKER_  
VEGET_UPDATE = _AUTO_
```

#### Exercise

Go into OOL\_SEC\_STO\_FG2 experiment directory, open PARAM/run.def and search for variables marked AUTO and AUTOBLOCKER. Try to find out which are the variables and using which options can they be changed from the config.card or comp.card.

## 4.2 SPINUP\_ANALYTIC experiment

You'll now set up a spinup simulation using libIGCM. Start by coping the SPINUP\_ANALYTIC\_FG1 directory:

```
cd modips1/config/ORCHIDEE_OL
cp -r SPINUP_ANALYTIC_FG1 MyTestSpinup
cd MyTestSpinup
```

Look into the config.card. The variables CyclicBegin and CyclicEnd describe the years to loop over. Setting these 2 variables in config.card makes the variable CyclicYear available. CyclicYear is used in the orchidee\_ol.card to copy the forcing file. For this exercise loop over years 1901-1910. Check CyclicBegin and CyclicEnd and modify if necessary. Limit the region to a grid-cell. Set in run.def:

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

Set up the simulation to run over 4 forcing periodes (40 years in total). In config.card, set DateEnd=1940-12-31. Set SpaceName=TEST to deactivate pack post treatment if running at curie/ada. Set TimeSeriesFrequency=20Y and SeasonalFrequency=NONE.

It is important to adjust the numer of cores used (number of MPI and OMP) to the domain which is used. Here we run on 1 grid-cell and therefor set 1 MPI on the line for the executable to run in sequential mode. You also need to deactivate XIOS server by removing the lines for IOS in ListOfComponents and Executable. Increase PeriodNb before launching the job.

- Why should you deactivate XIOS in server mode in this example?
- How do you calculate PeriodNb?
- Which forcing file is used and where is it stored? Where is the shared repository IGCM?
- 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 differences?
- 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_19401231_1Y_CARBON_ACTIVE.nc
use JobName_19010101_19401231_1Y_CARBON_SLOW.nc
use JobName_19010101_19401231_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]
```

### 4.3 OOL\_SEC\_STO experiment

Currently 4 different OOL\_SEC\_STO experiments exist. FG2 and FG3 are only different on the forcing files, both are historical set up with land use change activated. FG3nd is the same as FG3 but it uses the driver orchideedriver which is under development. FG1trans is to be used as a transient simulation between the SPINUP\_ANALYTIC\_FG1 and historical OOL\_SEC\_STO\_FG2.

Set up an experiment with sechiba and stomate using the experiment directory OOL\_SEC\_STO\_FG2.

- Set up the simulation by period of 1 month for a total simulation length of 3 months.
- Activate the option for floodplains in run.def. Look in the file orchidee.default in the source directory to have the exact name for the parameter activating this option.
- Add the input file floodplains.nc if needed. This file is only needed for the first period. The information will be stored in the restart file. Search in the shared repository IGCM (directory R\_IN) to find the input file.
- Output the variable floodplains in 2 files by frequency 1 month and 1 day. Look into sechiba.card and read comments to know how to set output frequencies.

Prepare the rest of the job as usual. Launch the test and analyse the results.

## 5 SVN

This exercise aims to learn to the basic use of svn. The setup is only available at ada during the training session. Type the commands listed below and answer questions.

### Determine version and revision

```
cd /workgpfs/cours/for/cfor028/TP/ORCHIDEE/modipsl/modeles/ORCHIDEE.2
svn info
```

Which version and which revision of ORCHIDEE is installed in this directory?

```
svn stat
svn diff
```

Are there any local modifications done after download? Using `svn stat`, the local so called working version is compared to the version and revision that was extracted (see information in `svn info`). Use `svn help` to know more about the commands `svn`, for example `svn help stat`.

```
svn -u stat
```

Are there any modifications done on the server more recent than your version? What is the difference when you add `-u`?

```
cd /workgpfs/cours/for/cfor001/TP/ORCHIDEE/modipsl/modeles/ORCHIDEE.3
```

Answer the same question for this version. Which version and which revision is installed here? Are there any local modifications? Modifications done later on the server?

Note that the use of `svn` is based on `.svn` folders stored in each directory and each sub-directory. If these directories are not there, it is not possible to get information about the version used. Therefore, when you copy a local version of ORCHIDEE, always copie all sub-folder using `cp -r`.

### Clean, update and resolve conflicts

```
cd $WORKDIR
cp -r /workgpfs/cours/for/cfor001/TP/ORCHIDEE/modipsl/modeles/ORCHIDEE.3 myORCHIDEE.3
cd myORCHIDEE.3
svn stat
svn diff
```

Now you've copied the version ORCHIDEE.3 into your workdir. Is this version modified? Is it up to date or has it been changes on the server? You will now clean and update this version. First remove modifications which are not needed and update to the latest version. You will need to resolve conflicts.

```
svn revert file_where_modifications_are_not_needed_to_be_saved.f90
svn -u stat
```

You will now update the full ORCHIDEE to the latest revision on the current version(path on `svn`). Always do this command from the basedirectory ORCHIDEE. Use `svn -u stat`. All files marked with `*` (star) will be updated. Files with `*` and `M`, might make conflicts.

```
svn update      # Answer p if confict is detected
svn stat
```

If a file is marked `C`, it means conflict. You have to resolve the conflict manually. Open the file and try to resolve the conflict. When this is done, tell `svn` that the conflict is resolve.

```
vi file_with_conflict      # look for sections with <<<<<<
svn resolved file_with_conflict  # tell svn conflict is resolved
svn diff file_with_conflict
```

## 6 Read parameter from run.def file using getin

### 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 for each parameter are set in the model fortran code and they 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 variables in run.def are read from the model using a call to getin\_p which is an interface taking different types of variables.

- **TIME\_LENGTH** gives the simulation length for each execution. 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. In global or regional simulations, we do not advice to run more than 1 year per execution but for site simulations it is recommended to run the full forcing file length in one execution.
- **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
```

Note that the current version of ORCHIDEE have problems to detect these kind of errors when running in parallel mode.

- **STOMATE\_OK\_STOMATE** parameter activates coupling to the stomate in ORCHIDEE, default value is TRUE.

### Exercises

- Search in the model to find where the parameter **STOMATE\_OK\_STOMATE**. In which file is the parameter read?
- Open ORCHIDEE/src.sechiba/hydrol.f90 and see how the variable frac\_froz\_corr is treated.
- Which parameter such be used to change default value to prescribe the atmospheric CO2 and which is the default value? Does the default value set in orchidee.default correspond to the value set in the code?
- In thermosoil.f90, the variables **THKICE**, **THKQTZ** and **THKW** are set in the code and can not be changed from run.def. Make this possible! The default values should be the same as before. Add a write to see the values used during run time. Change the code, recompile and run with default and modified values. Make sure that the modified values are taken into account.

## 7 Appendix

### 7.1 Description of xml files

The xml files are used to configure the output files when using XIOS. The xml files are stored in ORCHIDEE/src\_xml directory. When running the model using libIGCM, the file\_def\_orchidee.xml is changed for all occurrences of the keyword `_AUTO_`. The following 5 files are needed for ORCHIDEE:

- `iodef.xml`: this file is the first file read by XIOS. In this file the 2 following files named `context_*orchidee.xml` are include. When running in coupled mode, the other modeles `context_XXX.xml` files are also included from here.
- `context_orchidee.def`: containing axis and grid information
- `context_input_orchidee.def`: containing information about input files and related grids. This file is mandatory in the trunk since revision 5565 (since 09/11/2018) but not yet present in all branches.
- `field_def_orchidee.xml`: contains one line per output variable sent from the model. This file is only changed if new output are added in the model. A variable is output from the model with a call to subroutine `xios_orchidee_send_field`.
- `file_def_orchidee.xml`: contains specifications about the output files and contents. This is the file to be changed for all modifications in the output settings. This file is modified by `orchidee.ol.driver` when running with libIGCM. It is only modified where the keyword `_AUTO_` is set. You can change the `_AUTO_` as you wish and make other changes according to your needs, they will never be overwritten. When running without libIGCM you must change all `_AUTO_`, read comments in the begging of the file.

### 7.2 Example of a parallel job files

#### 7.2.1 Job using XIOS in attached mode at ada

This is a job file example for running orchidee with XIOS in attached mode at ada. Create `Job_orchidee` with following lines:

```
#!/bin/ksh
# @ job_name = test
# @ job_type = parallel
# @ output = Script_Output
# @ error = Script_Output
# @ total_tasks = 32
# @ wall_clock_limit = 1:00:00
# @ queue
#
# Execute the model
/usr/bin/time poe ./orchidee_ol
# End of file
```

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
```

### 7.2.2 Job using XIOS in attached mode at obelix

This is a job file example for running orchidee with XIOS in attached mode 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 -q shortp
#PBS -l nodes=1:ppn=8
#
# Source module
. /home/orchideeshare/igcmg/MachineEnvironment/obelix/env_atlas_obelix
# Go to current folder and execute
cd $PBS_O_WORKDIR
time mpirun ./orchidee_ol
# End of file
```

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
```

### 7.2.3 Job using XIOS in attached mode at irene/TGCC

This is a job file example for running orchidee with XIOS in attached mode at irene/TGCC. 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 nombre of cores
#MSUB -T 1800         # Time limit in seconds
#MSUB -q skylake
#MSUB -m store,work,scratch
#MSUB -A genXXX       # Set your project id. Most people working with
#                     # ORCHIDEE at LSCE belong to project gen6328
# Source modules
. /ccc/cont003/home/igcmg/igcmg/MachineEnvironment/irene/env_irene
# Go to current folder and execute the model
cd ${BRIDGE_MSUB_PWD}
/usr/bin/time ccc_mprun -n 32 ./orchidee_ol
# End of file
```

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
```



#### 7.2.4 Job using XIOS in server mode at ada

To run in server mode you need to create 2 files: main job and a run\_file. Create Job\_orchidee\_server with following lines(the headings are the same as before):

```
#!/bin/ksh
# @ job_name = test
# @ job_type = parallel
# @ output = Script_Output
# @ error = Script_Output
# @ total_tasks = 32
# @ wall_clock_limit = 1:00:00
# @ queue
/usr/bin/time poe -pgmmodel mpmd -cmdfile ./run_file
# End of 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
./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
```