Hands on exercises with ORCHIDEE OFF-LINE

Updated for training session 2023-02-06 - 2023-02-07

Josefine Ghattas/IPSL, Sebastiaan Luysaert/VU,
Bertrand Guenet/LG-ENS, Fabienne Maignan/LSCE,
Xiaoni Wang-Faivre/LSCE

Contents

1 Before starting

The goal of these exercises is to learn how to install, compile and launch basic test cases with ORCHIDEE in off-line mode. All exercises can be done at Jean-Zay(IDRIS), irene(TGCC), obelix(LSCE) or spirit(x)(IPSL ESPRI-mesocenter). All commands needed for the basic exercises are listed in the text.

1.1 Some notes for jean-zay/IDRIS

The compilation should be done from jean-zay-pp. The installation can be done from jean-zay or jean-zay-pp. The same files system is seen from both frontal machines.

1.2 Some notes for obelix/LSCE

While using libIGCM, the IGCM_OUT folder with the output are stored by default in /home/scratch01/login. This is not a peremanent folder but it is set like this in libIGCM because it's the only disk that all users have access to (except /home/users but it is too small). You can tell libIGCM to create the IGCM_OUT folder on another disk by adding ARCHVE in config.card, in the section [UserChoices]. For example set ARCHIVE=/home/orchidee01/yourlogin if you have access to the disk orchidee01.

When you run with libIGCM, the executaion of the model is done in a /tmp folder which you can not see at obelix. If you want to see the execution folder, it can be changed. You can uncomment the variable RUN_DIR in main Job with libIGCM and set it for example as following (change login to your personal login) :

```
mkdir /scratchu/jgipsl/RUN_DIR/231628_1331065/MyTestSpinup.1331065
vi Job_MyTest  => set RUN_DIR=/home/scratch01/login/RUN_DIR
```

You only need to create the RUN_DIR folder the first time. libIGCM will create a folder per job inside this folder. If the job finish well, the folder inside RUN_DIR will be removed.

If you want to work with LMDZ at obelix(not in this training), you can for example install the configuration LMDZOR_v6_work. Installation is done using the standard procedure. Only before running, you need to deactivate the FFT filter in LMDZ because no FFT was set during compilation. You also need to use less cores for running (use 32MPI with 1OMP). Change as follow:

```
vi PARAM/run.def
   => Change to have : use_fft_filtre=n
vi config.card
   => Change in section [Executables] to have :
      ATM= (gcm_${ResolAtm}_${OptMode}.e, lmdz.x, 31MPI, 1OMP)
      IOS= (xios_server_${OptMode}.exe, xios.x, 1MPI)
```

1.3 Some notes for spirit and spiritx/IPSL-ESPRI

At spirit, there are currently error messages while using svn. This is not the case at spiritx. The error messages look like below and dosen't impact the download. It is only the passsword that can not be stored.

```
--- Component : ORCHIDEE
Authentication realm: <svn://forge.ipsl.jussieu.fr:3690> f489ceea-5127-0410-b15c-c4a6149ed9a7
Password for 'josefine.ghattas': *******

Couldn't start kwalletd:  QDBusError("org.freedesktop.DBus.Error.Spawn.ChildSignaled", "Process org.kde.
Couldn't start kwalletd:  QDBusError("org.freedesktop.DBus.Error.Spawn.ChildSignaled", "Process org.kde.
Couldn't start kwalletd:  QDBusError("org.freedesktop.DBus.Error.Spawn.ChildSignaled", "Process org.kde.
Couldn't start kwalletd:  QDBusError("org.freedesktop.DBus.Error.Spawn.ChildSignaled", "Process org.kde.
Couldn't start kwalletd:  QDBusError("org.freedesktop.DBus.Error.Spawn.ChildSignaled", "Process org.kde.
org.kde.kwindowsystem: Could not find any platform plugin
Couldn't start kwalletd:  QDBusError("org.freedesktop.DBus.Error.Spawn.ChildSignaled", "Process org.kde.
Checked out revision 7879.
```

If you want to work with LMDZ at spirit or spiritx(not in this training), you can for example install the configuration LMDZOR_v6_work. Installation is done using the standard procedure. You just need to use less cores for running (use 32MPI with 1OMP). Change as follow:

```
vi config.card
   => Change in section [Executables] to have :
      ATM= (gcm_${ResolAtm}_${OptMode}.e, lmdz.x, 31MPI, 1OMP)
      IOS= (xios_server_${OptMode}.exe, xios.x, 1MPI)
```

1.4 Using training accounts at spirit

If you don't have a login to spirit or obelix, we'll distribute a tempoary login for you. Please connect first to following server using this login:

```
ssh -X login@spirit-tp.ipsl.fr
```

Then connect to spirit using the following:

```
ssh spirit-su-tp
```

Go to /scratchu/login and start the exercises.

1.5 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. Here are some very basic commands to use vi. What is set after the sign # are comments.

```
vi filename    # open the file
/toto          # search for toto in the file. Use n for next or
               # ? for preivous occurence of the word.
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
G              # go to the end of the file
1G             # go to the beginning of the file
nG             # go to line n in the file
```

Instead of vi, **emacs** is another text editor that can be used. It can be opened in a separate window and it has a menu board which might be easier to use. Open as follow:

```
emacs filename &
```

You can use the file menu to save and close the file.

2 Install and compile

2.1 Install ORCHIDEE trunk for offline use

We are first going to download **modipsl** and explore what is inside. modipsl contains some tools in the directory **util**. In util, scripts are found for extraction (*model and mod.def*). It also conatins files for creation of makefiles which were used in older version of the configurations (*ins_make, AA_make.gdef*). modipsl is also an empty file tree that will receive the models and tools when downloading a configuration.

These exercices can be done on the scratch of your machine as it is only temporary. For exemple at obelix install in /home/scratch01/yourlogin, at irene in $CCCSCRATCH or at jean-zay in $SCRATH. At spirit install in /scratchu/yourlogin and at spiritx in /homedata/yourlogin at climserv.

Start this exercise by extracting modipsl in a new directory as follow:

```
cd "your scratch folder as described above"
mkdir TESTOFFLINE; cd TESTOFFLINE
svn co --username icmc_users 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.

For these exercises you will use the configuration ORCHIDEE_trunk which is an offline set up using the latest revision of the trunk ORCHIDEE. Open mod.def file and look at lines beginning with ORCHIDEE_trunk, close the file without any changes and then extract the model as follow::

```
./model ORCHIDEE_trunk
```

Compilation is done using a script specific to the configuration. Note that at jean-zay the compilation takes to much resources and often crash in XIOS. Therefore, connect to jean-zay-pp for the compilation. In older versions of the model, makefiles where used instead. You'll find more information on the ORCHIDEE wiki if you need to compile with makefiles. Now compile the model using the script:

```
cd ../config/ORCHIDEE_OL
./compile_orchidee_ol.sh
```

The compilation takes between 12-15 minutes but maybe during the training it takes longer if lots of people compile at the same time. While the compilation is on going, open a new terminal, connect again to your account and continue the exercises.

The compilation script detects on which machine you are working and uses the corresponding predefined compiler options. By default the compilation script recognizes the following machines: irene-skl and irene-amd at TGCC, jean-zay at IDRIS, obelix at LSCE and spirit and spiritx at ESPRI mesocenter at IPSL. If you want to compile on another target machine you have to add files with the compiler and libraries specific to your machine in the arch folders in each model directory and in the config directory. This is described on the wiki igcmg_doc where the ICMC-tool chain is described, see here for the compilation:
http://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters/LocalPC

When the compilation is finished you will find the executables orchideedriver, orchidee_ol and xios_server.exe in modipsl/bin folder. orchideedriver and orchidee_ol are both executables for the full ORCHIDEE model. The differences are only the driver part controlling the reading of the forcing files. orchideedriver is also called "the new driver" and orchidee_ol is called "the old driver". Both can be used with only some small differences in the set up related to the forcing file.

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 and output issues. In directory modipsl/config/ORCHIDEE_OL you find, beside the compilation script, folders to run ORCHIDEE using libIGCM. libIGCM is a tool developed at IPSL to run coupled and off-line simulations. Specific training sessions about libIGCM are given by the Plat-forme groupe at IPSL.

Go into each directory (using **cd** and **ls**) and check which versions have been extracted. You're supposed to find the same information as you can see in mod.def. Use svn to know the version and the revision number.

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

If the compilation is still ongoing, you can take the waiting time to do **Exercises to learn SVN**.

2.2 Optional: Install a branch of ORCHIDEE

Do not this exercise during the training session. 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 util/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:

```
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
```

Finaly extract as before:

```
./model ORCHIDEE_trunk
```

Note that *branches/ORCHIDEE-MICT/ORCHIDEE* can be changed to another path on the svn repository, for example *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.

3 First test simulation without libIGCM

We will now do some test simulations using the ORCHIDEE trunk offline installation you just compiled. This will first be done interactively which means to launch directly in the terminal without passing trough the batch system. In other words, launch directly "./orchideedriver". libIGCM is used here only to create the test case but not to launch the model. We want you first to understand how the model works and what is needed as input files.

3.1 Prepare a test run directory

To prepare the test case, we'll use a predefined experiment in the config folder. We'll use libIGCM to prepare the test case but we'll not use libIGCM to run the model. The option DRYRUN=4 in libIGCM enables creation of the run folder without launching the model. Do as follow:

```
cd modipsl/config/ORCHIDEE_OL
cp -r OOL_SEC_STO_FG2nd TestDRYRUN
cd TestDRYRUN
```

The file config.card contains the main settings for the experiment. Change this file only for the JobName and create the job. You can choose the name of JobName as you like but it is good if it is the same as the folder, here we use TestDRYRYN. You can use vi, emacs or an editor of your choice. Do as follow:

```
vi config.card => Change to JobName=TestDRYRUN
```

Now create the main job, change to DRYRUN=4 and launch it:

```
../../../libIGCM/ins_job
vi Job_TestDRYRUN  =>   Change to DRYRUN=4
                        At obelix, also change the header to
                        #PBS -l nodes=4:ppn=8

at obelix: qsub Job_testDRYRUN
at spririt(x) or jean-zay: sbatch Job_testDRYRUN
at irene: ccc_msub Job_testDRYRUN
```

Using DRYRUN=4, only the run folder will be created. Look in the end of the **Script_Output** file that will be produced, where the run folder has been created.

3.2 Launch the model

Go to the folder created by libIGCM. Everything needed to run the model has been copied to this folder. Look at the different files and try to understand what they are needed for. There are different type of files; netcdf files, parameter .def files, xml files and executables.
You can use ncdump to see what is in the netcdf files. For example:

```
ncdump -h forcing_year.nc
```

Use vi or another text editor to open the parameter files:

```
vi run.def
vi file_def_orchidee.xml
```

Question: How can the model know that this is the forcing file? Hint: look in the run.def.
By default, the model will run on the full domain given by the forcing file. In this first exercises, you can set a pixel or a small domain. Open run.def and use the parameters LIMIT_WEST, LIMIT_NORTH, LIMIT_SOUTH and LIMIT_EAST to define a latitude-longitude region. For example, set following to run on 1 pixel in run.def:

```
LIMIT_WEST=8
LIMIT_NORTH=48
LIMIT_SOUTH=46
LIMIT_EAST=10
```

Before launching the model you need to source the same modules as the those used during compilation. Do as follow by changing path_to_your_modipsl to the path on your machine:

```
source path_to_your_modipsl/config/ORCHIDEE_OL/ARCH/arch.env
```

Note that at obelix, the source of the file arch.env doesn't work interactively. **Instead at obelix, open the file and copy paste the lines module purge and module load directly into the terminal.**

Now launch the model:

```
./orchideedriver
```

When the execution is completed correctly, following log message is found in the output text file out_orchidee_0000:

```
 END of orchideedriver
```

3.3 Relaunch in the same directory

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

```
rm *rest_out.nc
rm *history*
rm out_*
./orchideedriver
```

3.4 Introduce an error

Make one test while you remove one of the input files. Don't forget to clean the restart files before relaunching. Do as follow:

```
mv soils_param.nc soils_param.nc.BACKUP
./orchideedriver
```

What happened? In which file do you see the error message?

Move back the file before continuing with the exercises.

```
mv soils_param.nc.BACKUP soils_param.nc
```

3.5 Change simulation length and output level

You can change the length of the simulation in run.def using the parameters START_DATE and END_DATE. Note that the simulation can not be longer than one year using global forcing files. If you did another test using the old driver orchidee_ol, the parameter TIME_LENGTH is used instead.

The level of output is set in file_def_orchidee.xml. Each output file has a section starting with **file id**. 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. Read more about xml files in the appendix (A.1).

4 First test simulation using libIGCM

In the previous exercise you launched the model ORCHIDEE without any script. It was done in sequential mode, without the XIOS server and without re-use of the restart files. So it was a very limited run. You'll now use libIGCM which is a script library which help us to set up and launch longer and more complex simulations with post-treatmenets.
During the training session, ask as much as you want to learn more about libIGCM. For the rest of the year, you have the full documentation to libIGCM and the tools to run the IPSL configurations here: **http://forge.ipsl.jussieu.fr/igcmg_doc** . We advice to to follow the dedicated training course.

If you already used libIGCM you will see that 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.

We'll now make a first test for one day using the experiment set up in OOL_SEC_STO_FG2nd. This experiment is a global historical set up using the full ORCHIDEE model in offline mode. It uses the driver orchideedriver also called "the new driver". You'll launch 1 day using the default set up.

4.1 Prepare the experiment folder and create the job

Do as follow:

```
cd modipsl/config/ORCHIDEE_OL
cp -r OOL_SEC_STO_FG2nd MyFirstTest
cd MyFirstTest
```

The file config.card contains the main settings for the experiment. Change this file and create the job. You can use vi, emacs or an editor of your choice. Do as follow:

```
vi config.card => Change to have the following:
                    JobName=MyFirstTest
                    DateEnd=1901-01-01
                    PeriodLenght=1D
                    OOL=  (orchideedriver_${OptMode}, orchideedriver, 15MPI)
                    TimeSeriesFrequency=NONE
                    SeasonalFrequency=NONE
```

By default, monthly mean output are set. For this test case, we want daily output. Therefor modify in sechiba.card and stomate.card.

```
vi COMP/sechiba.card => Change to:  output_freq_sechiba_history = 1d
vi COMP/stomate.card => Change to:  output_freq_stomate_history = 1d
                                    output_freq_stomate_ipcc_history = 1d
```

Now create the main job:

```
../../../libIGCM/ins_job
```

4.2 Launch the job

When working on a cluster you need to know how to subitt and control a job on the queue system. Different clusters have different systems to handle this.

	obelix	spirit(x) and jean-zay	irene
Submit a job	qsub Job	sbatch Job	ccc_msub Job
Check job status	qstat -u $USER	squeue -u $USER	ccc_mstat -u
Deleate a job	qdel JobID	scancel JobID	ccc_mdel JobID

Now do the following, using the commands in the table corresponding to your machine:

- Using ins_job, the main job was created. Launch it using the correct command from the table above. For example at obelix : qsub Job_MyFirstTest
- Use the command to check job status to see if the job is already running.
- When the job finished running, you'll find a file run.card in the folder and also a Script_Output_MyFirstTest. Look at these files.
- Search the variable IGCM_OUT in the Script_Output_MyFirstTest file to find where the output are stored.

4.3 Continue the simulation

Now you should continue same simulation for 3 more days. First be sure that the first day finished correctly. For this, look into run.card, you should find **PeriodState= Completed**.
Now change in config.card the DateEnd:

```
vi config.card => Change only the following: DateEnd=1901-01-04
```

Before relaunching the job you need to prepare the run.card (because it says Completed). Use as follow the script clean_PeriodLength.job and answer yes to the questions:

```
../../../libIGCM/clean_PeriodLength.job
```

You can now launch again as the first time. For example at spirit or jean-zay:

```
sbatch Job_MyFirstTest
```

Look at the output in IGCM_OUT folder. All output from the test simulation are found in a folder IGCM_OUT/OL2/SpaceName/ExperimentName/JobName where SpaceName, ExperiementName and JobName are set in config.card. They can be changed for the next simulation if you want.

4.4 Visualization with ferret

The file format for the output files are netcdf (suffix .nc). You need specific tool to visualize the output. For example ferret can be used. Here is a small example to visualize sechiba_history.nc using ferret. The signe **!** is interpreted as comment in ferret.
For this example, we use the file sechiba_out_2.nc. First find where the file is stored by looking into the file Script_Output_MyFirstTest.000001. Do the following:

```
> grep sechiba_out_2 Script_Output_MyFirstTest.000001
> module load ferret
> ferret
use "path_to_file/MyFirstTest_19010101_19010101_HF_sechiba_out_2.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
```

4.5 Understand what you were running

When you launched the main job, before executing the model, libIGCM created a temporary run directory (called RUN_DIR), copied and edited all input files needed for the experiment you choose to run. All the input files are listed in the COMP/*card files. You can see the full list of files in the run directory before the model is executed by looking into the Script_Output file. Search for the key word "DIR BEFORE RUN EXECUTION". There are different type of files listed in the subsections below: netcdf files, .def files, .xml files and executables.

4.5.1 Netcdf input files

The meteorological forcing files for this set up are: **forcing_yearm1.nc, forcing_year.nc, forcing_yearp1.nc**. For the new driver, the forcing for current time-step, the time-step before and the time-step after are needed. That's why 3 files are needed. Look in COMP/orchideedriver.card and you can see in section BoundarFiles which files should be copied:

```
[BoundaryFiles]
List=  (${R_IN}/SRF/METEO/CRUJRA/v2.2.2/twodeg/crujra_twodeg_v2.2.2_${year_m1}.nc, forcing_yearm1.nc),\
       (${R_IN}/SRF/METEO/CRUJRA/v2.2.2/twodeg/crujra_twodeg_v2.2.2_${year}.nc, forcing_year.nc),\
       (${R_IN}/SRF/METEO/CRUJRA/v2.2.2/twodeg/crujra_twodeg_v2.2.2_${year_p1}.nc, forcing_yearp1.nc)
```

You can also see in the Script_Output file where the copie of these files were done:

```
IGCM_sys_Cp : /home/orchideeshare/igcmg/IGCM/SRF/METEO/CRUJRA/v2.2.2/twodeg/crujra_twodeg_v2.2.2_1900.nc forcing_yearm1.nc
IGCM_sys_IsFileArchived : /home/orchideeshare/igcmg/IGCM/SRF/METEO/CRUJRA/v2.2.2/twodeg/crujra_twodeg_v2.2.2_1901.nc
IGCM_sys_Cp : /home/orchideeshare/igcmg/IGCM/SRF/METEO/CRUJRA/v2.2.2/twodeg/crujra_twodeg_v2.2.2_1901.nc forcing_year.nc
IGCM_sys_IsFileArchived : /home/orchideeshare/igcmg/IGCM/SRF/METEO/CRUJRA/v2.2.2/twodeg/crujra_twodeg_v2.2.2_1902.nc
IGCM_sys_Cp : /home/orchideeshare/igcmg/IGCM/SRF/METEO/CRUJRA/v2.2.2/twodeg/crujra_twodeg_v2.2.2_1902.nc forcing_yearp1.nc
```

Other input netcdf files needed are **PFTmap.nc** containing the vegetation map and **nfert_pasture.nc, nfert_cropland.nc, nmanure_pasture.nc, nmanure_cropland.nc, ndep_nhx.nc, ndep_noy.nc, FMmap.nc and bnf.nc** which are nitrogen input files. Theses files are listed in COMP/sechiba.card and COMP/stomate.card.

When starting the model the first time, you don't have a restart file, you then need some more input files which are **soil_bulk_and_ph.nc, soils_param.nc, cartepente2d_15min.nc, floodplains.nc, reftemp.nc and routing.nc**. These files are listed in InitialStateFiles in sechiba.card. The files in InitialStateFiles are only copied if there is no restart file.

4.5.2 def files

run.def is the main parameter input file. This is the file containing the parameter set up of the model. This file can include other def files and in the standard set up of ORCHIDEE, the files **orchidee.def and orchidee_pft.def** are also needed. These files are specified in COMP/orchideedriver.card. The files are installed with the model and are copied from the folder PARAM in the experiment set up.

4.5.3 xml files

xml files are needed to parametrize the netcdf output files. The xml files are stored in the model, in folder ORCHIDEE/src_xml. You can see where they are specified in section [ParametersFiles] in the file COMP/orchideedriver.card. The xml files are read by XIOS which is the tool ORCHIDEE use to produce the output files in an optimized way.

4.5.4 Executables

The executable for ORCHIDEE in offline mode is orchideedriver (also called the "new driver") used in this experiment set up. In for example the experiment OOL_SEC_STO_FG2, in stead, the executable orchidee_ol is used ("old driver").
The executable xios_server_prod.exe is also copied and renamamed to xios.x. This executable is created during the compilation from the sources in modipsl/modeles/XIOS folder. XIOS is used to create output in an optimal way. It is optional to run with the XIOS server but it is recommanded. The default set up of all experiments with ORCHIDEE uses XIOS is server mode, and this executable is necessary.

4.5.5 Files created by libIGCM

Finally you'll find some files created by libIGCM to facilitate the execution of the model. These files are for example as run_file, compiler.txt or in some cases script_xios.x.ksh, script_lmdz.x.ksh... These files depend on how the model will be launched and on which machine you're working on. These files wont be discussed here.

4.5.6 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: orchideedriver.card, sechiba.card, stomate.card), some parameters will be changed in the run.def, orchidee.def and in the file_def_orchidee.xml. We advice you to control after the beginning of the simulation that the parameters correspond to what you want in the files seen by the model, after modification by libIGCM. Theses files are stored after run in IGCM_OUT/...../JobName/OOL/Debug folder. The modifications are done by the *comp*.driver (orchideedriver.driver, sechiba.driver or stomate.driver). The parameters who will be modified are always marked as AUTO or AUTOBLOCKER in the original .def or .xml files.

- 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, PARAM/orchidee.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 using the information from config.card.

For example, in PARAM/orchidee.def:

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

It is not possible to add new parameters in the *comp*.cards. If you want to modify a parameter which is not set to AUTO, you should modify it directly in the files in PARAM/*.def or in ORCHIDEE/src_xml/*xml.

4.5.7 Exercise

In your experiment folder, open PARAM/run.def and search for variables marked AUTO and AUTOBLOCKER. Try to find out which are the variables and how they can be changed from the config.card, orchideedriver.card, sechiba.car or stomate.card using the appropriate options.

4.5.8 Output files

During the simulation, several type of output files are created. After the execution finished, they are copied to the output folder IGCM_OUT by libIGCM. You'll find the output organized in IGCM_OUT folder, under a specific folder for each job, as the following:

- Stored in folders SRF/Output and SBG/Output : netcdf output from the model to be analysed. The output can be on different frequencies.
- Stored in folders SRF/Restart and SBG/Restart : the restart files which are needed to continue the simulation. You don't need to look into these files. libIGCM will copy them automatically.
- Stored in OOL/Debug, SRF/Debug and SBG/Debug : text output files and input parameter files. The model writes text output, one file per running core out_orchidee_000x. These files are concatenated by libIGCM after the run and copied to the Debug folder. These files are very interesting especially if the model makes an error and crash.

5 Simulation experiments

The number of simulation experiments is almost unlimited but typical experiments involve running the model with different climate forcing, different atmospheric CO_2 concentrations, different land uses, different land management, and/or different parameters. As is the case with field experiments, a model experiment also requires a control and a treatment. The control is often a simulation that matches the historical changes. This has the advantage that the control can be evaluated against observations. The treatment is often based on a scenario describing what could have happened in the past if this or that would (not) have happened or what may happen in the future.

The results of the simulation experiment to a large extent depend on the quality and realism of the spin-up. In this training we just demonstrate the principle and some main settings but make sure to spend enough time thinking about the spinup and further developing ORCHIDEE such that it can handle the spinup you need (see section 5.1).

5.1 Making a spinup analytic

When you've launched the spinup job in this exercise, go to next chapter Debug while waiting for the results.

When running a simulation experiment one has to be sure that the signal from the experiment, e.g., an increase in plant growth or an decrease in evaporation comes from the experimental treatment and not from the initial conditions of the model. To this aim a model spin-up is used. There are several pools in the model that need to reach equilibrium: soil water content, vegetation biomass and the soil carbon and nitrogen pool. Because the latter pools are the slowest to reach equilibrium, those are used as a criterion to stop the spin-up.

ORCHIDEE now has a nitrogen cycle which makes the spin-up an even more important part of the simulation experiment than before. The vegetation needs soil nitrogen to grow but one important source of soil nitrogen is the decomposition of plant litter. Soil and biomass pools thus depend on each other. It may take as much thinking and creativity to set-up a good model spinup as it takes to design a good simulation experiment. Note that adding new functionality to the model, e.g., P-cycle, forest management, abrupt mortality, ... may require an effort to rethink the spin-up.

A typical spin-up requires about 340 years of simulation which is way too long for this training. For this reason we have selected a faster test case.

Start by copying the SPINUP_ANALYTIC_FG1 directory:

```
cd modipsl/config/ORCHIDEE_OL
cp -r SPINUP_ANALYTIC_FG1nd 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 orchideedriver.card to copy the forcing file. As the spinup should represent the land surface at the start of the actual simulation experiment we often use historical climate files and boundary files. For this exercise loop over years 1901-1910. This implies that the spinup will represent the land surface at the beginning of the 20^{th} century. Check CyclicBegin and CyclicEnd and modify if necessary. Ideally we should use the climate data from 1571 to 1910 to spinup the model and reach equilibrium in the year 1910 which in this exercise the year at which we want to start the actual simulation experiment. Such climate data are not yet available so we will cycle over the data we think best represent the conditions of our spinup. Set up the simulation to run over 4 forcing periods thus 40 years in total (note that for a global spinup it should be 34 cycles but we want to go faster in this exercise).

In config.card, set JobName=MyTestSpinup (or the name you want), DateEnd=1940-12-31, set SpaceName=TEST to deactivate pack post treatement if running at Jean-Zay or Irene. Set also TimeSeriesFrequency=5Y and SeasonalFrequency=NONE in config.card. Limit the region to a grid-cell to make this a fast test case. Set in PARAM/run.def:

```
LIMIT_WEST = 8.
LIMIT_EAST = 10.
LIMIT_NORTH = 48.
```

```
LIMIT_SOUTH = 46.
```

Because we do not specify the PFT's we want to simulate, ORCHIDEE will read the default PFT map for this pixel.

It is important to adjust the number of cores used (number of MPI and OMP) to the domain which is used. Here we run on 1 grid-cell and therefore set 1 MPI on the line for the executable to run in sequential mode. You should also deactivate XIOS server by setting IOS=("","") in the Executable section.

```
vi config.card => Change to the following:
                    JobName=MyTestSpinup
                    DateEnd=1940-12-31
                    OOL= (orchideedriver_${OptMode}, orchideedriver, 11MPI)
                    IOS= ("","")
                    TimeSeriesFrequency=5Y
../../../libIGCM/ins_job
```

Create the job using ins_job in libIGCM. Increase PeriodNb to at least 5 (so you only have to queue once) before launching the job. Launch the job

```
sbatch Job_MyTestSpinup    # adapt sumbission to your machine
```

and while waiting answer following questions:

1. Why should you deactivate XIOS in server mode in this example?
2. How do you calculate PeriodNb?
3. Which forcing file is used and where is it stored? Where is the shared repository IGCM? You can find the location in the main job.
4. Where is the output stored? How can you change the place for the ARCHIVE directory? Note that this is recommended only at obelix.
5. In orchideedriver.card you can use the variable **year** or **CyclicYear**. Which are the differences?
6. 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 simulation is finished and the time-series are completed (this may take 30 to 40 minutes), you can have a look at the evolution of the carbon pools. The spinup is mainly addressing vegetation and soil related issues (the soil water reaches its equilibrium in less than a decade). We will, therefore, have a closer look at the stomate files which can be found in the SBG folder.

```
cd .../IGCM_OUT/..../MyTestSpinup/SBG/Analyse/TS_YE
module load ferret
ferret
use MyTestSpinup_19010101_19401231_1Y_SOIL_ACTIVE_c.nc
use MyTestSpinup_19010101_19401231_1Y_SOIL_SLOW_c.nc
use MyTestSpinup_19010101_19401231_1Y_SOIL_PASSIVE_c.nc
show data
```

If something when wrong with the analysis by libIGCM you can easily concatenate the files yourself as following but for longer simulations we recommand that you use the TimeSeries_Checker.job (see documentation http://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/CheckDebug):

```
cd .../IGCM_OUT/..../JobName/SBG/Output/YE
module load nco
ncrcat -v SOIL_PASSIVE_c,SOIL_ACTIVE_c,SOIL_SLOW_c *_stomate_history.nc all.nc
module load ferret
ferret
use all.nc
show data
```

Note that nco is another software package. ncrcat is a command within the nco package. It will concatenate the unlimited-dimension of the netcdf file. This requires us to ensure that time is the unlimited-dimension when writing the netcdf files in ORCHIDEE. No worries, libIGCM takes care of that.

After typing show data, you should get an overview of the data and their dimensions that were loaded into ferret. In the ORCHIDEE output files the I and J dimension are reserved for the longitude and latitude. Given the simulation was set up for a single pixel, both dimensions should be 1. The K dimension is often (but not always) the number of PFTs. This should be 15. Do you know why?

Have a look in

```
COMP/orchideedriver.card
COMP/sechiba.card
```

Do you see how ORCHIDEE links its PFT map to its PFT-specific parameter file? ORCHIDEE can also be run without a PFT map, the user then needs to prescribe the PFTs (see section 6.2). This creates a large flexibility in terms of the number of PFTs and how they are defined but it is not at all useful for global simulations because with this set up the same PFTs will occur in each pixel.

Go back to ferret. There are two more dimensions: L specifies the number of time steps and D gives the number of data set. Each of these dimensions can be referred to in the ferret command.

```
set v ll; plot SOIL_PASSIVE_c[k=12]
```

The figure shows the temporal evolution of the passive soil carbon pool during the 40 year spinup for PFT 12. Which vegetation is PFT12 representing? Have a look in

```
PARAM/orchidee_pft.def_15pft.1ac
```

One of the most defining features of the graph are the sudden step changes. Make a more detailed plot by distinguishing the different components of the soil carbon pool.

```
set v ul; plot SOIL_ACTIVE_c[k=12,d=1]
set v ur; plot SOIL_SLOW_c[k=12,d=2]
set v ll; plot SOIL_PASSIVE_c[k=12,d=3]
```

The results you are looking at come from a semi-analytical spinup. Every 10 years the model calculates the soil and nitrogen carbon given the actual litter inputs. The following 9 years the soil nitrogen and carbon result in new litter inputs, ... Every step represents one calculation of the semi-analytical spinup. The magnitude of the changes should decrease over time.

We only looked in PFT 12 but can you think of a way to show which other vegetation is present in this pixel? Open one of the output files in ferret and list the content of the variable VEGET_MAX. Make a new output file and concatenate VEGET_MAX and plot its time series. What this this figure tells you about land cover changes?

Let's compare different PFTs

```
plot soil_passive_c[k=4], soil_passive_c[k=12]
```

Which PFTs are you comparing? What does this figure tells you about the time needed for different PFTs to reach an equilibrium? Is this what you expect?

Why is the semi-analytical spinup calculated every 10 years? The calculation of the semi-analytical spinup is 30% more expensive than running a year without the semi-analytical calculations. Using it too often reduces the computational benefits of using a semi-analytical method. Also, some of the vegetation (forest) take several decades if not centuries to reach maturity. We need to run the spin-up long enough to reach this developmental stage and its litter inputs. How can I change the frequency of the semi-analytical spinup?

Go to your config.card and change the CyclicEnd to 1905. Rerun the simulation for 10 years. Make a plot, what happened to the frequency? How does ORCHIDEE determine the spinup period?

5.2 Starting an experiment from a spinup

In the config.card you have set the parameter PeriodLength. This parameter determines when to write an output file but it also determines when to write a restart file. If PeriodLength = 1Y the output will consist of one output file per year. Note that in COMP/stomate.card and COMP/sechiba.card the write frequency determines how many time stamps there will be in this annual file (see section **??**). This PeriodLength=1Y instructs ORCHIDEE to write a restart file every 31st of December. You can find the restart files in:

```
.../IGCM_OUT/..../JobName/SBG/Restart/
.../IGCM_OUT/..../JobName/SRF/Restart/
```

If your simulation didn't finish yet, you can also take the restart files from simulation already done. See the paths here:

```
# At spirit:
/data/jgipsl/IGCM_OUT/OL2/TEST/test/MyTestSpinup
# At obelix:
/home/scratch01/jghattas/IGCM_OUT/OL2/TEST/test/MyTestSpinup
```

The restart files are not intended for human use. If you open them you will notice that much of the meta-data are missing. That's fine ORCHIDEE knows how to handle them.

As the name suggest a restart file contains all the information that is needed to restart the model. It thus contain all the state variables of the model. This is a key feature of the model and because it is so important it is checked weekly whether the restart files are up to their task. How this is done can be read here:

```
https://forge.ipsl.jussieu.fr/orchidee/wiki/Documentation/UserGuide/restartability
```

This implies that if you add a state variable to ORCHIDEE (typically a carbon, nitrogen, phosphorous, water, or energy pool but also a long-term temperature, productivity, season water stress, etc), this variable will need to be added to the restart files to ensure that ORCHIDEE is working properly. Although it is easy to show that the restart files are corrupt, it can be (very) time consuming to find the cause. Test the restartability of your developments regularly, it will also help with debugging (see section **??**).

In this exercise we will set-up a new simulation that starts from the end of the spinup rather than starting from scratch. Linking simulations is essential to set-up more complex simulation experiments.
Start by copying the OOL_SEC_STO_FG2 directory:

```
cd modipsl/config/ORCHIDEE_OL
cp -r OOL_SEC_STO_FG2nd MyTestExp1
cd MyTestExp1
```

Look into the config.card and note that the variables CyclicBegin and CyclicEnd are no longer defined. This is expected because in most experiments we want to use the actual climate rather than having to cycle of the same 10 years of climate forcing.

Also note that the FG2 configuration uses land cover change. This can be seen by

```
vi COMP/sechiba.card
```

The parameter VEGET_UPDATE is now set to 1Y. Compare this setting with the setting for the spinup. Moreover the PFT map has moved from the [InitialStateFiles] section to the [BoundaryFiles] because it now has to be read every year. Also, where the spinup made use of a single PFT map (indicated by the year that was hardcoded), the FG2 configuration uses a PFT map that matches the years set in the config.card. This can be seen by the use of the variable ${year} in the name of the PFT file.

You can only restart from a restart file that covers the exact same spatial domain. If you want to start from the restart file you just produced, several settings will have to be similar. Make sure that in the config.card you:

- give a unique jobname
- ensure that only 1 processor is used
- run the model for 5 years

- note that the path of the output files has changed

In the PARAM/run.def

- the exact same pixel needs to be selected

Above are the basic settings but we still need to tell libIGCM that ORCHIDEE needs to read the initial values of all its state variables from a restart file. Open config.card again

```
#D-- Restarts -
[Restarts]
#D- OverRule=y : All component will restart from the same simulation
OverRule=y
```

Set OverRule to y. This instructs the model to use all its restart files from the same spinup. More complex set-up are possible lower in the config.card.

```
#D- Date of restart file to be used
RestartDate=1940-12-31
```

Use the last year of the spinup. The restart file is written on December 31st (if PeriodLength=1Y). ORCHIDEE will read these values and use them starting on January 1st. It is nice if the years form a continues series but that is not necessary. The last year of the spinup could be 2240 and the first year of your simulation could be 1901. What really matters is a smooth environmental transition. If you used the 1901-1910 climate data in the spinup, there is no conceptual problem to start the experiment in 1901. If, however, the spinup cycled over 1901-1910 and you start the experiment in 1990, the response of the model could be due to the sudden change in the climate variables. Similar considerations apply to the other environmental conditions such as atmospheric CO_2, land cover change, nitrogen deposition, fertilisation, irrigation, land management, ...

```
#D- JobName for the restart file
RestartJobName=MyTestSpinup
```

List the job name which restarts you want to use.

```
#D- Path to directory where the RestartJobName directory is stored
#D- NB! ${ARCHIVE} depend on login. The full path can also be set
RestartPath=${ARCHIVE}/IGCM_OUT/OL2/TEST/spinup
```

Make sure to give the path where the spinup was saved. If you use the default paths, this differs between the ANALYTICAL_SPINUO_FG1 and OOL_SEC_STO_FG2nd templates. You can find all elements of the path (constructed as IGCM_OUT/TagName/SpaceName/ExperimentName/JobName) in the config.card.

Use ../../../libIGCM/ins_job to make a new job. remove the letter p from the queue, change it from medium to short, and set the parameter for PeriodNb to 5 (which is the length of this simulation)

This set-up already involves several changes so there is a fair change mistakes or typo's will occur. If the model crashes, first look into the Script file and search for ERROR. Carefully read the message. They are often sufficiently informative to solve the problem. If the error points towards the executable there should be a Debug folder. Go into the debug folder and open the *_out_orchidee file or the *_out_execution. Have a look at these files.

5.3 A simple experiment

An experiment requires a control and a treatment. We will use MyTestExp1 as the control. To reduce the chances to make a mistake we will copy MyTestExp1 to configure the treatmet.
Start by copying the MyTestExp1 directory:

```
cd modipsl/config/ORCHIDEE_OL
cp -r MyTestExp1 MyTestExp2
cd MyTestExp2
rm Job_MyTestExp1 run.card Script*
```

Open config.card and change the job name into MyTestExp2. The rest of the setting can be kept as we want the treatment to differ in only one aspect from the control. The fact that we can launch treatments that differ in only one aspect from the control is one of the strengths of simulation experiments compared to field experiments where such a high level of treatment control is extremely demanding.

As a treatment we will plant a different crop (PFT 12) with a higher nitrogen use efficiency.

```
vi PARAM/orchidee_pft.def_15pft.1ac
```

Add the following line

```
# Use a more efficient C3 crop ((mumol[CO2] s-1) (gN[leaf])-1)
NUE_OPT__12=70.00
```

Note that there must be two underscores between the parameter name and the PFT number. Parameter names have to be written in capitals. Make a job, change the queue, set PeriodNb, and launch the job.

Concatenate the files of MyTestExp1 and MyTestExp2 for TOTAL_M_c. Open ferret, load both concatenated files and plot the total biomass in a single plot

```
plot TOTAL_M_C[k=12,d=1], TOTAL_M_C[k=12,d=2]
```

Confirm that the crop with a higher nitrogen use efficiency grow faster than the crop with a lower nitrogen use efficiency.

6 Compile and run with debug options

By default the compilation is done in production mode which means that the compiler will use options for optimization so the model will run faster. For debugging purpose and to test your developments, it is good to switch to compilation in debug mode. You might then find errors more easier and your code will be more trustable. In the same modipsl, you can compile your code in both prod and debug mode. For each compilation the executables will be saved with the suffix _prod or _debug to make it easier to switch. While running with libIGCM, you need to specifiy in config.card which executables to use by setting OptMode=prod or OptMode=debug.

6.1 Compilation in debug mode

In this exercise you'll work on a revision on the model which contains an error. This revision of the model can run without any direct errors when compiled in default production mode. But when compiling with additional debug options, the model will crash due to errors in the code. The errors are always there even when using the default compilation but they are less visible and the model continues running instead of crashing.

Install a new modipsl, extract the ORCHIDEE_trunk configuration and update to use the revision 7847. Do as follow (see also the exercises on SVN):

```
mkdir TESTDEBUG; cd TESTDEBUG
svn co http://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl
cd modipsl/util/
./model ORCHIDEE_trunk
cd ../modeles/ORCHIDEE
svn info   # => check that you have the trunk version
svn update -r 7847
```

Use the option -debug for the compilation script this time. -debug activates compilation debug options whereas the default option -prod activates optimized compilation options.

```
cd ../../config/ORCHIDEE_OL
./compile_orchidee_ol.sh -debug
```

When the compilation finished, create a run directory with libIGCM as before (see 4.1). Always choose a new JobName. Before launching the job, you need to set OptMode=debug in config.card.

```
cp -r OOL_SEC_STO_FG2nd TestDebug
cd TestDebug
vi config.card       # Change at least JobName, OptMode, DateEnd
../../../libIGCM/ins_job
vi Job_TestDebug     # Check headers
```

Submit the job as before depending on the machine.

After the job finished running, confirm in the run.card that the model crashed. When the model crash, a Debug folder is created with output text files from the model. Check inside the Debug folder and try to find the error message. Check mainly in the files containing _out_ in the names. Try to find out why the model crashed. Have a look at the line number where the model crashed.

Note that the line number in the output text file doesn't correspond exactly to the line number in the source fortran file. Every time ORCHIDEE is compiled the fortran files are preprocessed and stored in

```
../modeles/ORCHIDEE/build/ppsrc/...
```

When you'll correct the model, do not change in the build folder but make sure to change the files in

```
../modeles/ORCHIDEE/src_xxx/xxx.f90
```

If you change in build folder, your changes will not take effect when compiling the model again.

Conclusion
Use the debug mode to ensure that the code meets all the requirements that can be checked by the compiler. It well greatly help you to improve the quality of your code. If you have access on more than one server, compiling the same code on different servers may also help you to find bugs.

Sadly many errors cannot be found by the compiler and if you will need some detective work to find and fix those problems. Run the model until it crashes. If it crashes in year 1 when running on a single pixel (thus early in your code developments) you can use the set-up that crashed for debugging but it is probably a good idea to increase the write frequency of the output and to reduce the number of PFTs (see further below in this section).

If a single-pixel run crashes after many years of simulations you can make use of the restart files to set up a configuration that crashes in single-year simulation. If a global simulation crashes you will have to find the pixel where the model crashes (this can be found in the Debug/MyJob_out_orchidee file) and set-up a single pixel run. If a global simulation crashes isolate the pixel and run until the model crashes again which now will be faster for the case you pick out a pixel which crashes. The challenge is to set-up a test case where the model crashes quickly, consistently, and always for the same reason. The latter two requirements can only be achieved if the model is restartable (see section 5.2). The vast majority of the bugs can be reproduced on a single pixel but if the bug depends on the interpolation of one of the boundary files or initial state files (see MyJob/COMP/xxx.card) it might be necessary to include the neighboring pixels in the test area as well. Hence, the text case will be 3x3 pixels.

More write statements

Debugging often makes use of WRITE statements to monitor what happens with a specific variable in the code. If you get an ipsl ERROR message in the Debug/MyJob_out_orchidee file it means that another developer has put it in the code. Read the message carefully and search the fortran file where it comes from. Many of these messages are (very) informative. This also implies that another developer predicted this problem or experienced something similar. There is a fair chance that person has already put WRITE statements in the code. The user can ask for each subroutine (especially stomate subroutines) to write additional output to the Debug/MyJob_out_prchidee file.

```
vi PARAM/orchidee.def
PRINTLEV=3             # Write more information in all modules
PRINTLEV_stomate_lpj=4 # Write more information in stomate_lpj module
```

Similar statements can be used for most modules. The label used after the PRINTLEV should be the name of the submodule. To be sure open the fortran file and search for get_printlev. One of the arguments passed in this function is the label that will be searched for in the orchidee.def file.

Note that if you use this option for global runs the Debug/MyJob_out_orchidee quickly becomes excessively large (Gb) which will slow down your simulation a lot and may fill up the disks as well. To make good use of this function you need a single-pixel, single-year test case. Many of the write statements are coded such that they only write the output for a single pixel and/or a single PFT. This single pixel option comes in handy if you have to use a 3x3 test case. The single PFT options is almost always handy.

```
vi PARAM/orchidee.def
  # Change to have the following
  TEST_GRID=1
  TEST_PFT=12
```

Rerun the model and have a look in Debug/MyJob_out_prchidee. You should now get output for PFT 12. If your test case only has a single pixel, TEST_GRID should be 1. If not, the model will crash.

6.2 Run without PFT map

Sometimes you may want to run the model for a PFT that is not present in the PFT map or you may have some hints that the crash comes from the PFT map. In that case it is a good idea to run ORCHIDEE without a PFT map. Setting the parameter IMPOSE_VEG=y all pixels will have the same fractions of vegetation. The fractions are given by the parameter SECHIBA_VEGMAX which is a vector. The sum of the vector must be equal to 1.0. If you set all PFTs to zero except for one, this will be the only PFT in your simulation. This can largely facilitate reading and interpreting Debug/MyJob_out_prchidee. The option IMPOSE_VEG=y is not compatible with land cover change so you must have VEGET_UPDATE=0Y. Do the following:

```
vi PARAM/orchidee.def
  # Change to have
  IMPOSE_VEG=y

vi PARAM/orchidee_pft.def_15pft.1ac
  # Change to set SECHIBA_VEGMAX as you want
  SECHIBA_VEGMAX__01=0.0
  SECHIBA_VEGMAX__02=0.0
  SECHIBA_VEGMAX__03=0.0
  SECHIBA_VEGMAX__04=0.0
  SECHIBA_VEGMAX__05=0.0
  SECHIBA_VEGMAX__06=0.0
  SECHIBA_VEGMAX__07=0.0
  SECHIBA_VEGMAX__08=0.0
  SECHIBA_VEGMAX__09=0.0
  SECHIBA_VEGMAX__10=0.0
  SECHIBA_VEGMAX__11=0.0
  SECHIBA_VEGMAX__12=1.0
  SECHIBA_VEGMAX__13=0.0
  SECHIBA_VEGMAX__14=0.0
  SECHIBA_VEGMAX__15=0.0

vi COMP/sechiba.card
  # Change to have
  VEGET_UPDATE=0Y
```

6.3 Daily output

There are often visual hints for a bug in one of the many output variables ORCHIDEE can produce. If the PeriodLength in the config.card is set to 1Y but the model crashes before the end of the year you will have no history files. One way around this issue is to set PeriodLength to 1D. This requires that you also set the write frequency in the COMP/xxx.card files to 1d. If you run such a test, the model will write a history file for every completed day.

If you didn't already correct the bug in the revision 7847, do it now. Compile again in debug mode. Do the following:

```
cd modipsl/modeles/ORCHIDEE
vi src_stomate/stomate_lpj.f90
  # Change on line 2707-2708 : index j into ivm

cd ../../config/ORCHIDEE_OL
./compile_orchidee_ol.sh -debug
```

Now set to use an annual run period (to have an annual output file) but with a daily write frequency. You can use the same experiment as before or create a new one. Do following:

```
cd TestDebug
vi config.card
  # Change to : PeriodLength=1Y

vi COMP/stomate.card
  # Change to : (the other output files can be set to NONE)
  output_level_stomate_history = 10
  output_freq_stomate_history = 1d
```

If you use the same experiment as before, remove the output from the previous simulation and relaunch :

```
../../../libIGCM/purge_simulation.job
submit the main job
```

Make a plot of two very powerful output variables of the stomate modules. Note that LAI_MEAN_GS is divided by 4 to make both variables scale between 0 and 1. Do the following :

```
module load ferret
ferret
use "../IGCM_OUT/..TestDebug/SBG/Output/MO/TestDebug_19010101_19011231_1M_stomate_history.nc"
plot PLANT_STATUS[k=12], LAI_MEAN_GS[k=12]/4
```

Search for 'Indices for phenology' in

```
vi ../modeles/ORCHIDEE/src_parameters/constantes_var.f90
```

That line and the following lines show the meaning of the values shown in the variable PLANT_STATUS. Do you see the logic of the variable?

7 Add a new output variable in ORCHIDEE

Use one of your previous installtions of modipsl. Create a diagnostic output variable for the variable resp_hetero_litter calculated in stomate_litter.f90 using XIOS. To do this, add following in modipsl/modeles/ORCHIDEE/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
./compile_orchidee_ol.sh
```

Create a new run directory as before (see 4.1). The nex variable has dimension DIMENSION(npts,nvm) in the subroutine. The corresponding grid to be used in the xml files is grid_ref="grid_nvm" in field_def_orchidee.xml and grid_ref="grid_nvm_out" in file_def_orchidee.xml. Add in modeles/ORCHIDEE/src_xml/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 modeles/ORCHIDEE/src_xml/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 the main job and use ferret to visualize the new output variable.

8 Read parameters 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 ORCHIDEE 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, orchidee.def and orchidee_pft.def are read from the model using a call to getin_p which is an interface taking different types of variables. See here some example of parameters:

- **LIMIT_EAST, LIMIT_WEST, LIMIT_NORTH** and **LIMIT_SOUTH** are borders (in degrees) for the horizontal domain to be modelized. The default values correspond to the domain of the forcing file. 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
```

- **RIVER_ROUTING** parameter activates the river routing, default value is TRUE.
- **NVM** number of PFTs used in the model. Default value is 13. This number must correspond to the number of PFTs in the vegetation map (PFTmap.nc). For example, the default set up for ORCHIDEE_2_0 which is used for the CMIP6 simulations uses NVM=15.
- **VEGET_UPDATE** frequency for updating the vegetation map. By default, VEGET_UPDATE=0Y which means that the vegetation map will not be updated. Using VEGET_UPDATE=1Y, the map will be updated each 1st of January. The PFTmap.nc for the current year must be available.

Answer following questions:

- Search in the model to find where the default value for the parameter **RIVER_ROUTING** is set. In which fortran file is the parameter read?
- Open ORCHIDEE/src_sechiba/hydrol.f90 and see how the variable **froz_frac_corr** is modified. Which is the default value and how can you change it without recompiling?
- Which parameter should be used to change default value to prescribe the atmospheric **CO2 concentration** and which is the default value? Does the default value set in orchidee.default correspond to the value set in the code?

8.1 Add a new parameter to be read during run-time

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 by adding call to getin_p in thermosoil_initialize. The default values should be the same as before. Add also a write command 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. For this exercise you can choose to run using a simple run directory as in the beginning of the exercises or using a libIGCM experiment such as OOL_SEC_STO_FG2nd.

9 Exercises to learn SVN

This exercise aims to learn the basic use of svn also called subversion. Different versions of ORCHIDEE are prepared with local modifications that you'll analyse. Go the the folder corresponding to your machine your're working on to do this exercies. Type the commands listed below and answer questions.

```
# At obelix / LSCE :
cd /home/orchideeshare/igcmg/TRAINING/ORCHIDEE_2023/modipsl/modeles
# At spirit (x) / IPSL MESO centre :
cd /projsu/igcmg/TRAINING/ORCHIDEE_2023/modipsl/modeles
# At jean-zay / IDRIS :
cd /gpfswork/rech/psl/commun/TRAINING/ORCHIDEE_2023/modipsl/modeles
```

Note that if you do not use the same version of svn as the one used when extracting the sources you might have an error message like this:

```
 > svn stat
svn: The path '.' appears to be part of a Subversion 1.7 or greater
working copy.  Please upgrade your Subversion client to use this
working copy.
```

For example at irene you need version 1.9.7 of subversion. If you have this error at irene, change module used as follow:

```
module unload subversion
module load subversion/1.9.7
```

Determine version and revision

1. Which version and which revision of ORCHIDEE is installed in directory ORCHIDEE.2?

```
cd ORCHIDEE.2
svn info
```

2. 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).

```
svn stat
svn diff
```

3. Are there any modifications done on the server more recent than your local version? What is the difference when you add -u? Use svn help to know more about the commands svn, for example *svn help stat*.

```
svn -u stat
```

4. Answer the same question for the version ORCHIDEE.3: Which version and which revision is installed here? Are there any local modifications? Modifications done later on the server?

```
cd ../ORCHIDEE.3
```

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

5. Now copy the version ORCHIDEE.3 into your workdir, besides the TESTOFFLINE folder. Is this version modified? Is it up to date or has it been changed on the server?

```
cp -r ORCHIDEE.3 /your_work_dir/myORCH.3
cd /your_work_dir/myORCH.3
```

6. You will now clean this version before updating it. Remove modifications which are not needed by using revert and check which files have been updated on the server. In this exercises, do not revert stomate.f90 and stomate_max.f90 because there modified versions will be used in next exercises.

```
svn stat; svn diff
svn revert file_where_modifications_are_not_needed_to_be_saved
svn -u stat
```

7. You will now update the full ORCHIDEE to the latest revision on the current version (path on svn). Always do this command from the base directory 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 conflict is detected
svn stat
```

8. 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 by using svn resolved.

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

9. What happend to the file stomate_vmax.f90? Where there any conflicts detected? Open the file and check the lines with the local modifications in your working version. Are the modifications still accurate?

A Appendix: More details

A.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.
- 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 begining of the file.