

# Getting started with the IPSL tools: modipsl and libIGCM

## Exercises for training course

*Revised for January 2023 training sessions.*

A. Caubel, P. Cadule, A. Cozic, C. Ethé, L. Fairhead, L. Falletti, J. Ghattas, N. Lebas, T. Lurton, S. Nguyen, R. Pennel, R. Person

---

### !!! Please read this first introduction carefully !!!

The aim of this document is to give you all the information on how to install, compile and launch simulations with reference configurations using *modipsl* and *libIGCM* environment.

**During the exercises, we show you step by step how to handle these tools and simulations but you will have to search in the IGCMG documentation for all of the details: [http://forge.ipsl.jussieu.fr/igcmg\\_doc](http://forge.ipsl.jussieu.fr/igcmg_doc). It's all part of the training!**

The current document contains an introduction section (0.) followed by 12 sections with exercises (see the table of contents thereafter). Depending on your knowledge of modipsl and libIGCM, we advise you to use this document as follows:

- **For beginners** (if you never used the tools or just a little), first you have to focus on **sections 0, 1, 2 and 3** which detail how to *install, compile and launch a basic simulation*. Note that subsection 3.7 is only useful for LMDZ users (LMDZ and LMDZOR).  
If you have time, you can continue with **sections 4 to 9. (intermediate level)**  
If you finish all of them, you can choose some other exercises from section 10 to 13, depending on your future use of the tools. (**specialised level**)
- **For more advanced users**, we advise to start with **sections 2 and 3** as you will need the *basic simulation* for other sections. But you should not spend too much time on these two sections.  
Then continue with **sections 4 to 9** to learn about *debugging, post-processing and monitoring*.  
If you finish all of them, you can choose other exercises from **section 10 to 13**, depending on your future use of the tools.
- Note that the exercise using **NEMO** configuration (**section 10**) is proposed as a complement.

- Note that **section 11** about **Ensemble** is only for users who know what an Ensemble is. If not, it probably means that you won't need to do ensemble runs.

All exercises can be done at **Jean-Zay/IDRIS** or **Irene/TGCC** and most of them at **Spirit/spiritx/IPSL** and **obelix/LSCE**.

For **Spirit/IPSL** and **obelix/LSCE**, you can only do the following sections: **0 -> 3.3 ; 4 ; 9**

Exercises proposed in this training session use LMDZOR\_v6 (LMDZ + ORCHIDEE) and ORCHIDEE\_4\_1 (ORCHIDEE offline) configurations. But everything you learn will be usable with all model configurations (IPSLCM6, LMDZORINCA, etc...).

You will find several questions in all this document. Answers are in a separate file that you can find in the IGCMG online documentation.

---

# Table of content

<b>0. Introduction</b>	<b>5</b>
0.0 For TGCC users	5
0.1 How to correctly install your environment?	5
0.2 Essentials notes on today's training	6
0.3 Subscribe to the platform-users mailing list	7
<b>1. Check your quota</b>	<b>8</b>
1.1 For IDRIS	8
1.2 For TGCC	8
1.3 For IPSL/Spirit	9
1.4 For LSCE/obelix	9
<b>2. Installing and compiling</b>	<b>10</b>
2.0 Install modipsl	10
2.1 Extract the LMDZOR_v6 configuration	11
2.2 Compile with the resolution 144x142x79	14
<b>3. Basic simulations</b>	<b>17</b>
3.1 Create the first experiment directory	17
3.2 Define and launch your first simulation of 1 day	19
3.3 How to clean up and relaunch (if needed)	26
3.4 Continue the simulation 4 more days	27
3.5 Create another simulation with pack	31
3.6 Use different forcing files	32
3.7 CREATE_clim and CREATE_ampi: Experiments to create initial state files and boundary conditions for LMDZ	33
3.8 Summary on how to extract, compile and launch a simulation	35
<b>4. Debug</b>	<b>37</b>
4.0 How can you analyse the Job Output: Script_Output?	37
4.1 Debug: setup error	37
4.2 Debug: error during the simulation	38
4.3 Compilation in debug mode	39
4.4 Use of RUN_DIR directory to debug without libIGCM infrastructure	41
<b>5. Create Time Series</b>	<b>43</b>
5.1 Launch 2 years with default Time Series	43
5.2 Add variables to Time Series and relaunch with the TimeSeriesChecker.job	45
<b>6. Monitoring and Inter-monitoring</b>	<b>47</b>
6.1 Monitoring	47
6.2 Inter-monitoring	47
6.2.1 from web interface tool "Inter-monitoring application"	47
6.2.2 from web interface tool "simu finder application"	48

<b>7. How to REDO part of a simulation</b>	<b>49</b>
7.1 Launch a 6 days simulation of LMDZOR experiment	49
7.2 Remove daily output file for ATM component (SBG on Spirit) of the days 3 and 4 (i.e. 1980-01-03/04)	52
7.3 Apply the method to redo day 3 and 4 of the simulation (to recover missing output file)	53
<b>8. Modify output using XIOS</b>	<b>56</b>
8.1 Create a new output file for ORCHIDEE	56
8.2 Enable a new output file in LMDZ	58
8.3 XIOS in other models	60
8.4 XIOS and CMIP workflow	60
<b>9. Output files manipulations</b>	<b>61</b>
9.0 Protocol and environment	61
9.0.1 Protocol	61
9.0.1 Environment	62
9.1 Network Common Data Form (NetCDF) format	63
9.2 NetCDF Operator (NCO)	64
9.3 Climate Data Operators (CDO)	65
9.4 NetCDF Visual browser (NCView)	66
9.5 Ferret	66
9.6 NCAR Command Language (NCL)	67
9.7 Python	69
9.7.1 NetCDF4 / Numpy	69
9.7.2 XArray	71
<b>10. Install and run NEMO-PISCES</b>	<b>73</b>
10.1 Run a 1 month online experiment of NEMO-PISCES	73
10.2 Run a 1-year offline experiment of NEMO-PISCES	75
<b>11. Ensembles</b>	<b>79</b>
11.1 Install and configure ensemble	80
11.2 Configure the ensemble	80
<b>12. Coupled model</b>	<b>85</b>
<b>13. ICOLMDZOR configuration</b>	<b>87</b>

# BEGINNER

## 0. Introduction

### 0.0 For TGCC users

Send an email to Anne Cozic or Arnaud Caubel to ask for adding your login to igcmg group. It's mandatory to access input model data, and environment files.

### 0.1 How to correctly install your environment?

Before working with modips/libIGCM on IDRIS or TGCC, you need to install your environment. For this, you will find all the necessary information in the subsection "How to install your environment" specific to the machine you are using:

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/ComputingCenters](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters)

Besides, take some time to read the information about the machine you are using:

- For IDRIS file system  
[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/ComputingCenters/IDRIS#Thingstoknowaboutfilesystems](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters/IDRIS#Thingstoknowaboutfilesystems)
- For TGCC file system  
[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/ComputingCenters/TGCC#Aboutfilesystems](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters/TGCC#Aboutfilesystems)
- Working on Spirit/IPSL  
[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/ComputingCenters/ESPRImesocenter](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters/ESPRImesocenter)
- Working on Obelix/LSCE  
[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/ComputingCenters/LSCE](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters/LSCE)

#### **Note on environment variables:**

**Warning** : In this document, we mainly use the disk spaces' environment variables for IDRIS (`$WORK...`). Replace it by a work or scratch director at your machine. For example by `$CCCWORKDIR` if you are on **Irene**, by `/data/$USER/` if you are on **spirit**, by `/homedata/$USER/` if you are on **spiritx** or by `/home/scratch01/$USER/` if you are on **obelix**.

For other environment variables, please refer to the documentation of each machine (see above).

## 0.2 Essentials notes on today's training

All exercises can be done at **Irene/TGCC** or at **Jean-Zay/IDRIS** and most of them at **obelix/LSCE**, read specifications in the text.

**Note that for this training session we will work only on IDRIS temporary accounts.** There are a few specific commands that you will not need when you will work on other machines and they are marked as **“For January 2023 training course at IDRIS”**.

### **For January 2023 training course at IDRIS: use training account**

During the training session, specific training accounts on Jean-Zay will be used. They have login `cforXXX` with password `****`. Connect first to the machine **ipcours** and then use your temporary login.

*If you need to switch between qwerty and azerty you can use the command **alt+shift**.*

To access Jean-Zay, open a terminal and type the command:

```
ssh -Y jean-zay4
```

For your first connection to Jean-Zay, you need to install the IPSL environment. Copy/past the following 3 lines:

```
balo=$WORK/../../../../rech/psl/commun/MachineEnvironment/jeanzay/bash_login
```

```
cp $balo $HOME/.bash_login
```

```
source $HOME/.bash_login # don't forget the "." before the file
```

## 0.3 Subscribe to the platform-users mailing list


Before working with modipsl/libIGCM and IPSL models, you should subscribe to the **platform-users mailing list**. Do this by following the link:

<https://listes.ipsl.fr/sympa/info/platform-users>

The purpose of this list is to share information about IPSL tools (such as modipsl and libIGCM) and computing centres, and to help users with their needs and problems. *Anyone can ask any question, and we encourage everyone to answer the questions if you know the answer.*

To write to the mailing list: [platform-users@listes.ipsl.fr](mailto:platform-users@listes.ipsl.fr)

You can consult the archives directly in <https://listes.ipsl.fr/sympa/info/platform-users>:

 List Options
Owners: anne cozic, arnaud caubel, josefine ghattas, olivier marti
Moderators: (same as owners)
Contact owners
List Home
Subscribe
Unsubscribe
Archive
Post
Shared documents

## platform-users@listes.ipsl.fr

Subject: Liste utilisateurs groupe plate-forme modélisation climat IPSL

List archive 

Search

Advanced search

2017 [01](#) [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#) [10](#) [11](#) [12](#)

2018 [01](#) [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#) [10](#) [11](#) [12](#)

2019 [01](#) [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#) [10](#) [11](#) [12](#)

2020 [01](#) [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#) [10](#) [11](#) [12](#)

2021 [01](#) [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#) [10](#) [11](#) [12](#)

2022 [01](#) [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#) [10](#) [11](#) [12](#)

2022/02 2 mails

[Chronological](#) [Thread](#) [<<](#) [<](#) page 1 / 1 [>](#) [>>](#)

[\[platform-users\] irène](#). Cozic Anne. 02/07/2022

[\[platform-users\] Scratchdir Irene TGCC](#). Cozic Anne. 02/15/2022

## BEGINNER

# 1. Check your quota

In all computing centers, space and number of inodes (files and directories) are limited.

Do the exercises below on the computing center where you have a login.

Remember that you will find the questions' answers in the presentation of the first day and in the [IGCMG doc](#).

## 1.1 For IDRIS

→ Use the command `idrquota -m` to check the HOME quota, `idrquota -w` for WORK quota and `idrquota -s` for STORE quota. Analyse what you see on the screen.

### Question 1a

- Is the quota individual? What happens to the other users if you exceed the quota?
- What kind of quotas do you have?
- What is the meaning of "non\_files"?
- Which type of files do you store in your HOME? your WORK? and your STORE?

To make cleaning up easier, you can use the "find" command to list all small files at STORE:

```
cd $STORE
find . -type f -size -32M
```

## 1.2 For TGCC

→ Use the command `ccc_quota` to show your current quota and the limits on all file systems. Analyse what you see on the screen.

### Question 1b

- Is the quota individual? What happens to the other users if you exceed the quota?
- What kind of quotas do you have?
- What is your global score?
- What is the meaning of "non\_files"?
- Which type of files do you store in your HOME, WORKDIR and STOREDIR?
- What is the size of the files that you are supposed to store in the STOREDIR?



To make cleaning up easier, you can use the “find” command to list all small files at STOREDIR:

```
cd $CCCSTOREDIR
find . -type f -size -32M
```

### 1.3 For IPSL/Spirit

On Spirit, you have individual quotas for your home and for the data space.

→ Use the `quota` command to check the quota

### 1.4 For LSCE/obelix

On the LSCE cluster, there is an individual quota only in your home, in `/home/users/login`.

→ Use the `quota` command to check the quota in your home. On the other disks, there is no quota control but they can saturate. Use `df -h` to see the occupation of the disks.

Note that the default base directory for the archive of output files is defined in libIGCM to `/home/scratch01/yourlogin` for obelix. This scratch directory might be purged and therefore you have to save your important simulations on another disk. You can change the archive by setting the variable “ARCHIVE” directly in the `config.card` or change it in `modips1/libIGCM/libIGCM_sys_obelix.ksh`.

## BEGINNER

## 2. Installing and compiling

In this section, you will learn how to install the tools and compile the configuration. All the necessary commands are listed in the text.

→ Start by creating a new directory in your `$WORK`.

**Warning** : in all commands, replace `$WORK` by `$CCCWORKDIR` if you are on **Irene**, and `/data/$USER/` if you are on **Spirit**.

```
mkdir $WORK/MYFIRSTTEST ; cd $WORK/MYFIRSTTEST
```

### 2.0 Install modipsl

→ Download modipsl:

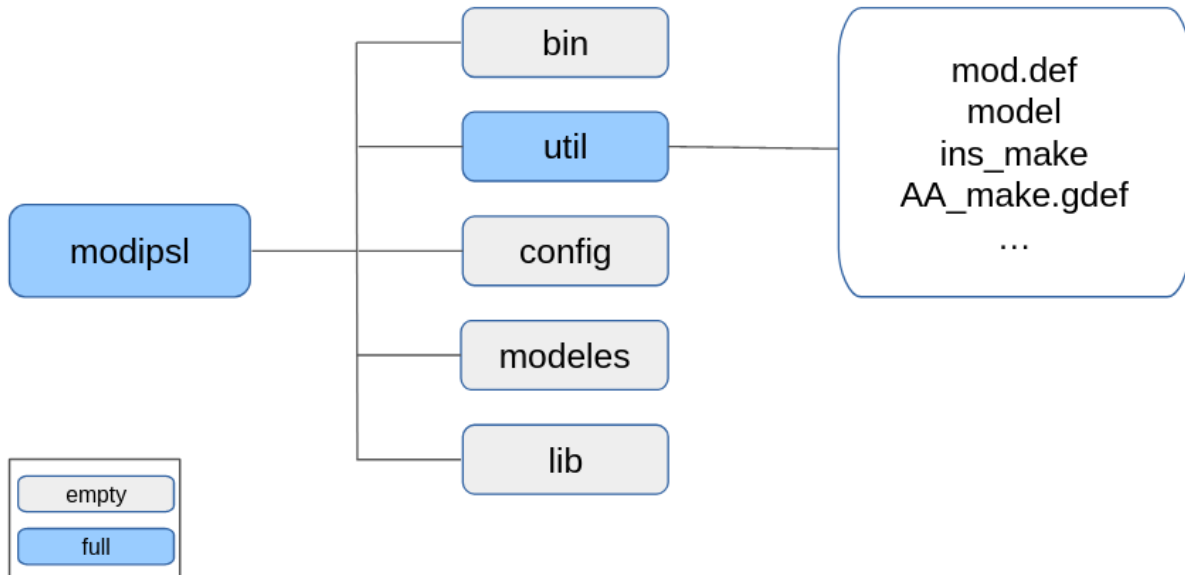
```
svn co --username icmc_users  
https://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl
```

The passwords must be asked to a Platform group member.

Other method: as you have installed the IPSL environment, you can use the `svn_ano` command instead of the previous one (`svn_ano` is an alias for the command, it's define in IPSL environment) on TGCC and IDRIS to download modipsl:

```
mkdir $WORK/MYTESTALIAS ; cd $WORK/MYTESTALIAS ; svn_ano
```

- Explore the `modipsl/` directory. You can see that some directories are empty. To download one model's configuration and all associated scripts you will use a script stored in the `modipsl/util/` directory.
- Compare your `modipsl/` tree and the following diagram.



You can find the description of all these directories here:

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Install#Themodipsldirectories](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Install#Themodipsldirectories)

Scripts stored in the `util/` directory can be used to:

- Review the models configurations to download (`mod.def`)
- Download the chosen configuration (`model`)
- Create a makefile adapted to this configuration and the supercomputer on which you work (`ins_make`, `AA_make.gdef`) if you are extracting a configuration older than v6.2 (if you are working with v6.2 or newer configuration versionn we no longer create anymore a makefile)

→ Explore the `util/` directory:

```
cd $WORK/MYFIRSTTEST/modipsl/util
ls
```

## 2.1 Extract the LMDZOR\_v6 configuration

**Note for Spirit users:**

Because not all the configurations can run efficiently on Spirit, the experiment used for the purpose of this training will be different from that on other machines: it shall be an ORCHIDEE offline run, whose creation procedure is slightly different. Please refer to the notes indicated in the instructions. You can use LMDZOR\_v6.2 and more recent configuration as well but not using more than 32 cores in total.

*Description:* The `model` script is used to download a specific predefined configuration with the model source codes and the necessary tools. The script uses the `mod.def` file which contains the specifications for each predefined configuration.

→ Use the `./model -h` command to see all existing configurations and `./model -h config_name` (in this exercise `config_name` will be “LMDZOR\_v6.2.2”) to get information about a specific configuration.

The same information can be found by reading the `mod.def` file. You can find information on how to read the `mod.def` file on this page:

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Install#Syntaxinmod.def](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Install#Syntaxinmod.def)

**Question 2a:** Using the `./model -h` command, find out which version of LMDZ, ORCHIDEE and libGCM are currently defined in the **LMDZOR\_v6.2.2** configuration. Note the SVN revision number and SVN branch or tag name. Check that you can find the same information in the `mod.def` file.

Note on Subversion (SVN) - a version control software :

IPSL models are saved via svn, this allows to keep track of changes done over the time, backup and store all previous versions, centralise all existing developments done on each model.

Each modification on svn will match with a revision number and a save path (with prefix trunk, tag or branches). To display them, you should use the `svn info` command.

```
cd $WORK/MYFIRSTTEST/modipsl/util
./model -h
./model -h LMDZOR_v6.2.2
vi mod.def
```

→ Now download the LMDZOR\_v6.2.2 configuration (IDRIS and TGCC users) by using the `model` script.

Note: for the first extraction, passwords for IOIPSL, ORCHIDEE, LMDZ and libIGCM are needed.

When prompted for password:

press ENTER to erase the default login which is proposed, and then use specific login credentials given at the beginning of the day.

**For TGCC and IDRIS users (for Spirit users see below) :**

```
./model LMDZOR_v6.2.2
```

**For Spirit users:**

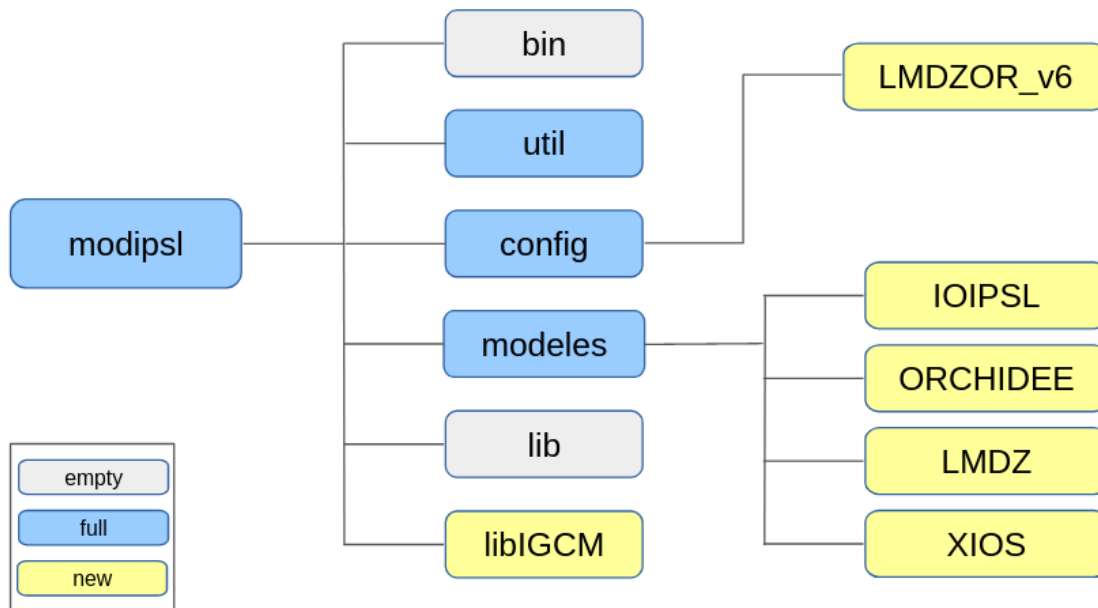
Download the ORCHIDEE\_4\_1 configuration by using the `model` script.

```
./model ORCHIDEE_4_1
```

→ Now explore the directories in `modipsl`.

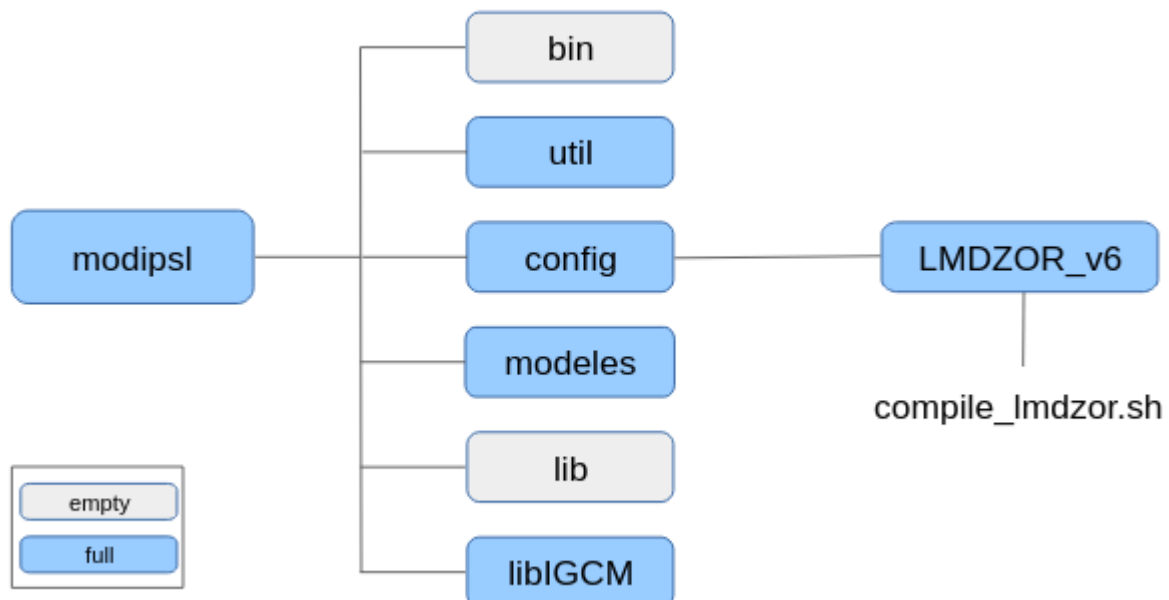
You can see in `modipsl/modeles` that you have one directory per model. You also have the `modipsl/config/LMDZOR_v6` directory (`modipsl/config/ORCHIDEE_OL` for Spirit) and the `modipsl/libIGCM` directory.

→ Type `svn info` in each model directory to get information about the extracted version and compare them with your answers to question 2.a.



## 2.2 Compile with the resolution 144x142x79

We use a specific script to launch the configuration compilation. This script is located in the config/LMDZOR\_v6 directory.



**Question 2b:** Open the compilation script and try to find all available options for the compilation. Find out which resolution is the default one, then launch the compilation for the resolution 144x142x79. You can use this page

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Compile#Scriptforconfigurations\\_v6.2andnewer](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Compile#Scriptforconfigurations_v6.2andnewer) to help you to understand the script syntax.

If you are working on **Jean-Zay/IDRIS**, don't forget to log in to the preprocessing front-end (otherwise the compilation of XIOS will not work):

```
ssh jean-zay-pp
```

We will use the LMDZOR default resolution 144x142x79 for the atmosphere. To compile, use the `compile_lmdzor.sh` compilation script with the option for the chosen resolution:

```
cd $WORK/MYFIRSTTEST/modipsl/config/LMDZOR_v6
./compile_lmdzor.sh -resol_atm 144x142x79
```

#### For Spirit users:

We will compile an ORCHIDEE offline configuration, using the `compile_orchidee_ol.sh` script.

```
cd /data/$USER/MYFIRSTTEST/modipsl/config/ORCHIDEE_OL
./compile_orchidee_ol.sh
```

#### For January 2023 training course at IDRIS:

Launch the compilation as explained above.

**In case that the compilation duration is too long**, you need to done the 2 following points :

1- copy the executable in your bin directory:

```
cp
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2023/LMDZOR_v6
/jean-zay/bin/* $WORK/MYFIRSTTEST/modipsl/bin/.
```

2- create the environment file (it will be used by the simulation to install the environment):

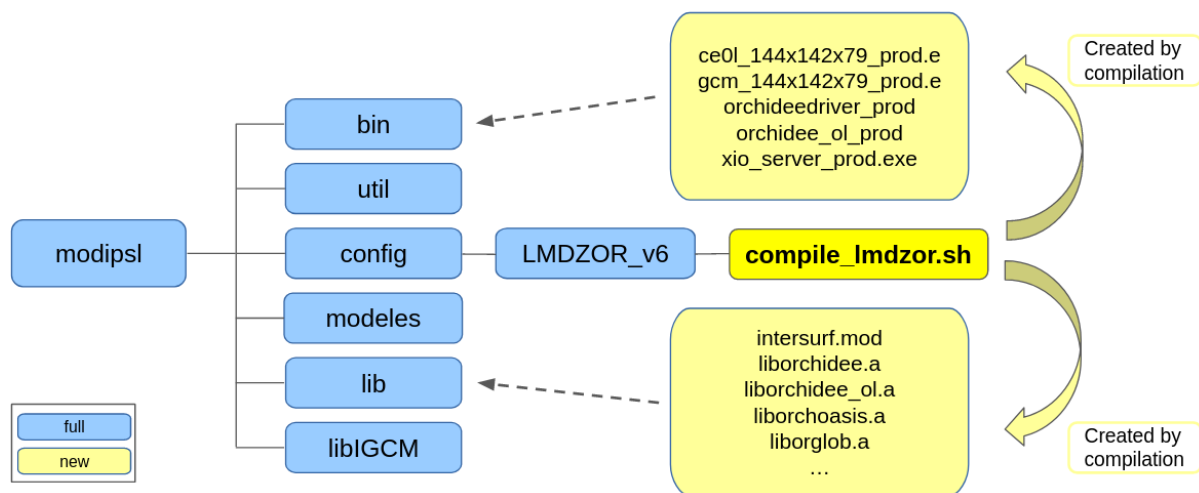
```
cd modipsl/config/LMDZOR_v6/ARCH
ln -s arch-X64_JEANZAY.env arch.env
```

## Comments on the compilation

The compilation creates executables which are necessary to launch simulations. Note that the executables are built for the specific configuration of models that you have downloaded (see [1.1 section](#) ).

When the compilation is over, you will find the executables in the `modipsl/bin` directory. The compilation takes from 30 to 60 minutes depending on the platform. After the first compilation, if you run a new one, only the modified files and the files depending on them will be compiled.

→ Don't forget to check that the executables are present in the `modipsl/bin` directory!



In the case of the `LMDZOR_v6` configuration, the `orchideedriver_prod` and `xios_server_prod.exe` executables are specific to the offline model. In the following exercises we will use the coupled configuration with `ce0l`, `gcm`, and `xios_server` executables.

**Question 2c:** What if you want to recompile the whole code? Open the compilation script and check the different script options.

If you are working on **Jean-Zay/IDRIS**, don't forget to log out from the preprocessing front-end

`exit`

## Specific installation of LMDZ on obelix/LSCE:

Read more about using LMDZ on obelix here:

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/ComputingCenters/LSCE](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters/LSCE)



## BEGINNER

### 3. Basic simulations

*For Spirit users: Orchidee\_ol does not work exactly like the other configurations (as described below). You can read this section to know how other configurations work, and then do what is written in the “spirit” box at the end of part 3.1.*

**In a configuration with the same executable, we can choose between several types of experiments.** All the available experiments are stored in the `config/your_config/EXPERIMENTS/` directory.

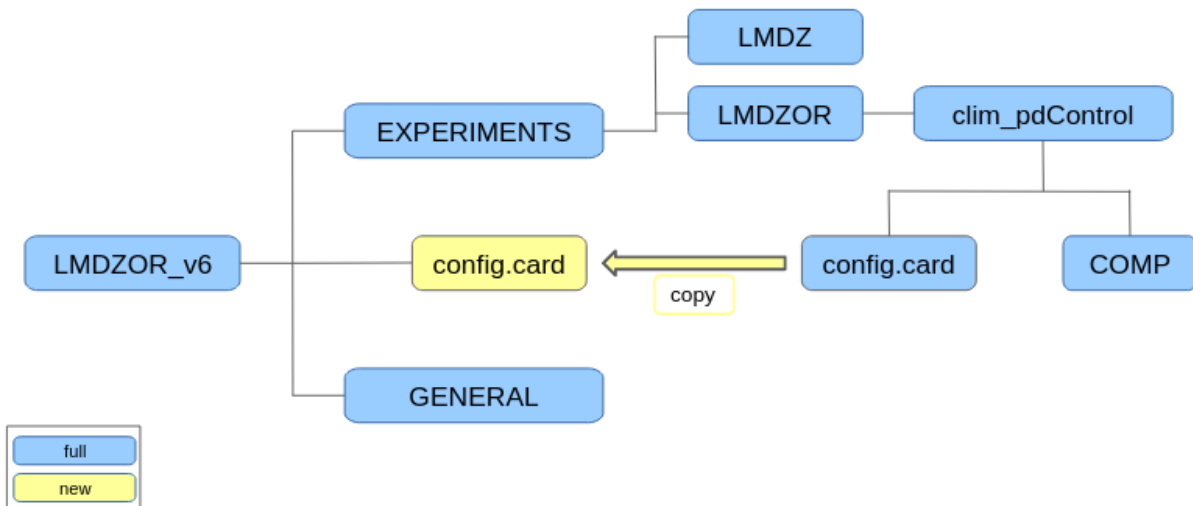
For instance, with the LMDZORINCAREPR configuration (lmdz + orchidee + inca + reprobis) you can launch a lmdz simulation, or lmdzor, or lmdzorinca, or lmdzrepr, or lmdzorincarepr and **all of them share the same executable.**

So, once you have chosen which model configuration you want to work with, you need to download it, compile it, and then you can choose which type of experiment you want to use.

#### 3.1 Create the first experiment directory

In the `EXPERIMENTS` directory you can find different predefined experiments which you can possibly run using the configuration you extracted. For the LMDZOR\_v6 case, you can choose between the LMDZOR and LMDZ types of experiments.

For this exercise we will create an experiment from `LMDZOR/clin_pdControl`. To do this, we will copy the `config.card` file located in `EXPERIMENTS/LMDZOR/clin_pdControl` into the `config/LMDZOR_v6/` directory.



The experiment directory will be created with information found by libIGCM in the `config.card` file. Before creating this directory, we need at least to indicate the name of the experiment. Depending on the machine you are logged in, you also have to change parallelization's information (see below).

The `modips1/libIGCM/ins_job` script will be used to create the experiment directory from the experiment name and other information in `config.card`.

→ Now follow the instructions:

```

cd $WORK/MYFIRSTTEST/modips1/config/LMDZOR_v6
cp EXPERIMENTS/LMDZOR/clim_pdControl/config.card .

vi config.card

    # Modify JobName=MyJobTest

    # Modify in [Executable] part for the parallelization (and NOT in
[List of Components] part)
[Executable]
ATM= (gcm_${ResolAtm}_${OptMode}.e, lmdz.x, 71MPI, 8OMP)
SRF= ("", "")
SBG= ("", "")
IOS= (xios_server_${OptMode}.exe, xios.x, 1MPI)

    # On JeanZay, in ATM: replace 8OMP by 5OMP
    # On obelix, in ATM: replace 71MPI by 7MPI ; and 8OMP by 1OMP
    # On Irene skl or amd, change nothing for parallelization

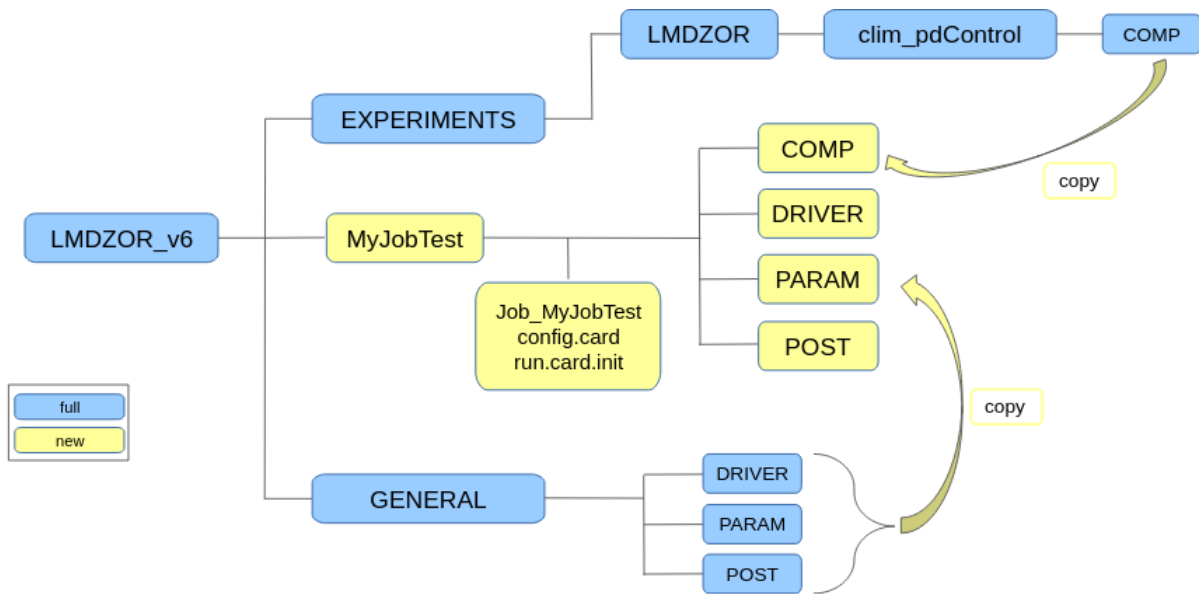
../../libIGCM/ins_job # At JeanZay and Irene, enter your project ID
                      # Answer default for other questions
                      # Press "enter" for default choices

cd MyJobTest

```



The submission directory has been created with the same name as the `JobName`. Explore this directory and compare its content to the following diagram.



### **For Spirit users:**

Since the experiment used on Spirit is an offline run of ORCHIDEE, the procedure for creating the experiment is a bit different: we do not copy a `config.card` file to the parent directory. Instead, you need to copy the prepared directories as an experiment directory. We will also customize our domain so that it is not too large. However, the `modipsl/ins_job` script is still used to create the job when needed.

```
cd /data/$USER/MYFIRSTTEST/modipsl/config/ORCHIDEE_OL
cp -r OOL_SEC_STO_FG2 MyJobTest
cd MyJobTest
vi config.card # => Change JobName=MyJobTest
                # SpaceName=TEST
                # Modify Executable part for the parallelization
OOL= (orchidee_ol_${OptMode}, orchidee_ol, 31MPI)
IOS= (xios_server_${OptMode}.exe, xios.x, 1MPI)
```

## 3.2 Define and launch your first simulation of 1 day

In this subsection, you will prepare and launch your first test simulation.

Generally, before any important experiment, it is good practice to check the correct behaviour of the workflow with a test simulation. In particular, we need to check that pre- and post-processing steps do not induce any errors and that the simulation meets our expectations.

### How to define a simulation?

To define a simulation, you need to know the answers of the following questions:

1. Which start and end dates for the simulation ?
2. Is your simulation a TEST, DEVT or PROD ? This choice defines where your simulation output will be stored  
[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Running#Theoutputfiles](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Running#Theoutputfiles)
3. Which calendar will you use?  
[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Setup#config.card](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Setup#config.card)
4. Which initial state files?
5. Which boundaries files?
6. Which output variables? With which frequency?
7. Which post-processing?

If the simulation is a TEST (as in this exercise), we will not answer the last question (number 7) because there is no post-processing for TEST simulations. For this training session, we will use default arguments in `config.card` for questions 3, 4, 5 and 6. The 7th question will be seen in a later exercise.

### Setup the `config.card` file

- You must be in the directory specially created for your simulation (`MyJobTest/`).
- Now set the `config.card` to run a short 1 day simulation. (`DateEnd` = last day of simulation):

```
DateBegin=1980-01-01  
DateEnd=1980-01-01
```

This is a first test simulation so keep `SpaceName=TEST`. This option will deactivate pack functions and no archiving will be done. Outputs will therefore be found on `$SCRATCH` (JeanZay).

```
JobName=MyJobTest  
#---- Short Name of Experiment  
ExperimentName=clim  
#---- DEVT TEST PROD
```

SpaceName=**TEST**  
LongName="LMDZOR configuration"  
TagName=LMDZOR  
#D- Choice of experiment in EXPERIMENTS directory  
ExpType=LMDZOR/clim\_pdControl

**Note for Spirit users:**

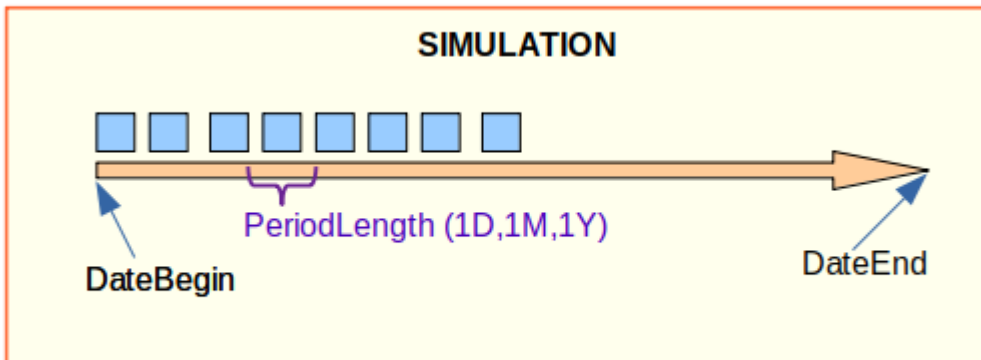
You should find LongName as empty, TagName=OL2, ExperimentName=test, and no ExpType field. All of these settings are normal.

For a 1 day simulation you will indicate PeriodLength=1D:

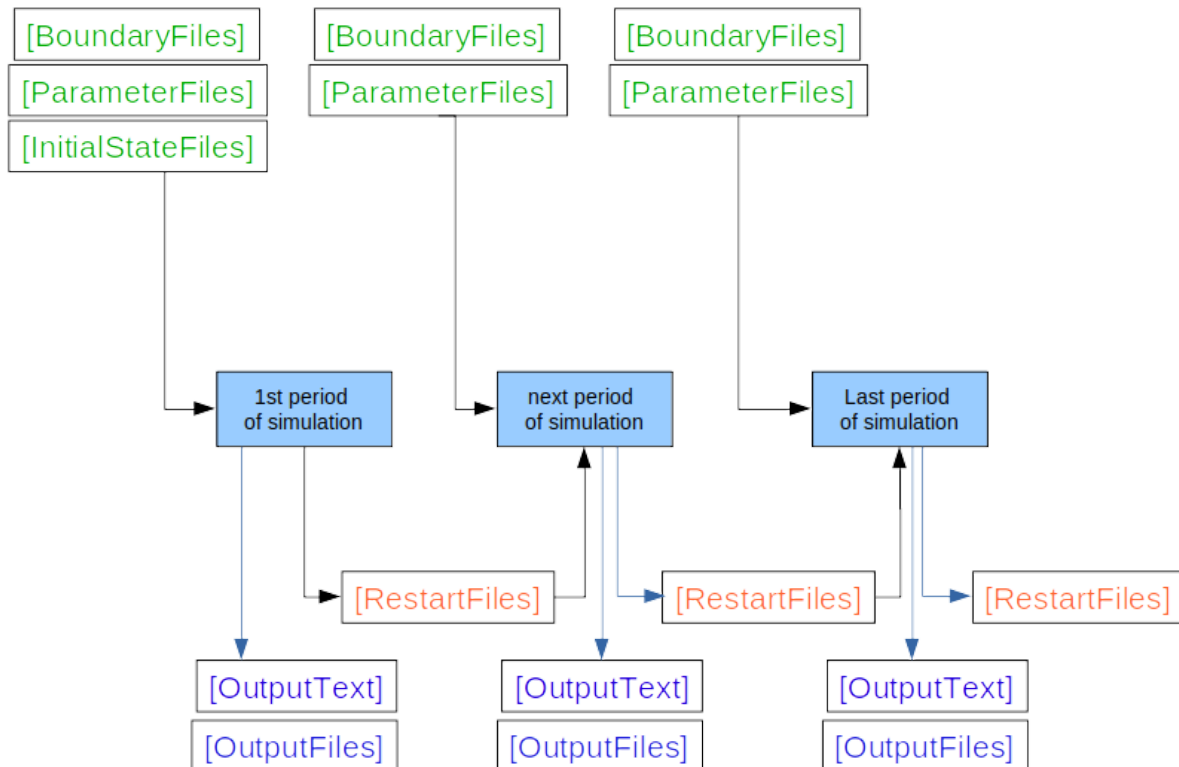
PeriodLength=**1D**

**What is a period?**

A simulation is a succession of **periods**.



At the end of each of them the simulation creates output files used for some of them as input files for the next period.



### Post-processing in `config.card`

→ We will deactivate all post-processing in `config.card` (you will see how to use them in sections 2.4 and 4.):

```
#D-- Post -
[Post]
#D- PackFrequency determines the frequency of pack submission
PackFrequency=NONE
#D- TimeSeriesFrequency determines the frequency of post-processing submission
#D- Set NONE to deactivate the creation of all time series
TimeSeriesFrequency=NONE
#D- SeasonalFrequency determines the length for each seasonal average
#D- Set NONE to deactivate the creation of all seasonal average
SeasonalFrequency=NONE
#D- Offset for seasonal average first start dates ; same unit as SeasonalFrequency
#D- Usefull if you do not want to consider the first X simulation's years
SeasonalFrequencyOffset=0
#D- If you want to produce compute PCMDI metrics from seasonal average
#D- Set TRUE or FALSE to activate/deactivate the metrics computation.
MetricsPCMDI=FALSE
```

### Definition of Output files in COMP/\*.card

Since we run a 1 day simulation, we want to activate daily outputs (and deactivate monthly outputs which are default outputs). To do that, it is needed to modify output specifications in component cards as follows :

- COMP/lmdz.card

```
output_level_histmth = NONE
output_level_histday = 3
```

- COMP/orchidee.card

```
output_freq_sechiba_history = 1d
output_freq_sechiba_history_4dim = 1d
```

- COMP/stomate.card

```
output_freq_stomate_history = 1d
output_freq_stomate_ipcc_history = 1d
```

### The main job Job\_MyJobTest

This file is the one used by the job scheduler to launch the simulation. It requires information in the header that is specific to the machine you are using.

- Now, you have to check the header in the Job\_MyJobTest main job then you can submit the job.

You can find a documentation on job headers syntax for Irene and Jean Zay here [https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Setup#Jobheaders](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Setup#Jobheaders).

To launch a [test](#) (on Jean Zay or Irene) you have to modify the CPU time and indicate that you will use the test queue.

- Header for Jean Zay:

```
#####
## JEANZAY IDRIS ##
#####
```

```

#SBATCH --job-name=MyJobTest # Job Name
#SBATCH --output=Script_Output_MyJobTest.000001 # standard output
#SBATCH --error=Script_Output_MyJobTest.000001 # error output
#SBATCH --nodes=9
#SBATCH --exclusive
#SBATCH --ntasks=72 # Number of MPI tasks
#SBATCH --hint=nomultithread # 1 processus MPI par par physical core (no
hyperthreading)

#SBATCH --time=00:30:00 # Wall clock limit (hours:minutes:seconds)
OR (other syntax)
#SBATCH --time=30 # Wall clock limit (minutes)

#SBATCH --account for@cpu
#SBATCH --qos=qos_cpu-dev # Queue test

```

For more information about `--qos` available on Jean-Zay, visit the [official webpage](#).

- **Header for Irene skl or rome:**

```

#####
## IRENE TGCC/CEA ##
#####
#MSUB -r MyJobTest # Job name
#MSUB -o Script_Output_MyJobTest.000001 # Standard output
#MSUB -e Script_Output_MyJobTest.000001 # Error output
#MSUB -eo
#MSUB -n 976 # Number of MPI tasks allocated
#MSUB -x # Node exclusivity
#MSUB -T 1800 # Wall clock limit (seconds)
#MSUB -Q test # Test queue (max: 1800 seconds)
#MSUB -A gen**** # Project allocation
#MSUB -q skylake (or rome) # Partition used
#MSUB -m store,work,scratch # Visible spaces

```

(for Irene, the wall clock limit for the test queue is 1800 seconds maximum, if you are not running on the test queue you can ask for 86400 seconds max).

- **Header for spirit and spiritx:**

This should come up as default:

```

#####
## MESO ESPRI IPSL ##
#####
#SBATCH --job-name=MyJobTest # Job Name

```



```
#SBATCH --output=Script_Output_MyJobTest.000001 # standard output
#SBATCH --error=Script_Output_MyJobTest.000001 # error output
#SBATCH --ntasks=32 # Number of MPI tasks
#SBATCH --hint=nomultithread # 1 processus MPI par par physical core (no
hyperthreading)
#SBATCH --time=30 # Wall clock limit (minutes)
```

### **Launch the job**

→ Now, use one of these commands (depending on your machine) to launch the job:  
*sbatch* (IDRIS, spirit/spiritx) / *ccc\_msub* (TGCC) / *qsub* (Obelix)

```
cd $WORK/MYFIRSTTEST/modipsl/config/LMDZOR_v6/MyJobTest/

JeanZay: sbatch Job_MyJobTest
```

### **The `run.card` file: to follow the status of your simulation**

To keep track of the status of your simulation a `run.card` file is created. Please read the pages [https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Running#Statusoftherunningsimulation](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Running#Statusoftherunningsimulation) and [https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Running#Endofthesimulation](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Running#Endofthesimulation) for more information.

→ Use the `run.card` file to check that your simulation was completed correctly.

You can also use the following commands to check the job queue and check if your simulation is waiting/is still running/has finished:

*squeue* (IDRIS, Spirit) / *ccc\_mpp* (TGCC) / *qstat* (Obelix)

To see only your jobs, you can add the option `-u $user`.

```
JeanZay : squeue -u $USER
```

### **How to delete a job?**

If you need it, you can use one of these commands to delete a job, depending on the machine you are using:

*scancel* (IDRIS, Spirit) / *ccc\_mdel* (TGCC) / *qdel* (Obelix)

followed by your job ID:

### On JeanZay

```
squeue -u $USER
>> JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
>> 389685 cpu_p1 LMDZOR02 rpsl592 R 11:05 15 r5i7n[9-23]
scancel 389685
```

### On Irene

```
ccc_mpp -u $user
>> USER ACCOUNT BATCHID NCPU QUEUE (...)
>> p24cozic aercmip6 3351314 624 skylake (...)
ccc_mdel 3351314
```

### On Spirit

```
squeue -u $USER
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
93158 zen4 MyJobTes snguyen R 0:06 1 spirit64-03
Scancel 93158
```

→ Explore the **Script\_Output\_\*.0001** and **run.card** files in the submit directory.

If your simulation encountered a problem the first thing to do is to read and analyze the file **Script\_Output**. It will give you important information on your simulation.

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/CheckDebug#AnalyzingtheJoboutput:Script\\_Output](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/CheckDebug#AnalyzingtheJoboutput:Script_Output)

→ Explore the output directories.

**Question 3a:** Which files are produced and where are they stored ? You did not find any files in the archive directory at `$STORE` (Jean Zay) or `$CCCSTOREDIR` (Irene)? Why not?

Answer these questions with the help of this documentation:

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Running#Theoutputfiles](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Running#Theoutputfiles)

## 3.3 How to clean up and relaunch (if needed)

If an error occurred and you need to relaunch the whole experiment, **you need to erase all the outputs created during the previous submission**, stored in the different `IGCM_OUT/LMDZ/JobName` directories:

- IDRIS: `$WORK`, `$SCRATCH` and `$STORE`,
- TGCC: `$CCCSTOREDIR`, `$CCCSCRATCHDIR` and `$CCCWORKDIR`,
- Obelix: within `/home/scratch01/login`.
- spirit : `/data/login/`
- spiritx : `/homedata/login/`

In the submit directory you also have to remove `run.card`.

To ease the cleaning you can use either one of two scripts: `clean_PeriodLength.job` and `purge_simulation.job`

The script `clean_PeriodLength.job` in `libIGCM/` can be used to clean up everything related to the **last period that failed**. This script will only clean if `run.card` exists and with `PeriodState=Fatal` or `PeriodState=Running` in it.

To use this script, stay in the submit directory `modips1/config/LMDZOR_v6/MyJobTest:`  
(do not do it now if you want to keep results of your first simulation)

```
../../../../libIGCM/clean_PeriodLength.job # Read questions and  
answer yes to erase the files.
```

When you want to remove everything related to a simulation (every period, every file etc...) you can use the script `purge_simulation.job` in `libIGCM/` directory. This script will clean up everything created during the simulation. Note that this script works only if `run.card` exists but regardless of what is in `PeriodState`: it will remove everything even for completed simulations. So be really careful, and take the time to read the dialog.

To use this script, stay in the submit directory `modips1/config/LMDZOR_v6/MyJobTest:`

```
../../../../libIGCM/purge_simulation.job # Read questions and answer  
yes to erase the files. You may have to give the experiment name.
```

### 3.4 Continue the simulation 4 more days

Now you will continue your simulation for 4 more days.

To continue a simulation, you will need to:

- Change the `DateEnd` in `config.card`, based on the last date you want to calculate. **Do not change `DateBegin` nor `PeriodLength`.**
- Modify the parameter `PeriodState` in `run.card`: currently it's equal to "Completed" and you need to put it to "OnQueue".

- Look at the value of the parameter `CumulPeriod` in `run.card` and then in the job header (`Job_*` file), modify the suffix value of lines “`#MSUB -o / -e Script_Output_`” with this number (see example below).

The example below illustrates a simulation of 1 month which is extended for one more month.

- For the exercise, you have to **adapt this case to continue your simulation for 4 more days**.

```
vi config.card
```

```
(...)
#=====
#-- leap, noleap, 360d
CalendarType=noleap
#-- Begin and end of job
#-- "YYYY-MM-DD"
DateBegin=1995-01-01
DateEnd=1995-01-31
#=====
#-- 1Y, 1M, 5D, 1D Period Length for one execution
PeriodLength=1M
(...)
```

```
vi run.card
```

```
(...)
#=====
[Configuration]
#Compute date of loop
PeriodDateBegin= 1995-02-01
PeriodDateEnd= 1995-02-28
CumulPeriod= 2
# State of Job "Start", "Running", "OnQueue", "Completed"
PeriodState= Completed
(...)
```

```
vi Job_myjob
```

```
#!/bin/ksh
#####
## IRENE TGCC/CEA ##
#####
#MSUB -r myjob # Job Name
#MSUB -o Script_Output_myjob.000001 # standard output
#MSUB -e Script_Output_myjob.000001 # error output
#MSUB -eo
#MSUB -n 569 # Number of MPI tasks (SPMD case) or cores (MPMD case)
#MSUB -x # exclusive node. To specify only for MPMD
together with the one below
```

```

#MSUB -T 1800          # Wall clock limit (seconds)
#MSUB -Q test
#MSUB -A gen0826
#MSUB -q skylake
#MSUB -m store,work,scratch

BATCH_NUM_PROC_TOT=$BRIDGE_MSUB_NPROC
set +x
(...)

```

If we want to continue for one more month, we will make modifications like this :

```
vi config.card
```

```

(...)
#=====
#-- leap, noleap, 360d
CalendarType=noleap
#-- Begin and end of job
#-- "YYYY-MM-DD"
DateBegin=1995-01-01
DateEnd=1995-02-28
#=====
#-- 1Y, 1M, 5D, 1D Period Length for one execution
PeriodLength=1M
(...)

```

```
vi run.card
```

```

(...)
#=====
[Configuration]
#Compute date of loop
PeriodDateBegin= 1995-02-01
PeriodDateEnd= 1995-02-28
CumulPeriod= 2
# State of Job "Start", "Running", "OnQueue", "Completed"
PeriodState= OnQueue
(...)

```

```
vi Job_myjob
```

```

#!/bin/ksh
#####
## IRENE TGCC/CEA ##
#####
#MSUB -r myjob      # Job Name
#MSUB -o Script_Output_myjob.000002 # standard output
#MSUB -e Script_Output_myjob.000002 # error output
#MSUB -eo

```

```

#MSUB -n 569 # Number of MPI tasks (SPMD case) or cores (MPMD case)
#MSUB -x      # exclusive node. To specify only for MPMD
together with the one below
#MSUB -T 1800      # Wall clock limit (seconds)
#MSUB -Q test
#MSUB -A gen0826
#MSUB -q skylake
#MSUB -m store,work,scratch

BATCH_NUM_PROC_TOT=$BRIDGE_MSUB_NPROC
set +x
(...)

```

Note that values for parameters `PeriodDateBegin` and `PeriodDateEnd` in `run.card` are already ready for the next period (the second month in the example). You don't need to modify these parameters.

*NB: in future versions (after rev 1571) of libIGCM you can use the script `continue_simulation.job` to do these manipulations. In this case you can use `../../../../libIGCM/continue_simulation.job` which will update `run.card` and the job header.*

→ Continue your simulation for 4 more days:

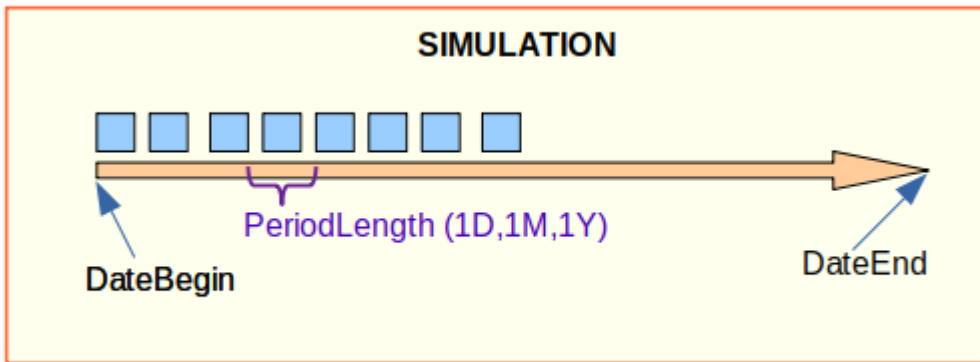
```

vi config.card # → Modify DateEnd
vi run.card # → Modify PeriodState
Vi Job_MyJobTest # → Modify suffix number on #MSUB -e and #MSUB -o lines

sbatch Job_MyJobTest / ccc_msub Job_MyJobTest / qsub Job_MyJobTest

```

**Question 3b:** How many times did the job go into the queue?



To avoid all these submissions, you will modify the parameter `PeriodNb` in the main Job. `PeriodNb` will be the number of Periods that the job tries to launch for the given CPUtime.

**Question 3c:** Create a new simulation of 5 days, always with `PeriodLength=1D`, but with a different `PeriodNb` parameter to submit the job only one time to the queue.

**Question 3d:** Look into your first simulation run.card file. How long did one day take? Did every day take the same time?

Once we are done with the test simulation, we need to be sure that we have all the desired output files and that they store all the variables required to analyse the simulation.

To know where your output files are stored at TGCC and IDRIS, you can read this page [https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Running#Theoutputfiles](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Running#Theoutputfiles)

So far we ran the simulation in TEST mode. In the next exercise you will run a simulation in DEVT mode. So you will see the difference between these two modes.

### 3.5 Create another simulation with pack

**Note for Spirit and Obelix users:**

This exercise can not be done on obelix or on spirit because the pack function is not activated on these computers.

→ Create a new experiment of `LMDZOR/clim_pdControl` type.

This time we will also enable the archiving Pack functionality. The pack is activated when `SpaceName=PROD` or `DEVT`. In this example, put `SpaceName=DEVT`.

- To test the pack functionality, set `PackFrequency=2M` in this exercise.
- Launch 4 months with a 1 month period length.

```

cp EXPERIMENTS/LMDZOR/clim_pdControl/config.card .

vi config.card
# Modify : JobName=MyJobTest2
# Modify : DateBegin=1980-01-01
# Modify : DateEnd=1980-04-30
# Modify : PeriodLength=1M
# Modify number of OMP threads if you are running on Obelix or JeanZay:
## On Irene skl or amd, change nothing for parallelization
## On JeanZay, in ATM: replace 8OMP by 5OMP
## On obelix, in ATM: replace 71MPI by 7MPI ; and 8OMP by 1OMP

# Activate pack : SpaceName=DEVT, PackFrequency=2M
# Deactivate TimeSeries and Seasonal average as before

../../libIGCM/ins_job

cd MyJobTest2

vi Job_MyJobTest2
# for information : one month on JeanZay takes between 550 and 650s in
Time Elapsed. Define the CPU Time and the queue accordingly.

sbatch Job_MyJobTest2 / ccc_msub Job_MyJobTest2

```

Continue with the next exercises while this job is running (~35-45 minutes).  
Check how it is proceeding in the queue every now and then.

**Question 3e:** explore the output directories, can you understand what was done ?  
Read this page to check that you understood correctly and what is really done  
[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Running#ConcatenationofPACKoutputs](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Running#ConcatenationofPACKoutputs)

## 3.6 Use different forcing files

Forcing files are divided in two categories: Initial State Files and Boundary files. They are defined in the **COMP/model.card** (`COMP/lmdz.card`, `COMP/orchidee.card` etc.) files.

**Initial State Files:** these files give information on the state (atmospheric concentrations, temperatures etc.) of your domain at the beginning of the simulation. To start a new simulation you can choose to use default files given by models, or to start from the state of a previous simulation, or use the atmosphere state from one, and surface from another...

Read this documentation to learn how you can do these 3 choices

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Setup#Setupinitialstateforthesimulation](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Setup#Setupinitialstateforthesimulation)



**Boundary Files:** There are two kinds of boundary files, those depending on time and those that will not change during the whole simulation.

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Setup#TheBoundaryFilessection](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Setup#TheBoundaryFilessection)

**Exercise:**

- Do a new simulation of 2 days using the restart created at the end of your simulation *MyJobTest* (exercises 3.1 -> 3.4) as an initial state file.

Reminder: to do a new simulation, copy a `config.card` file (from your previous simulation for example, or from the `EXPERIMENT/` directory) in `LMDZOR_v6/` directory ; change some parameters as the `JobName`, `DateBegin` and `DateEnd` ; use `ins_job` command to initialise the simulation directory.

NB: It's not a problem if the date of the restart is not the date preceding the beginning of your simulation. It may be better for coherence but it's not mandatory

```
vi config.card

#D-- Restarts -
[Restarts]
OverRule=y
#D- Last day of the experience used as restart for all components
RestartDate=1980-01-05
#D- Define restart simulation name for all components
RestartJobName=MyJobTest
#D- Path Server Group Login
RestartPath=$SCRATCH/IGCM_OUT/LMDZOR/TEST/clin # use $CCCSCRATCHDIR
on TGCC
```

**Question 3f:** which files are used as `start.nc`, `startphy.nc`, `sechiba_rest_in.nc` ? Read the `Script_output` file to answer this question.

**Exercise:** modify `COMP/orchidee.card` to use the PFTmap of the current year of simulation, by using the variable `${year}`. Run one more day (for this, don't forget to modify `config.card` and `run.card`).

**Question 3g:** Verify in the `Script_output` file you use the file you want.

### 3.7 CREATE\_clim and CREATE\_amip: Experiments to create initial state files and boundary conditions for LMDZ

**Note for Spirit and Obelix users:**

This exercise can not be done on obelix or on spirit

EXPERIMENT/LMDZ/CREATE\_clim and EXPERIMENTS/LMDZ/CREATE\_amip are two experiments set-up that launch the program `ce01.e`, a program based on LMDZ. This program is used to create initial state files (`start.nc` and `startphy.nc`) and boundary conditions files (`limit.nc`, `climoz_LMDZ.nc`) needed by LMDZ. The normal use of the LMDZOR\_v6 configuration is to first run the experiment CREATE\_clim or CREATE\_amip and then the experiment clim or amip. The CREATE\_clim/\_amip experiment needs to be done only one time per resolution.

You will create and launch the CREATE\_clim experiment. Note that for a standard use of CREATE\_clim you don't need to change anything.

→ Now install the submit directory for CREATE\_clim:

```
cd modips1/config/LMDZOR_v6
cp EXPERIMENTS/LMDZ/CREATE_clim/config.card .

../../libIGCM/ins_job

cd ELC-144x142x79
```

The directory ELC-144x142x79 was created and the config.card was moved inside. The resolution in the JobName was taken from the `config.card` file (parameter `ResolAtm`).

This experiment will launch the executable `ce01_144x142x79_prod.e`. It is possible to use a test queue because the run will not take more than a few minutes. You can set the test queue in the beginning of Job\_ELC-144x142x79.

```
Check paths for Relief.nc and landiceref.nc files in
COMP/lmdz.card, they need to be :
```

```
(...)
(${R_IN}/ATM/INPUT_CE0L/Relief_orig.nc , Relief.nc), \
```

```
(${R_IN}/ATM/Rugos.nc, .), \  
(${R_IN}/ATM/INPUT_CE0L/landiceref_orig.nc, landiceref.nc)  
\  
(...)
```

→ Submit the job as before:

```
sbatch Job_ELC-144x142x79 ccc_msub Job_ELC-144x142x79 /  
qsub Job_ELC-144x142x79
```

Output files are found in the directory **IGCM\_OUT/LMDZ/ELC-144x142x79** on the \$STORE at IDRIS, in \$CCCSTOREDIR at TGCC or at /home/scratch01/login at obelix.

→ Explore the script output text file in the submit directory and the files in the output directory ELC-144x142x79.

**Question 3h:** Where can you find the output? Which files are produced and where are they stored?

**Question 3i:** What type of calendar is used? How many days contains a year? Check also the number of time steps in the output file limit.nc. Do you know how you can change the calendar that has been used?

**Question 3j:** Now create a new experiment clim\_pdControl using the initial state files and boundaries files created by ELC-144x142x79. For this you will modify the path in COMP/lmdz.card for start.nc, startphy.nc, limit.nc and climoz\_LMDZ.nc files.

## 3.8 Summary on how to extract, compile and launch a simulation

### 1. Download modipsl

```
mkdir $WORK/MYFIRSTTEST ; cd $WORK/MYFIRSTTEST  
svn co https://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl  
(+usernames and passwords)
```

### 2. Extract a configuration (ex: LMDZOR\_v6)

```
cd $WORK/MYFIRSTTEST/modipsl/util  
./model LMDZOR_v6.2.2
```

```
(+usernames and passwords)
```

### 3. Compil

```
cd $WORK/MYFIRSTTEST/modips1/config/LMDZOR_v6  
./compil_lmdzor.sh [options]
```

### 4. Create experiment directory

```
cd $WORK/MYFIRSTTEST/modips1/config/LMDZOR_v6  
cp EXPERIMENTS/LMDZOR/clim_pdControl/config.card .  
vi config.card   ### Modify at least JobName=MyJobTest & // options  
../../../../libIGCM/ins_job   # At JeanZay enter your project ID  
                                # At Irene enter your project ID and default  
                                answer for other questions
```

### 5. Launch simulation

```
cd $WORK/MYFIRSTTEST/modips1/config/LMDZOR_v6/MyJobTest/  
sbatch Job_MyJobTest / ccc_msub Job_MyJobTest /  
qsub Job_MyJobTest
```

## INTERMEDIATE

### 4. Debug

We will now work on three small exercises for debugging. For these exercises we will use files prepared and stored :

- On irene, TGCC:

```
$CCCWORKDIR/../../../../igcmg/igcmg/TRAINING/MODIPSL_HandsOn_2023/LMDZOR_v6
```

- On jean-zay, IDRIS :

```
$WORK/../../../../rech/psl/commun/TRAINING/MODIPSL_HandsOn_2023/LMDZOR_v6
```

- On spirit/spiritx:

```
/projsu/igcmg/TRAINING/MODIPSL_HandsOn_2023/ORCHIDEE_OL/TP_3
```

#### 4.0 How can you analyse the Job Output: Script\_Output?

If your simulation has a problem the first thing to do is to read and analyse the file `Script_Output`. It will give you first important information on your simulation.

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/CheckDebug#AnalyzingtheJoboutput:Script\\_Output](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/CheckDebug#AnalyzingtheJoboutput:Script_Output)

#### 4.1 Debug: setup error

For this part we will work with an experiment like “MyJobTest” from **beginner** part. On Irene or on JeanZay copy the file `lmdz.card_1` from the directory above into the `lmdz.card` file in the `COMP` sub-directory in your submit directory. You can create a new Experiment.

On spirit/spiritx copy the file `sechiba.card_1` from the directory above into the `sechiba.card` file in the `COMP` sub-directory in your submit directory (use an `OOL_SEC_STO_FG2` experiment).

Now launch the simulation and debug it. Look at the output and error files in the `Debug` directory that is created when a simulation fails. Don't forget to clean up as done in exercise 2 before re-launching the simulation. Use `clean_PeriodLength.job` to do this.

**Question 4a:** What was the error?

On Irene and JeanZay, copy the lmdz.card\_2 and debug again. On spirit/spiritx, copy the sechiba.card\_2 and debug again. **Question 4b:** What was the error?

On Irene and JeanZay, copy the lmdz.card\_3 and debug again. On spirit/spiritx, copy the sechiba.card\_3 and debug again. **Question 4c:** What was the error? If you don't find the solution, try to find the difference between your actual lmdz.card file and the last one that was working.

## 4.2 Debug: error during the simulation

If you add a "print" directive in a model you can check during the simulation the output in the temporary directory RUN\_DIR/.

Try to add a "print" in LMDZ or ORCHIDEE model

```
cd modipsl/modeles/LMDZ/libf/phyImd/
vi physiq_mod.F90
Look for line
  IF (iflag_pbl/=0) THEN
And add just before
write(lunout,*) 'debug LMDZ - iflag_pbl = ', iflag_pbl

OR

cd modipsl/modeles/ORCHIDEE/src_sechiba
vi sechiba.f90
Look for the line
  IF ( river_routing .AND. nbp_glo .GT. 1) THEN
And add just before
WRITE (numout,*) 'debug ORCHIDEE - river_routing = ', river_routing
```

Note: The unit used by the `WRITE` instruction will be different from one model to another.

Re-compile your models and launch a 1 month test. Now don't wait for the end of the simulation, check your simulation id and go to the RUN\_DIR directory (on the scratchdir),

```
cd $SCRATCH/RUN_DIR/Id_job/****/ (JeanZay)
cd $CCCSCRATCHDIR/RUN_DIR/Id_job/****/ (Irene)
ls
```

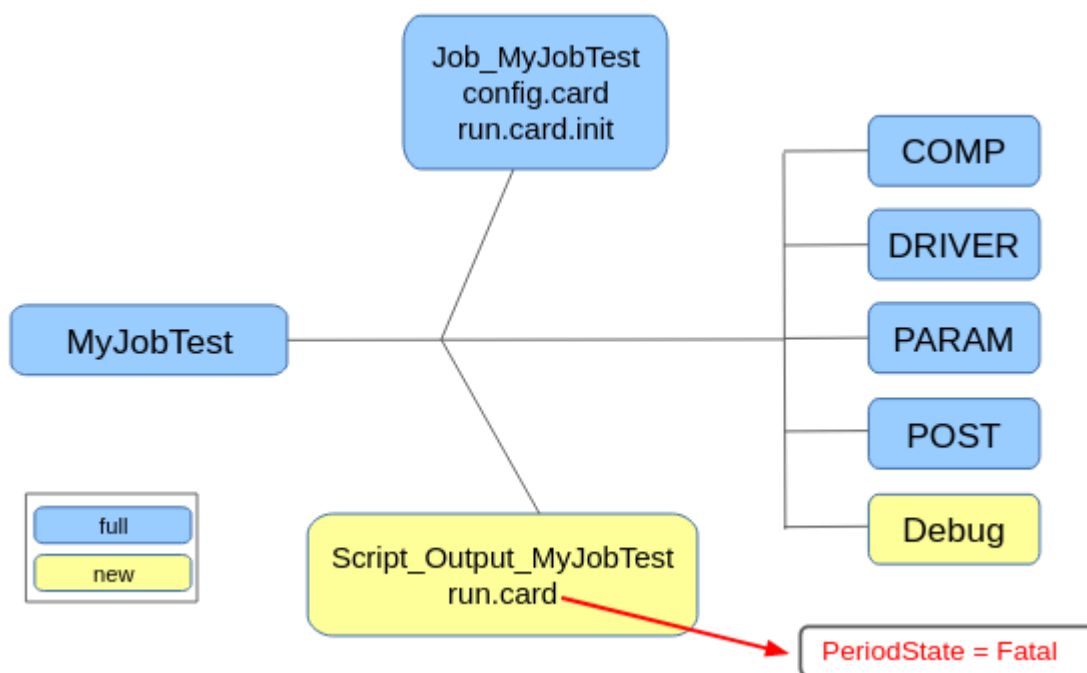
To monitor the values of your previous print you need to open, for LMDZ, `out_lmdz.x.out_***` files, or, for ORCHIDEE, `out_orchidee_****` files.

In each case you can notice that there are several output files, there is one for each OMP thread (if you are running a parallel simulation). In each of them you will find the output text print for this specific thread or proc.

If you have a problem during a simulation, you can try to debug by adding prints in your model code.

### 4.3 Compilation in debug mode

When a simulation does not finish successfully, it creates a new directory called `Debug` in your experiment directory.



You can read the description of this directory here

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/CheckDebug#TheDebugdirectory](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/CheckDebug#TheDebugdirectory)

Errors are stored in the `Debug/***_out_lmdz.x.err` file even for other models than LMDZ.

In this file you will find information for each proc mpi. You can read that there is a bug but there is no more information on the localisation of this bug in the source code. It's because to compile we use permissive options that will not track the bug precisely.

To obtain more information on a bug we need to recompile in debug mode. For this you will use the option "debug" of the compilation script.

```
./compile_lmdzor.sh -debug
```

Create a new simulation and modify the `config.card` file to indicate that we will use the debug executable.

```
vi config.card → modify Optmode=debug
```

This new simulation will crash again, but now you will find more information in the file `Debug/***_out_lmdz.x.err` file.

In most cases you will have the line number indicating where the code crashed. To find this information, you can read all the files or look for the keyword “gcm” (the name of LMDZ main program).

For example:

```
9fortrtl: severe (174): SIGSEGV, segmentation fault occurred
9Image      PC          Routine      Line   Source
9lmdz.x      0000000005146D79 Unknown      Unknown Unknown
9libpthread-2.17.s 00002AAAB10C45D0 Unknown      Unknown Unknown
9lmdz.x      000000000AA8C70 physiq_mod_mp_phy      1619 physiq_mod.f90
9lmdz.x      00000000009872B2 callphysiq_mod_mp      81
callphysiq_mod.f90
9lmdz.x      0000000000979F74 calfis_loc_      729 calfis_loc.f
9lmdz.x      0000000000687DAA call_calfis_mod_m      214
call_calfis_mod.f90
9lmdz.x      00000000004AF7AE leapfrog_loc_      807 leapfrog_loc.f
9lmdz.x      0000000000427DCA MAIN_      454 gcm.f90
9libiomp5.so      00002AAAB3DE0ED3 __kmp_invoke_micr Unknown Unknown
9libiomp5.so      00002AAAB3DA3726 Unknown      Unknown Unknown
9libiomp5.so      00002AAAB3DA50FD __kmp_fork_call      Unknown Unknown
9libiomp5.so      00002AAAB3D66020 __kmpc_fork_call      Unknown Unknown
9lmdz.x      0000000000424A19 MAIN_      445 gcm.f90
9lmdz.x      000000000041EC62 Unknown      Unknown Unknown
9libc-2.17.so     00002AAAB4105495 __libc_start_main      Unknown Unknown
9lmdz.x      000000000041EB69 Unknown      Unknown Unknown
```



is telling you that there is a problem at line 1619 of `physiq_mod.f90`, called by `callphysiq_mod.f90` at line 81, called by `calfis_loc.f` at line 729, called by `call_calfis_mod.f90` at line 214, called by `leaproc_loc.f` at line 807, called by `gcm.f90` at line 454.

Warning : all line numbers don't refer to the code sources, but to pre-compiled sources

```
In LMDZ : modeles/LMDZ/libo/computer_resolution/.config/ppsrc/  
In ORCHIDEE: modeles/ORCHIDEE/build/ppsrc/  
In INCA : modeles/INCA/build/ppsrc/  
In NEMO/PISCES :  
modeles/NEMOGCM/CONFIG/ORCA1_LIM3_PISCES/BLD/ppsrc
```

## 4.4 Use of `RUN_DIR` directory to debug without libIGCM infrastructure

This function can be used with any experiment and configuration. Continue with the same `LMDZOR_v6` or `ORCHIDEE_OL` configuration as in previous exercises.

Sometimes, particularly in the development phase, it could be useful (and more effective) to have all the information of the run in the same directory : that allows you to run directly into the `RUN_DIR` directory, using a `Job_debug` to be launched. To activate this debug functionality (available from libIGCM rev 1569), have a look on the following documentation [http://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/CheckDebug#UseofRUN\\_DIRdirectorytorunwithlibIGCMinfrastructure](http://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/CheckDebug#UseofRUN_DIRdirectorytorunwithlibIGCMinfrastructure)

First, be sure to use a quite recent libIGCM (i.e > rev 1569) otherwise :

```
cd $WORK/MYFIRSTTEST/modipsl  
mv libIGCM libIGCM_old  
svn co https://forge.ipsl.jussieu.fr/libigcm/svn/trunk/libIGCM libIGCM
```

Note : “`svn info $WORK/MYFIRSTTEST/modipsl/libIGCM`” gives you the revision.

Then, create a new experiment called `MyJobTest.debug` (see part 3.8 if needed)

```
cd $WORK/MYFIRSTTEST/modipsl/config/LMDZOR_v6  
cp EXPERIMENTS/LMDZOR/clim_pdControl/config.card .  
vi config.card   ### Modify JobName=MyJobTest.debug & // options  
../../../../libIGCM/ins_job
```

Enable "the debug into RUN\_DIR" functionality in the Job Job\_MyJobTest.debug before submitting the Job.

```
DRYRUN=4
```

Follow the message at the end of the Script\_Output :

```
#####  
#      DEBUG PHASE : CREATION OF RUN_DIR      #  
#####  
  
You are in development or debug phase  
You can run directly into the running directory which is here  
...../your_login/RUN_DIR/.....  
Inside the run directory you will find a Job_debug_MyJobTest.debug  
to be used to launch the run as follows :  
ccc_msub (or sbatch or ...) Job_debug_MyJobTest.debug
```

## INTERMEDIATE

### 5. Create Time Series

A [Time Series](#) is a file which contains a single variable over the whole simulation period or for a shorter period.

- Write frequency is defined in the `config.card` file by `TimeSeriesFrequency` option.
- The Time Series are set in the `COMP/*.card` files by the `TimeSeriesVars2D` and `TimeSeriesVars3D` options.

The goal of this exercise is to understand how to control the creation of Time Series.

#### 5.1 Launch 2 years with default Time Series

In order to have a model that runs quickly, we will use an ORCHIDEE offline configuration with a regional domain (small horizontal domain). The principle will be the same for other configurations.

- Install a new modipsl, download the configuration ORCHIDEE\_4\_1 and compile (~15 minutes).

If you are working on **Jean-Zay/IDRIS**, don't forget to log in to the preprocessing front-end (otherwise the compilation of XIOS will not work):

```
ssh jean-zay-pp
```

If you are on Irene/TGCC, change `$WORK` to `$CCCWORKDIR` in the code below.

```
mkdir $WORK/MYPOSTTEST; cd $WORK/MYPOSTTEST
svn co https://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk
modipsl
cd modipsl/util
./model ORCHIDEE_4_1
cd ../config/ORCHIDEE_OL
./compile_orchidee_ol.sh
```

If you are working on **Jean-Zay/IDRIS**, don't forget to log out from the preprocessing front-end

`exit`

In this configuration (ORCHIDEE offline), you don't need to create the experiment directory. Instead different experiment directories already exist: `OOL_SEC`, `OOL_SEC_STO_FG**` and `SPINUP_ANALYTIC_**` are experiments that follow the standard rules described in this tutorial. You can find them in `modipsl/config/ORCHIDEE_OL/` directory.

Note that the `DRIVER` directory does not exist for this configuration, but "drivers" files are found in the `COMP` directory.

We will work here with the `OOL_SEC_STO_FG2` experiment which is a full ORCHIDEE offline setup with `sechiba` and `stomate` components (refer to ORCHIDEE training for more information about these components).

- Copy the `OOL_SEC_STO_FG2` directory into a new one (named `MyPostExp` for example).
- Modify `config.card` and `run.def`.
- Create the job.

How to modify `run.def`:

- We use a regional domain by setting `LIMIT` parameters in `PARAM/run.def`.

How to modify `config.card`:

- Because we use a smaller domain, no need to run on many processors: change to `3MPI` (instead of `31MPI`) for `orchidee_ol`.
- It is better to run ORCHIDEE offline configurations with `PeriodLength=1Y`.
- Just like every time you configure a new experiment, you have to change `JobName` (your experiment name = `MyPostExp` in this exercise)
- In order to write all outputs in the `STORE` directory, use `SpaceName=DEVT`.
- For `OOL_SEC_STO_FG2` experiment, the default `Pack` frequency is `10Y`. For this exercise, change it to `1Y` (`PackFrequency=1Y`).
- For `OOL_SEC_STO_FG2` experiment, the default Time Series frequency is `10Y`. For this exercise, change it to `2Y` (`TimeSeriesFrequency=2Y`).
- We will not calculate Seasonal Means, so use `SeasonalFrequency=NONE`.

```
cd $WORK/MYPOSTTEST/modipsl/config/ORCHIDEE_OL/  
cp -r OOL_SEC_STO_FG2 MyPostExp  
cd MyPostExp
```

```

vi PARAM/run.def
# Add these lines
LIMIT_WEST = -10.
LIMIT_EAST = 20.
LIMIT_NORTH = 60.
LIMIT_SOUTH = 30.

vi config.card # => Change JobName, SpaceName=DEVT
                # DateEnd=1902-12-31, PeriodLength=1Y,
                # => in [Executable]:
                # OOL= (orchidee_ol_${OptMode}, orchidee_ol, 3MPI)
                # IOS= (xios_server_${OptMode}.exe, xios.x, 1MPI)
                # => in [Post]:
                # PackFrequency=1Y, TimeSeriesFrequency=2Y,
                # SeasonalFrequency=NONE

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

```

- In the main Job, set `PeriodNb=2` and `#SBATCH --time=30` or `#SBATCH --time=00:30:00` (IDRIS, Spirit(x)) or `#MSUB -T 1800` (TGCC).
- Submit the job (using `sbatch`, `ccc_msub` or `qsub` depending on the machine).

**Question 5a:** Once the simulation and post-processing are complete (~15 minutes), check Time Series in the archive directory (see in the following directories: `IGCM_OUT/[...]/JobName/**/Analyse/TS_MO`).

## 5.2 Add variables to Time Series and relaunch with the TimeSeriesChecker.job

All variables in the output files can be used to create Time Series. A selection of variables are done by default and are defined in card files (`COMP/*.card`).

Now we will add the creation of Time Series for `z0h` ("Surface roughness for heat") and `z0m` ("Surface roughness for momentum") variables.

- First be sure that they are produced and exist in the file `sechiba_history.nc` (in directory `IGCM_OUT/[...]/JobName/SRF/Output/MO/`).
- Then add them in `COMP/sechiba.card`:

```
[Post_1M_sechiba_history]
```

```
Patches = ()
GatherWithInternal= (lon, lat, veget, time_counter, Areas, Contfrac, time_centered,
time_centered_bounds)
TimeSeriesVars2D = (nobiofrac, alb_nir, alb_vis, ....., z0h, z0m)
...
```

→ Read the documentation about the script [TimeSeries\\_Checker.job](#) and launch it to create missing and new Time Series.

**Question 5b:** How to check that “z0h” and “z0m” variables exist in the simulation file?

**Question 5c:** How to use [TimeSeries\\_Checker.job](#)?

**Question 5d:** Check that Time Series for z0h and z0m were created.

## INTERMEDIATE

# 6. Monitoring and Inter-monitoring

The monitoring is a web-interface tool that visualizes the global mean over time for a set of key variables. The inter-monitoring web-interface allows to simultaneously monitor various simulations. More details can be found in:

[http://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Running#Monitoringandintermonitoring](http://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Running#Monitoringandintermonitoring)

## 6.1 Monitoring

For example, you can visualize the monitoring on the web for the **CM61-LR-pi-03 simulation** (IPSLCM6-CMIP6 piControl simulation performed on Curie-TGCC).

[https://thredds-su.ipsl.fr/thredds/fileServer/tgcc\\_thredds/work/p86maf/IPSLCM6/PROD/piControl/CM61-LR-pi-03/MONITORING/index.html](https://thredds-su.ipsl.fr/thredds/fileServer/tgcc_thredds/work/p86maf/IPSLCM6/PROD/piControl/CM61-LR-pi-03/MONITORING/index.html)

## 6.2 Inter-monitoring

### 6.2.1 from web interface tool “Inter-monitoring application”

Now you will use the web interface tool “inter-monitoring” to superpose several simulations. The default inter-monitoring is found at address:

<http://webservices.ipsl.jussieu.fr/interMonitoring/>.

For this exercise choose following 2 simulations: **CM61-LR-pi-03** (IPSLCM6-CMIP6 piControl simulation performed on Curie-TGCC) and **CM61-pi-valid.02.JZ** (IPSLCM6 piControl simulation performed on JeanZay-IDRIS). These simulations have been used to validate porting on JeanZay.

To do the inter-monitoring comparison, set the corresponding paths :

- CM61-LR-pi-03:

[http://thredds-su.ipsl.fr/thredds/catalog/tgcc\\_thredds/work/p86maf/IPSLCM6/PROD/piControl](http://thredds-su.ipsl.fr/thredds/catalog/tgcc_thredds/work/p86maf/IPSLCM6/PROD/piControl)

- CM61-pi-valid.02.JZ:

[http://thredds-su.ipsl.fr/thredds/catalog/tgcc\\_thredds/work/p86caub/IPSLCM6/DEVT/piControl](http://thredds-su.ipsl.fr/thredds/catalog/tgcc_thredds/work/p86caub/IPSLCM6/DEVT/piControl)

And follow the following Mini how to use the inter-monitoring :

→ Go to <http://webservices.ipsl.jussieu.fr/interMonitoring/>

1. Enter the first path and click on the button List Directories.
2. You'll see a list of all simulations at this path. Go back to step 1.
3. Go back to step 1, enter the second path **and click on Append Directories**.

4. You'll now see all simulations on the 2 paths. Choose the two simulations with the corresponding names. (use the mouse and type ctrl to select only 2 simulations). Click on Search files.
5. Select one variable and click on Validate.
6. Choose default setting for "plot01:Time series" and click on Validate. Then click on the button below called "Prepare and run the ferret script".
7. Now a ferret script will appear on the screen and one image. Click on the button "Run this script on the server" below on the page. The inter-monitoring for all variables will now appear on the screen.

Note: CM61-pi-valid.02.JZ simulation is shorter than CM61-LR-pi-03. Go back to Step 4 to select only the range 1850-1900 (using "Dates range" cursor) which is the common period between both simulations then click again on "Prepare and run the ferret script".

### 6.2.2 from web interface tool "simu finder application"

There is another way to configure your inter-monitoring, for this you can use the "simu finder application" (<http://webservices.ipsl.jussieu.fr/simuFinder/>). In the left window of this application you can write "tags" to describes simulations you want to visualize. For example: login, Tagname, JobName ((everything you think necessary to find a simulation).

You can re-do the previous inter-monitoring for simulations: CM61-LR-pi-03 and CM61-pi-valid.02.JZ. FOr this follow the following Mini how:

- Go to <http://webservices.ipsl.jussieu.fr/simuFinder>
  1. in left window write "CM61-LR-pi-03". There is only one simulation with this name, you can verify the path to be sure it's the one you want use (login : p86maf, TagName: PROD, ExperimentName: picontrol)
  2. click on the path on the right window and drag it on the bottom one
  3. remove the previous tag in the left window, and write a new one "CM61-pi-valid.02.JZ". One more time there is only one simulation with this name.
  4. click on the path and drag it to the bottom window
  5. click on "run intermonitoring" button and now you are on a specific version of "inter-monitoring application". Click on "search files" and go to the step5 of exercise 6.2.1.



## INTERMEDIATE

### 7. How to REDO part of a simulation

Sometimes, due to machine problems (or other unknown reasons), output files are missing. Here is how to recover missing output files. The general method is explained on FAQ of the documentation:

[http://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/FAQ#HowdoIrestartasimulationtorecovermissingoutputfiles](http://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/FAQ#HowdoIrestartasimulationtorecovermissingoutputfiles)

As an example, we suggest you to:

- launch a 6 days simulation of LMDZOR experiment, with pack frequencies of 2 days
- remove output files for 1 pack of the simulation
- apply the method to recover missing output files

#### 7.1 Launch a 6 days simulation of LMDZOR experiment

```
cd modipsl/config/LMDZOR_v6
cp EXPERIMENTS/LMDZOR/lim_pdControl/config.card .

vi config.card
# Modify JobName=MyJobTest-6D
# SpaceName=DEVT
# Note : REDO method does not work with TEST as SpaceName
# DateBegin=1980-01-01
# DateEnd=1980-01-06
# PeriodLength=1D
# PackFrequency=2D
# TimeSeriesFrequency=NONE
# SeasonalFrequency=NONE
# Modify Executable part for the parallelization
[Executable]
ATM= (gcm_${ResolAtm}_${OptMode}.e, lmdz.x, 71MPI, 8OMP)
SRF= ("", "")
SBG= ("", "")
IOS= (xios_server_${OptMode}.exe, xios.x, 1MPI)
```

```

# At obelix only, change to 7 MPI and 1 OMP in
# At Irene, change nothing for parallelization
# At JeanZay, change 8 OMP by 5 or 10

../../libIGCM/ins_job # At JeanZay enter your project ID
                        # At Irene enter your project ID and default answer for other
questions

cd MyJobTest-6D
vi Job_MyJobTest-6D
    # Modify the job header to launch on test queue
    # Modify the periodNb to not relaunch the simulation between two periods of simulation.
    PeriodNb=6

    # Modify the LMDZ's output files frequencies (cancelled monthly outputs, and activated
    daily ones)
vi COMP/lmdz.card
output_level_histmth = NONE
output_level_histday = 10

    # Modify the ORCHIDEE's outputs files frequencies (cancelled monthly outputs, and
    activated daily ones)
vi COMP/orchidee.card
output_freq_sechiba_history = 1d
output_freq_sechiba_out_2 = 10800s
output_freq_sechiba_history_4dim = 1d

Vi COMP/stomate.card
output_freq_stomate_history = 1d
output_freq_stomate_ipcc_history = 1d

Submit the job

```

**For Spirit users:** You can use the REDO method with Orchidee\_ol

```

cd modips1/config/ORCHIDEE_OL
cp -r OOL_SEC_STO_FG2 MyJobTest-6D
cd MyJobTest-6D

```

```

vi config.card
    # Modify JobName=MyJobTest-6D
    # SpaceName=DEVT
    # Note : REDO method does not work with TEST as SpaceName
    # DateBegin=1980-01-01
    # DateEnd=1980-01-06
    # PeriodLength=1D
    # PackFrequency=2D ### there is no pack on spirit
    # TimeSeriesFrequency=NONE
    # SeasonalFrequency=NONE
    # Modify Executable part for the parallelization
[Executable]
SRF= ("", "")
SBG= ("", "")
OOL= (orchidee_ol_${OptMode}, orchidee_ol, 31MPI)
IOS= (xios_server_${OptMode}.exe, xios.x, 1MPI)

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

vi Job_MyJobTest-6D
    # Modify the periodNb to not relaunch the simulation between
two periods of simulation.
PeriodNb=6

    # Modify the ORCHIDEE's outputs files frequencies (cancel
monthly outputs, and
    activate daily ones)

vi COMP/sechiba.card
output_freq_sechiba_history = 1d
output_freq_sechiba_out_2 = 10800s
output_freq_sechiba_history_4dim = 1d

[OutputFiles]
List=
                                (sechiba_history.nc,
${R_OUT_SRF_O_D}/${PREFIX}_1D_sechiba_history.nc,
Post_1D_sechiba_history), \
    (sechiba_history_4dim.nc,
${R_OUT_SRF_O_D}/${PREFIX}_1D_sechiba_history_4dim.nc, NONE), \
    (sechiba_out_2.nc,
${R_OUT_SRF_O_H}/${PREFIX}_HF_sechiba_out_2.nc, NONE)

Change [Post_1M_sechiba_history] to [Post_1D_sechiba_history]

vi COMP/stomate.card

```

```

output_freq_stomate_history = 1d
output_freq_stomate_ipcc_history = 1d
output_freq_stomate_history_4dim = 1d

[OutputFiles]
List=                                     (stomate_history.nc,
${R_OUT_SBG_O_D}/${PREFIX}_1D_stomate_history.nc,
Post_1D_stomate_history), \
    (stomate_ipcc_history.nc,
${R_OUT_SBG_O_D}/${PREFIX}_1D_stomate_ipcc_history.nc,
Post_1D_stomate_ipcc_history), \
    (stomate_history_4dim.nc,
${R_OUT_SBG_O_D}/${PREFIX}_1D_stomate_history_4dim.nc, NONE)

Change
[Post_1M_stomate_history] to [Post_1D_stomate_history]
[Post_1M_stomate_ipcc_history] to [Post_1D_stomate_ipcc_history]

Submit the job :
sbatch Job_MyJobTest-6D

```

## 7.2 Remove daily output file for ATM component (*SBG on Spirit*) of the days 3 and 4 (i.e. 1980-01-03/04)

```

Check ($CCCSTOREDIR on TGCC)
$STORE/IGCM_OUT/LMDZOR/DEVT/clim/MyJobTest-6D/ATM/Output/DA/MyJob
Test-6D_19800103_19800104_1D_histday.nc exists...then remove it
rm -f
$STORE/IGCM_OUT/LMDZOR/DEVT/clim/MyJobTest-6D/ATM/Output/DA/MyJob
Test-6D_19800103_19800104_1D_histday.nc

```

### **Spirit users:**

```

Check
/data/$USER/IGCM_OUT/OL2/DEVT/test/MyJobTest-6D/SBG/DA/MyJobTest-
6D_19800103_19800103_1D_stomate_history.nc
/data/$USER/IGCM_OUT/OL2/DEVT/test/MyJobTest-6D/SBG/DA/MyJobTest-
6D_19800103_19800103_1D_stomate_ipcc_history.nc

```

```

/data/$USER/IGCM_OUT/OL2/DEVT/test/MyJobTest-6D/SBG/DA/MyJobTest-
6D_19800104_19800104_1D_stomate_history.nc
/data/$USER/IGCM_OUT/OL2/DEVT/test/MyJobTest-6D/SBG/DA/MyJobTest-
6D_19800104_19800104_1D_stomate_ipcc_history.nc

rm -f
/data/$USER/IGCM_OUT/OL2/DEVT/test/MyJobTest-6D/SBG/DA/MyJobTest-6D_198001
0[34]*.nc

```

### 7.3 Apply the method to redo day 3 and 4 of the simulation (to recover missing output file)

```

# Handling of the restart files of the end of the first period (day one and two) to redo the
second period (day three and four) of the new simulation

mkdir -p $STORE/IGCM_OUT/LMDZOR/REDO/clim/MyJobTest-6D
cd $STORE/IGCM_OUT/LMDZOR/REDO/clim/MyJobTest-6D
mkdir -p RESTART
cd RESTART
cp
../../../../../DEVT/clim/MyJobTest-6D/RESTART/MyJobTest-6D_19800101_
19800102_restart.tar .

# Set up of the new simulation

cd modips1/config/LMDZOR_v6
cp -pr MyJobTest-6D MyJobTest-6D-REDO
cd MyJobTest-6D-REDO

# In this new directory, change the run.card and config.card file and set the following
parameters to:

vi run.card
# we will do as if the REDO simulation already done the two
firsts days, and now we want to continue our simulation
# PeriodDateBegin= 1980-01-03
# PeriodDateEnd= 1980-01-03
# CumulPeriod= 3 # Specify the same period in the run.card of initial simulation
# PeriodState= OnQueue
# SubmitPath= ...modips1/config/LMDZOR_v6/MyJobTest-6D-REDO
# remove lines for periods 3 to 6 at the end of the file (because in our REDO
simulations, these periods don't yet exist)

```

```
vi config.card
# you don't need to change the name of the simulation, neither the DateBegin of the
simulation
  # SpaceName=REDO
  # DateEnd= 1980-01-04

Submit the Job
```

### Spirit users:

# Handling of the restart files of the end of the first period (day one and two) to redo the second period (day three and four) of the new simulation

```
mkdir -p /data/$USER/IGCM_OUT/LMDZOR/REDO/test/MyJobTest-6D
cd /data/$USER/IGCM_OUT/LMDZOR/REDO/test/MyJobTest-6D
```

```
mkdir -p OOL/Restart
cp
../../../../../DEVT/test/MyJobTest-6D/OOL/Restart/MyJobTest-6D_19800102
_driver_rest.nc OOL/Restart
```

```
mkdir -p SBG/Restart
cp
../../../../../DEVT/test/MyJobTest-6D/SBG/Restart/MyJobTest-6D_19800102
_stomate_rest.nc SBG/Restart
```

```
mkdir -p SRF/Restart
cp
../../../../../DEVT/test/MyJobTest-6D/SRF/Restart/MyJobTest-6D_19800102
_sechiba_rest.nc SRF/Restart
```

### # Set up of the new simulation

```
cd modips1/config/ORCHIDEE_OL
cp -r MyJobTest-6D MyJobTest-6D-REDO
cd MyJobTest-6D-REDO
```

# In this new directory, change the run.card and config.card file and set the following parameters to:

```
vi run.card
  # we will do as if the REDO simulation already done the two
firsts days, and now we want to continue our simulation
  # PeriodDateBegin= 1980-01-03
```

```
# PeriodDateEnd= 1980-01-03
# CumulPeriod= 3 # Specify the same period in the run.card of initial simulation
# PeriodState= OnQueue
# SubmitPath= ...modipsl/config/LMDZOR_v6/MyJobTest-6D-REDO
# remove lines for periods 3 to 6 at the end of the file (because in our REDO
simulations, these periods don't yet exist)
vi config.card
# you don't need to change the name of the simulation, neither the DateBegin of the
simulation
# SpaceName=REDO
# DateEnd= 1980-01-04

Submit the Job
```

Once the job is finished you can have a look at the file:

```
$STORE/IGCM_OUT/LMDZOR/REDO/clim/MyJobTest-6D/ATM/Output/DA/MyJobTest-6D_1
9800103_19800104_1D_histday.nc
```

Once the new run is validated (same results as the previous one: comparison of restart files at the end of the second period - you can use “cdo diffn file1.nc file2.nc”), you can copy the new file in the initial directory:

```
cp
$STORE/IGCM_OUT/LMDZOR/REDO/clim/MyJobTest-6D/ATM/Output/DA/MyJob
Test-6D_19800103_19800104_1D_histday.nc
$STORE/IGCM_OUT/LMDZOR/DEVT/clim/MyJobTest-6D/ATM/Output/DA/.
```

### Spirit users:

```
cp
/data/$USER/IGCM_OUT/LMDZOR/REDO/test/MyJobTest-6D/SBG/MyJobTest-
6D_1980010[34]*.nc
/data/$USER/IGCM_OUT/LMDZOR/DEVT/test/MyJobTest-6D/SBG/Output/DA/.
```

## INTERMEDIATE

# 8. Modify output using XIOS

The outputs of the IPSL models are managed by the XIOS library. Read the documentation [https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Tools#ShortpresentationonhowtomanageoutputsfilesusingXIOSmodelinIPSLconfigurations](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Tools#ShortpresentationonhowtomanageoutputsfilesusingXIOSmodelinIPSLconfigurations) to have an idea on how you can write a diagnostic in an output file using XIOS.

## 8.1 Create a new output file for ORCHIDEE

The different output files and their contents in ORCHIDEE are defined in the `modeles/ORCHIDEE/src_xml/file_def_orchidee.xml` file.

This file can be modified to write specific variables in the output files. The key words `_AUTO_` can be changed directly in the file or using the variables in `orchidee.card`, `sechiba.card` and `stomate.card` (section `[UserChoices]`). To save a variable, the file must also be listed in `orchidee/sechiba/stomate.card` (section `[OutputFiles]`). The same method is used when working in coupled mode with LMDZ or using ORCHIDEE in offline mode. The only difference is the name of the `comp.card`: `orchidee.card` when coupled to LMDZ and `sechiba.card` when running in offline mode. For this exercise, use a test in offline mode (like in exercise 5.1) as it is faster to run.

Here you should create a new output file from ORCHIDEE containing only the daily average rainfall and snowfall. The variables are already output from the model using `xios_send_field` and are declared in the `field_def_orchidee.xml` file with the id `precip_rain` and `precip_snow`. If you want to see where in the model they are written, search for `precip_` and `xios` in `ORCHIDEE/src*/*` using

```
grep precip_src_*/* | grep xios
```

in the `modipsl/modeles/ORCHIDEE/` folder.

To create this new file you can do the following:

- Continue in the same `modipsl` where you installed ORCHIDEE offline in exercise 5.
- Add a section in `file_def_orchidee.xml` with these specifications. (Take example of how the first `sechiba_history` file is defined and do the same)
  - a. The file should be named `myoutput_orch.nc`
  - b. The name of the variables in the output file should be “`rainfall`” and “`snowfall`”
  - c. Keep the default unit, mm/s



- d. Choose value for Output\_level (all variable with a lesser level will be write in the output file)
- e. File output frequency should be daily average. You have to set the file attribute `output_freq="1d"`
- f. File attribute `enabled=.TRUE.`

```
<file id="sechiba0" name="myoutput_orch" output_level="1" output_freq="1d"
enabled="true">

<field_group group_ref="remap_1d" grid_ref="grid_landpoints_out" >
  <field field_ref="precip_rain" name="rainfall" level="1"/>
  <field field_ref="precip_snow" name="snowfall" level="1"/>
</field_group>

</file>
```

→ Create a new experiment called "MyPostExp2" similar to MyPostExp used in 4.1. You can start from a copy of MyPostExp as follows:

```
cp -r MyPostExp MyPostExp2
cd MyPostExp2

vi config.card    # Change JobName

# Remove files related to MyPostExp
rm Job_MyPostExp run.card Script_Output_MyPostExp.000001

# Create a new job
../../libIGCM/ins_job
```

Note : you don't need to recompile because you haven't made any change in the code. The xml files are read directly during the execution.

→ Add the new file to be stored in `COMP/sechiba.card` (see example of `1M_sechiba_history.nc`)  
In `[OutputFiles]` section :

```
(myoutput_orch.nc, ${R_OUT_SRF_O_D}/${PREFIX}_1D_myoutput_orch.nc,
Post_1D_myoutput_orch), \
```

Also define the new Post section "`Post_1D_myoutput_orch`" and add the two new variables to be produced as TimeSeries.

```
[Post_1D_myoutput_orch]
Patches = ()
```

```
GatherWithInternal = (lon, lat, time_counter, time_centered,
time_centered_bounds)
TimeSeriesVars2D = (rainfall, snowfall)
ChunckJob2D = NONE
TimeSeriesVars3D = ()
ChunckJob3D = NONE
Seasonal = ON
```

→ Submit using `sbatch`, `ccc_msub` or `qsub` depending on the platform.

**Question 8a:** Verify that this new file is created and TimesSeries of two variables exist : since these variables are daily outputs, you have to search into ...SRF/Analyse/TS\_DA/

## 8.2 Enable a new output file in LMDZ

Similarly to the ORCHIDEE mechanism described above, the different output files and their contents in LMDZ are defined in the files

`modeles/LMDZ/DefLists/file_def_*_lmdz.xml`.

You can see that there are quite a few of these files. Each one describes the contents of one possible output file for LMDZ. These files may differ by the time averaging used to output variables (monthly means or instantaneous values for example) or may come from different parts of the LMDZ model (the \*COSP\* ones for example are output by the COSP simulator embedded in LMDZ).

As for the ORCHIDEE example above, the files can be modified to contain specific output if needed. The key words `_AUTO_` can be changed directly in the file or using the variables in `lmdz` (section `[UserChoices]`). To save a variable, the file must also be listed in `lmdz.card` (section `[OutputFiles]`) but you will see that most of the files are mentioned (and saved) in the default `lmdz.card`.

In this exercise, you will enable a new output file from LMDZ containing high frequency hourly average values of the sea-level pressure. Sea-level pressure is already output from the model using `xios_send_field` and is declared in the `field_def_lmdz.xml` with the id `slp`. So you will just need to declare the new file without modifying the code or the `field_def.xml` file. If you want to see where in the model they are written, all LMDZ output variables are defined and written in the LMDZ routine `phys_output_write_mod.F90` which can be found in the

`modipsl/modeles/LMDZ/libf/phy_lmd/` folder.

If you look at the files mentioned above, you will notice that there is already a file `modeles/LMDZ/DefLists/file_def_histhf_lmdz.xml` containing specifications to output average values every 3 hours for a long list of variables in a file called `histhf`. We will modify this file to output the desired file and variable.

To do this, follow the following:

- Continue in the same modipsl where you installed LMDZOR in exercise 2.1
- Modify `LMDZ/DefLists/file_def_histhf_lmdz.xml` with the specifications:
  - a. The file should be named `myoutput_lmdz.nc`
  - b. The level of the variable `slp` should be set to 5. We will set `output_level` to 5; that means that only variables with a `level` less than or equal to 5 will be written in the file. You can see that for the default LMDZ output files, the `output_levels` parameters are not defined (they are set to `_AUTO_`). Values are managed from the `lmdz.card` file in the section `[UserChoices]`.
  - c. File output frequency should be hourly average. You have to set the file attribute `output_freq="1h"`
  - d. File attribute `enabled=TRUE`

```
<file id="myoutputid" name="myoutput_lmdz" output_freq="1h"
output_level="5" enabled="true" compression_level="4">
...
  <field field_ref="slp" level="5" />
...
</file>
```

- Create a new experiment called "MyJobTestLMDZ" similar to `MyJobTest` used in 2.1. You can start from a copy of `MyJobTest`: as follows:

```
cp -r MyJobTest MyJobTestLMDZ
cd MyJobTestLMDZ

vi config.card          # Change JobName, Set Date=1980-01-05, Set
PeriodLength=5D

# Remove files related to MyPostExp
rm Job_MyPostExp run.card Script_Output_MyPostExp.000001

# Create a new job
../../../../libIGCM/ins_job

# Make sure the following two lines are in the header of your job file
# for jean-zay
#SBATCH --cpus-per-task=4
#SBATCH --qos=qos_cpu-dev
```

Note : you don't need to recompile because you didn't make modifications in the code. The xml files are read directly during the execution.

- Add the new file to be stored in `COMP/lmdz.card` (see example of `histhf.nc`)  
In `[OutputFiles]` section :

```
(myoutput_lmdz.nc, ${R_OUT_ATM_O_H}/${PREFIX}_HF_myoutput_LMDZ.nc, NONE), \
```

→ Submit using `sbatch`, `ccc_msub` or `qsub` depending on the platform.

**Question 8b:** Verify that this new file is created and that it contains the `slp` variable.

## 8.3 XIOS in other models

NEMO, REPROBUS, and INCA models also use XIOS to manage output files.

Where can you find the xml files for these models?

```
NEMO : modipsl/config/***/GENERAL/PARAM/ (note that directory
will be copy in your simulation directory)
REPROBUS : modipsl/modeles/REPROBUS/XML
INCA : modipsl/modeles/INCA/src/INCA_XML
```

These 3 models use XIOS in the same way as LMDZ and ORCHIDEE.

You can find here a documentation for XIOS in Inca model

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Models/INCA#ManageoutputusingXIOS](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Models/INCA#ManageoutputusingXIOS)

And here for XIOS in Nemo model <https://zenodo.org/record/3248739#.XhhOAOEo8ax> on page 229.

## 8.4 XIOS and CMIP workflow

In addition to (or instead of) the outputs generated by default by the components, it is possible to generate "CMIP6" type outputs, i.e. in a format close to the one constrained by the Data Request CMIP6. This format has the following specifications:

- time series (one variable per file)
- file name format like "hur\_CFday\_IPSL-CM6A-LR\_TRAINING\_gr\_18500101-18501231.nc" containing the name of the variable, frequency table to which it belongs, model name, simulation name, grid type and period covered by this file.

This protocol requires usage of additional xml files *ping\_file.xml* and *dr2xml\_file.xml* (in PARAM directory of the experiment). It allows on one hand to establish the correspondence between the fields produced by the model and those requested by the Data Request CMIP6 (*ping\_file.xml*) and on the other hand to satisfy the data request in terms of frequency, name,... (*dr2xml\_file.xml*) Please note that this protocol might be used in production only with a lot of attention because its activation can generate a non-negligible overhead in terms of storage space (size and inodes) on WORKDIR filesystem as well as in terms of calculation time (there is a specific libGCM flag to modify output file system). For a practical test, please see part 12.2 related to coupled model (specialized section). This functionality is not yet implemented in all configurations.

## INTERMEDIATE

# 9. Output files manipulations

This section provides some exercises to introduce common tools used in the climate/meteorology community to manipulate data. This is not an exhaustive list of tools. The idea here is to perform the same basic output manipulations and let you see which one seems the most suitable for you. Beware, in this simple use case some tools could appear complicated compared to others whereas it could be different for complex analysis ; that's why there is a quick concluding paragraph where we provide additional information and a point of view on the best use. This is only a point of view and everybody has to discuss with people, read docs and test them to conclude for himself.

*Note that we will not discuss Climaf in this section.*

## 9.0 Protocol and environment

### 9.0.1 Protocol

In the following sections you will use several tools/languages to load the daily atmospherical output file produced by LMDZ. Extract the “2m-temperature” field ( $t_{2m}$ ) and save it as a time-serie file. Then we propose to compute a zonal and global weighted mean (using latitude cosine) and finally plot them.

*Note that you could use another variable or output instead.*

We provide you, for each environment, a daily output file from a one month LMDZ simulation for this practical but you can work directly on your own output files.

- **spirit/spiritx:**

```
/projsu/igcmg/TRAINING/MODIPSL_HandsOn_20210129/Data/MyJobTest_19800101_19800130_1D_histday.nc
```

- **Irene:**

```
/ccc/work/cont003/igcmg/igcmg/TRAINING/MODIPSL_HandsOn_2022/Data/MyJobTest_19800101_19800130_1D_histday.nc
```

- **Jean-Zay:**

```
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_20210129/Data/MyJobTest_19800101_19800130_1D_histday.nc
```

Now create a new directory “`MYDIR`” and copy the example file (or your own outputs) from correct path (see above):

```
mkdir MYDIR
cd MYDIR
cp CORRECT_PATH_ABOVE/MyJobTest_19800101_19800130_1D_histday.nc .
```

*Note that in next sections, we will use `$MYDIR` to refer to your new directory containing the output file to analyze. This copy is to avoid potential multi-access during this practical, but in reality you could work directly with the output files.*

### 9.0.1 Environment

Before starting you need to check that the following modules are available (revision could be different in function of the computer): `module list`

```
1) netcdf/4.7.2-mpi      2) nco/4.8.1
3) ferret/7.2           4) netcdf/4.7.2-mpi
5) ncview/2.1.7-mpi     6) cdo/1.9.7.1
7) ncl/6.6.2-mpi        8) python/3.7.5
```

With the standard IPSL environment installed on supercomputers all modules will be available (see [documentation](#)). Otherwise you could add them using the `module load` command as follows:

#### **Jean-Zay**

```
module purge
module load nco
module load cdo
module load ncview
module load ferret
module load ncl
module load python/3.7.5
```

#### **Irene**

```
module purge
module load intel/19 mpi/openmpi/4.0.5
module load hdf5
module load python3/3.7.5
module load nco/4.9.1
module load cdo/1.9.5
module load ferret/7.2
module load ncl_ncarg/6.3.0
```

```
module load ncview/2.1.7
```

### Spirit(x)

```
module purge
module load netcdf4/4.4.1.1-parallel-ifort
module load nco/4.7.x
module load cdo/1.9
module load ferret/7.4.3
module load ncl/6.6.2
module load python/3.6-anaconda50
```

If you have not installed Xarray on Spirit(x) before, you will need it if you want to test this library. Here is the installation through the Conda environment (local installation of modules):

```
conda create -n py36env python=3.6 # Create conda environment with python 3.6
source activate py36env # load environment (the prompt change when you're inside)
conda install xarray matplotlib netcdf4
```

## 9.1 Network Common Data Form (NetCDF) format

In the IPSL models the output format is NetCDF. NetCDF is “*self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data*” (from [Wikipedia](#)). This is a binary format that requires special tools and/or libraries to be used.

When the NetCDF library is installed on a computer, some basic manipulation tools are supplied. This is the case of the `ncdump` command which allows you to see the content of a netCDF file.

Use it with the `-h` option to get only header information, no data:

```
cd $MYDIR
ncdump -h MyJobTest_19800101_19800130_1D_histday.nc
```

**Question 9a:** Look at the file structure, how is it composed ? Explore other variables or components (SBG, MBG, OCE, ICE...) that you already produced. Are they structured in the same way ?

Informations: <https://www.unidata.ucar.edu/software/netcdf/>

## 9.2 NetCDF Operator (NCO)

We are going to use the atmospheric output file produced in the “basic exercise” (section 1) copied in your account in 10.0.1 section.

The general atmospheric file `MyJobTest_19800101_19800130_1D_histday.nc` contains a lot of variables (list is in `lmdz.card`). To avoid manipulating this big file, we'll first create our own time series file for the 2D temperature `t2m` (this is the same process done during a simulation in post-processing jobs).

To extract this variable use `ncks` (one of the CDO tool) as follows:

```
cd $MYDIR
ncks -3 -v t2m MyJobTest_19800101_19800130_1D_histday.nc
t2m_TS.nc
```

**Question 9b:** Check the output file content using `ncdump -h`

Now, to calculate an area-averaged index, you first need to add the *latitude weights* to the file with `ncap2` before computing average with `ncwa` (-O option is to overwrite file):

```
# Add cos(latitude) to balance all grid point contribution
ncap2 -3 -h -O -s "weights=cos(lat*3.1415/180)" t2m_TS.nc
t2m_TS.nc
# Global average
ncwa -3 -h -O -w weights -a lat,lon t2m_TS.nc t2m_glob_mean.nc
# Zonal average
ncwa -3 -h -O -a lon t2m_TS.nc t2m_zon_mean.nc
```

**Question 9c:** add the keyword `time` before each command and note the time elapsed to compare performances with CDO presented in the following section.

**Conclusion:** NCO is a very common set of several tools used through a terminal. It is generally installed on computing centers and frequently updated. As shown in the exercise before it creates a lot of intermediary files if you need to perform a complex analysis but it is optimized for performing some complex analyses quickly and using large files. There is no visualization in NCO.

Informations: <http://nco.sourceforge.net>



## 9.3 Climate Data Operators (CDO)

CDO is a set of tools very useful to manipulate climate data. Its usage is close to NCO (see previous section) with its own operators. The CDO syntax is the following :

```
cdo <operator>,<option> input.nc output.nc
```

Let's start with variable extraction with the `selvar` operator:

```
cd $MYDIR
cdo selvar,t2m MyJobTest_19800101_19800130_1D_histday.nc t2m_TS_CDO.nc
```

**Question 9d:** Check the output file content using `ncdump -h`. You could print information and get basic statistics for each field of a dataset using `cdo info t2m_TS_CDO.nc` (mean is a non-weighted average).

Now perform the same treatment than before: global weighted average with `fldmean` (use directly grid info to find area weights) and zonal one using `zonmean`:

```
# Global average
cdo fldmean t2m_TS_CDO.nc t2m_glob_mean_CDO.nc
# Zonal average
cdo zonmean t2m_TS_CDO.nc t2m_zon_mean_CDO.nc
```

**Question 9e:** add the `time` keyword before each command and note the time elapsed to compare performances with NCO presented in the previous section.

**Conclusion:** CDO is a set of tools, developed by the Max Planck institute, similar to NCO. The syntax is a little bit different but it allows you to do almost the same things. Sometimes it is easier to perform some analysis with CDO, sometimes with NCO. Both could be used and chained. However the memory optimisation seems better with NCO. It also creates temporary files to be cleaned up afterwards and does not offer visualization. The documentation is not so easy to find on the Internet.

**Informations:** <https://code.mpimet.mpg.de/projects/cdo>

## 9.4 NetCDF Visual browser (NCView)

NCView is a very basic NetCDF file visual browser. We propose to use it to show outputs from previous exercises and let you play with its basic interface (need to select the *t2m* variable):

```
# plot global mean
ncview t2m_glob_mean.nc
# show zonal average
ncview t2m_zon_mean.nc
```

Conclusion: It could be useful to quickly check file content and show data (with `>>` you could play data along an axis) ; but it is still very basic and doesn't allow you to perform analysis.

Informations: [http://meteora.ucsd.edu/~pierce/ncview\\_home\\_page.html](http://meteora.ucsd.edu/~pierce/ncview_home_page.html)

## 9.5 Ferret

Open ferret and load the model daily output file (*histday*) or the *t2m time series* file (created with NCO in 9.2) otherwise:

```
cd $MYDIR
ferret # go into ferret app

##### IN FERRET (after "yes?" prompt) #####
USE MyJobTest_19800101_19800130_1D_histday.nc
SAVE/FILE="t2m_TS_FERRET.nc" t2m[d=1]
use t2m_TS_FERRET.nc
show data/f 2 ! show all info in dataset 2 (ie t2m TimeSerie)
```

**Note**: Ferret is not case sensitive so it ignores lower and upper case for commands and variables name.

Now you will compute the zonal mean using `@ave` command to do it (Ferret automatically weighted average using grid properties) and show it with `shade`:

```
shade t2m[y=@ave, d=2]
```

And then plot the global average:

```
plot t2m[x=@ave,y=@ave, d=2]
```

**Conclusion:** It is a very good tool for quick sanity checks. Very easy to load/save data, basic data manipulation (averages, sums) and plot time series and 2D view.

Otherwise syntax is not very friendly (there is no variable but alias), generates bad image quality (need to use PyFerret to solve this), only few docs and not very active developments.

**Informations:** <https://ferret.pmel.noaa.gov/Ferret/>

Short tutorial: <https://ferret.pmel.noaa.gov/Ferret/documentation/ferret-tutorial-script>

## 9.6 NCAR Command Language (NCL)

NCL is an environment developed by NCAR people. It was very popular in the weather and climate community, particularly for the large panel of visualization proposed.

First, you'll start the NCL environment using `ncl` command line (use `ctrl+d` to exit):

```
ncl
```

Then you'll create a `t2m` timeserie from model output file ; using `addfile()` function to load model output, then select the variable to finally create a new output with `"c"` option and write it:

```
df = addfile("MyJobTest_19800101_19800130_1D_histday.nc", "r")
temp = df->t2m ; store t2m in a variable
fout=addfile("t2m_TS_NCL.nc", "c") ; create out file
fout->t2m=temp ; write temp in t2m variable
```

**Note:** You could show quick information about a variable using `printVarSummary` command. For example to look at the temperature info: `printVarSummary(temp)`

**Question 9f:** Quit `ncl` via `ctrl+d` and look inside the new created file using `ncdump -h`

Now load the t2m file just created to compute the weighted global using `wgt_areaave_Wrap` function (`Wrap` is to keep metadata) and then plot it into a `ave.png` file (don't forget to start the ncl program first!):

```
df = addfile("t2m_TS_NCL.nc","r"); read t2m TS
temp = df->t2m ; store t2m in a variable
lat = df->lat ; store lat in a variable
rad = 4.0*atan(1.0)/180.0
clat = cos(lat*rad) ; lat cosine
globav = wgt_areaave_Wrap(temp, clat, 1.0, 0) ; global average

; *** create graphic into ave.png file ***
wks = gsn_open_wks("png","globave") ; send graphics to PNG
file
res = True
res@tiYAxisString= globav@long_name + " (" + globav@units + ")"
res@tiXAxisString= "Time Steps"
res@tiMainString = "Global Weighted Average"
x = ispan(0,dimsizes(globav)-1,1) ; create x-axis
plot = gsn_csm_xy(wks,x,globav,res) ; create plot
```

**Question 9g:** you could have a look at the output in the `globave.png` file using for example `display` command such as `display globave.png`

Now proceed to the zonal mean using `dim_avg_n_Wrap` which averaged the rightmost dimension (so you need to permute them if it is not the lon):

```
df = addfile("t2m_TS_NCL.nc","r"); read t2m TS
temp = df->t2m ; store t2m in a variable
zave = dim_avg_n_Wrap(temp,2) ; zonal average (=dim 2)

; *** create graphic into ave.png file ***
wks = gsn_open_wks("png","zonal") ; send graphics to PNG file
res = True ; plot mods desired
res@tiMainString = "Hovmoller" ; title
res@tmXBLLabelStride = 2 ; tick mark label
stride
res@tiYAxisString = "Time" ; y axis title
res@tiXAxisString = "Lat" ; x axis title

res@cnFillOn = True ; colour on
res@lbLabelStride = 2 ; every other label
res@lbOrientation = "Vertical" ; vertical label bar
res@cnLinesOn = False ; turn off contour
lines
res@cnFillPalette = "gui_default" ; set colour map
```

```
res@cnLevelSpacingF = 1 ; contour spacing
plot = gsn_csm_time_lat(wks, zave, res ) ; plot zonal ave
```

**Question 9h:** you could have a look at the output in the *zonal.png* file using for example `display` command such as `display zonal.png`

**Conclusion:** NCL is a very powerful tool with good documentation and community. For about 3 years, the developers announced that the environment won't be updated but all the functionalities will become Python libraries : [PyNIO](#) for data manipulation and [PyNGL](#) for the graphical part. The project is called the Geosciences Community Analysis Toolkit (GeoCAT), and now has a specific [website](#). So we advise you to directly use the Python version.

**Informations:** <http://www.ncl.ucar.edu> and <https://geocat.ucar.edu> (Python version)

## 9.7 Python

Python is a very popular general developing language, and a lot of libraries are available to analyze climate data.

First you need to load the python3 module (as previously explained in 10.0.1 section) and then start `python3`. If you work on spirit(x) don't forget to activate the Conda environment (`source activate py36env`).

### 9.7.1 [NetCDF4 / Numpy](#)

First, start the Python environment using `python3` command line (use `ctrl+d` to exit):

```
python3
```

Read NetCDF file, extract *t2m* variable and write its time series:

```
from netCDF4 import Dataset, num2date, default_fillvals
import numpy as np
import matplotlib.pyplot as plt

# load dataset
fnc=Dataset("MyJobTest_19800101_19800130_1D_histday.nc",
mode='r')
```

```

# extract t2m and dimension variables
temp = fnc.variables['t2m']
time = fnc.variables['time_counter']
lati = fnc.variables['lat']
long = fnc.variables['lon']

# Create output file
fout = Dataset("t2m_TS_NC.nc", mode='w')
# create dimensions
fout.createDimension('time_counter', None)
fout_tdim = fout.createVariable('time_counter', time.dtype,
('time_counter',))
fout.variables['time_counter'][:] = time[:]
for ncatr in time.ncattrs(): # copy metadata
    fout_tdim.setncattr(ncatr, time.getncattr(ncatr))

fout.createDimension('lat', len(lati))
fout_latdim = fout.createVariable('lat', lati.dtype, ('lat',))
fout.variables['lat'][:] = lati[:]
for ncatr in lati.ncattrs():
    fout_latdim.setncattr(ncatr, lati.getncattr(ncatr))

fout.createDimension('lon', len(long))
fout_londim = fout.createVariable('lon', long.dtype, ('lon',))
fout.variables['lon'][:] = long[:]
for ncatr in long.ncattrs():
    fout_londim.setncattr(ncatr, long.getncattr(ncatr))

# create variables
temp_var = fout.createVariable('t2m', temp.dtype,
('time_counter', 'lat', 'lon'), fill_value=True)
for ncatr in temp.ncattrs():
    # patch for some version of python
    if(ncatr == '_FillValue'):
        continue
    temp_var.setncattr(ncatr, temp.getncattr(ncatr))
fout.variables['t2m'][:] = temp[:]

fout.close() # close file

```

Now load the time series file and compute zonal and global averages using `numpy`:

```

ftemp=Dataset("t2m_TS_NC.nc", mode='r')
temp = ftemp.variables['t2m']
lat = ftemp.variables['lat']
wgt = np.cos(np.deg2rad(lat)) # lat cosine
zave= np.average(temp[:].data, axis = 2) # zonal average

```

```
gave= np.average(zave, axis = 1, weights = wgt) # global weighted
```

And plot results using matplotlib library:

```
plt.show(block=False) # let you continue to write
plt.plot(gave)
plt.figure() # create new figure
plt.contourf(zave, cmap=plt.cm.YlOrBr)
plt.colorbar() # show colorbar
plt.show()
```

### 9.7.2 XArray

XArray library is a Python package that makes working with labeled multi-dimensional arrays simple. It is based on Numpy and Pandas and uses Matplotlib by default to plot data.

Let's start with the t2m TS file creation in python (don't forget to start Python using `python` command). If you work on spirit(x) don't forget to activate the Conda environment (source activate py36env):

```
import xarray as xr
import numpy as np
import matplotlib.pyplot as plt
ds = xr.open_dataset("MyJobTest_19800101_19800130_1D_histday.nc")
temp = ds.t2m # store t2m in a variable
temp.to_netcdf("t2m_TS_XR.nc") # write temp in t2m variable
```

Now compute zonal and global averages (need to first compute zonal):

```
dst = xr.open_dataset("t2m_TS_XR.nc")
temp=dst.t2m
wgt = np.cos(np.deg2rad(dst.lat)) # lat cosine
zave= temp.mean(dim="lon") # zonal average
gave=(zave*wgt).sum(dim=('lat'))/wgt.sum(dim=('lat'))#glob ave weighted
```

And plot results using matplotlib library:

```
plt.show(block=False) # let you continue to write
```

```
plt.plot(gave)
plt.figure()          # create new figure
plt.contourf(zave, cmap=plt.cm.YlOrBr)
plt.colorbar()       # show colorbar
plt.show()
```

**Note:** when computing averages with XArray internal functions, the metadata will be kept. You could see it if you try to print variables `print(zave)`.

#### Conclusion:

- NetCDF is the basic library which allows you to work at a very low level in the same way that other environments based on it. It is powerful, but it requires a lot of explicit information (in particular to create dimensions and metadata) which could scare users.
- XArray, in another way, adds a lot of very comfortable simplicity to manipulate netCDF files and to manage metadata. It is powerful too and allows you to convert data in other numpy types to use with other libraries but need a bit of learning.

In a general way, Python seems to become the reference language for data analysis in climate or other fields through the impressive amount of libraries available (maybe too much) and each user gets their favorite's ones.

Informations: NetCDF4 - <https://unidata.github.io/netcdf4-python/netCDF4/index.html>  
XArray - <http://xarray.pydata.org>



## SPECIALISED

# 10. Install and run NEMO-PISCES

This exercise on NEMO-PISCES is divided into 2 parts. Part 1 presents the basic steps for running and installing NEMO-PISCES, and Part 2 provides a more in-depth use of a NEMO-PISCES configuration.

## 10.1 Run a 1 month online experiment of NEMO-PISCES

In this exercise, we will first perform a 1 month simulation of the coupled ocean-biogeochemical model NEMO-PISCES, using 30 MPI processes for NEMO and 1 MPI process for XIOS.

Download modipsl as before and then install the NEMO\_v6.5 configuration:

```
mkdir $WORK/NEMO_STD ; cd $WORK/NEMO_STD
svn co https://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl

cd modipsl/util
./model NEMO_v6.5
```

Compile the ORCA2\_ICE\_PISCES configuration :

If you are working on **Jean-Zay/IDRIS**, don't forget to log in to the preprocessing front-end (otherwise the compilation of XIOS will not work) :

```
ssh jean-zay-pp
```

```
cd ../config/NEMO_v6
./compile_nemo.sh &
```

If you are working on **Jean-Zay/IDRIS**, don't forget to log out from the preprocessing front-end once the NEMO code is compiled:

```
exit
```

Create your first experiment with NEMO:

```
cp EXPERIMENTS/ORCA2_ICE_PISCES/core/clim/config.card .
```

Now set up the `config.card` to do the simulation. You can see that for the ORCA2\_ICE\_PISCES configuration, there are 3 components: OCE for ocean, ICE for sea ice and MBG for PISCES.

Modify in `config.card` the following:

```
vi config.card  
JobName=OR2Si3P1 ; SpaceName=TEST ; DateEnd=1948-01-31 ; PeriodLength=1M
```

Create the experiment directory:

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

```
cd OR2Si3P1
```

**Question 10a:** Explore the `COMP/opa9.card` and `COMP/pisces.card` to see the input files needed for OPA and PISCES.

**Question 10b:** Explore in `PARAM/NAMELIST/ORCA2` the `namelist_core_clim_cfg` file to see some parameters of the run.

**Question 10c:** Explore in `PARAM/XML/file_def_nemo*` the files where the output fields are managed for OPA, SI3, and PISCES, respectively.

Submit the job as usual:

```
sbatch Job_OR2Si3P1 / ccc_msub Job_OR2Si3P1
```

**Question 10d:** Explore the `Script_Output` and `run.card` files in the submit directory.

**Question 10e:** Explore the output directories (`OCE/Output`, `ICE/Output`, `MBG/Output`) where the output files are stored.

Continue the simulation for one month.

## 10.2 Run a 1-year offline experiment of NEMO-PISCES

In this 2nd exercise, we will perform a 1 year long simulation of the coupled ocean-biogeochemical model NEMO-PISCES in an *offline* mode (ORCA2\_OFF\_PISCES), using 30 MPI processes for NEMO and 1 MPI process for XIOS. Here, only the biogeochemical fields are computed, NEMO outputs are used to force the dynamical state of the ocean. This allows the exploration of specific biogeochemical features with lower computational costs. We will see how to create 5 days output files as well as the good practice to modify the pisces parameters if needed.

Compile the ORCA2\_OFF\_PISCES configuration:

```
cd $WORK/NEMO_STD/modips1/config/NEMO_v6/  
./compile_nemo.sh OFFLINE &
```

Create the job for NEMO-PISCES offline:

```
cp EXPERIMENTS/ORCA2_OFF_PISCES/clin/config.card .
```

Set up the config.card to do the simulation. You can see that for the configuration ORCA2\_OFF\_PISCES, there is only 1 component : MBG for PISCES.

Modify in config.card the following lines:

```
vi config.card  
JobName=OR2OFFPIS1 ; SpaceName=TEST ; DateEnd=1850-12-31
```

Create the experiment directory:

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

**Question 10f:** Explore the `COMP/pisces.card` to see the input files from NEMO needed for PISCES

**Question 10g:** Explore in `PARAM/NAMELIST/ORCA2` the `namelist_offline_clim_cfg` to see the parameters for the run

Submit the job as usual:

```
cd OR2OFFPIS1
sbatch Job_OR2OFFPIS1 / ccc_msub Job_OR2OFFPIS1
```

**Question 10h:** Explore the output directories where the output files are stored (MBG/Output).

We will now create a new NEMO-PISCES *offline* configuration. We will modify the `config.card`, `pisces.card`, and `file_def_nemo-top.xml` files to get output of some fields at a frequency of 5 days. We will also see how to modify the parameters in the `namelist_pisces_cfg` file.

Create a new NEMO-PISCES offline configuration:

```
cd $WORK/NEMO_STD/modipsl/config/NEMO_v6/
cp EXPERIMENTS/ORCA2_OFF_PISCES/clim/config.card .
```

Modify in `config.card` the following:

```
vi config.card
JobName=OR2OFFPIS2 ; SpaceName=TEST ; DateEnd=1850-12-31 ;
[MBG]
WriteFrequency="5D 1M 1Y"
```

Create the experiment directory:

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

Edit the `pisces.card` to add 5 days outputs for `*.ptrcT` file:

```
cd OR2OFFPIS2/
vi COMP/pisces.card
```

Add the following line in the `[OutputFiles]` list of the `pisces.card` file:

```
...
(${config_UserChoices_JobName}_5d_ptrc_T.nc,${R_OUT_MBG_O_D}/${PREFIX}_5D_
ptrc_T.nc , NONE ) , \
...
```

Add in the `PARAM/XML/file_def_nemo-top.xml` the NO3, PO4, Si, Fer, DCHL, NCHL variables in the specific group of 5d files.

```
vi PARAM/XML/file_def_nemo-top.xml
```

Set in the `<!-- 5d files -->` section below

```
<file_group id="5d_pis" output_freq="5d" output_level="_AUTO_"
enabled="_AUTO_" enabled=".FALSE." > <!-- 5d files -->
```

the enabled parameter to “`_AUTO_`” (in bold) to allow to write the

```
<file_group id="5d_pis" output_freq="5d" output_level="_AUTO_"
enabled="_AUTO_" enabled="_AUTO_" > <!-- 5d files -->
```

Then add the following variables (lines in bold) below the list of bioscalar fields:

```
<field field_ref="pfertot" name="pfertot" unit="nmolFe"
operation="instant" level="2" > pfertot * 1e9 </field>
</file>
...
<file id="file41" name_suffix="ptrc_T" description="pisces sms
variables" >
<field field_ref="e3t" name="E3T" long_name="T-cell thickness" />
<field field_ref="PO4" name="PO4" operation="average"
freq_op="5d" level="2" > @PO4_e3t / @e3t </field>
<field field_ref="NO3" name="NO3" operation="average"
freq_op="5d" level="2" > @NO3_e3t / @e3t </field>
<field field_ref="Si" name="Si" operation="average"
freq_op="5d" level="2" > @Si_e3t / @e3t </field>
<field field_ref="NCHL" name="NCHL" operation="average"
freq_op="5d" level="2" > @NCHL_e3t / @e3t </field>
<field field_ref="DCHL" name="DCHL" operation="average"
freq_op="5d" level="2" > @DCHL_e3t / @e3t </field>
<field field_ref="Fer" name="Fer" operation="average"
freq_op="5d" level="2" > @Fer_e3t / @e3t </field>
</file>
...
</file_group>
```

We have finished to set up the configuration to get biogeochemical fields at an output frequency of 5 days for the `*ptrc_T` file.

Now we will see how to modify the namelist parameters of PISCES. For instance, we will change the values of the Photosynthesis-Irradiance ratio for both phytoplankton and will explore the impacts for surface chlorophyll, NO<sub>3</sub>, Si and Fe.

Open the `pisces.card`

```
vi COMP/pisces.card
```

**Question 10i :** Find where the reference namelist of PISCES is stored. Open the file.

```
vi ../../../../modeles/NEMOGCM/CONFIG/SHARED/namelist_pisces_ref
```

All the parameters of PISCES are listed here. This file should not be modified if you want/need to change some PISCES parameters.

**Question 10j :** Explore the `namelist_pisces_ref`

Copy the two parameters for the Photosynthesis-Irradiance ratio (P-I slope) in the `&namp4zprod` namelist section from the `namelist_pisces_ref` in the `namelist_pisces_cfg` of your configuration.

Copy the lines below from the `namelist_pisces_ref`

```
pislopen   =  2.          ! P-I slope
pisloped   =  2.          ! P-I slope for diatoms
```

Paste the copied line in the `namelist_pisces_cfg` in the section of `&namp4zprod`:

```
vi PARAM/NAMELIST/namelist_pisces_cfg
```

Set the `pislopen/pisloped` values to 3. in the `namelist_pisces_cfg`

```
pislopen   =  3.          ! P-I slope
pisloped   =  3.          ! P-I slope for diatoms
```

Submit the job:

```
sbatch Job_OR2OFFPIS2 / ccc_msub Job_OR2OFFPIS2
```

**Question 10k :** Explore the output directories (`MBG/Output`) where the output files are stored to check whether the 5d `*ptrc_T` file has been created.

**Question 10l :** Compare the annual output files of the 2 *offline* configurations (OR2OFFPIS1, OR2OFFPIS2) and explore the differences on surface CHL, NO3, Si and Fe.

## SPECIALISED

# 11. Ensembles

Note that this section about **Ensemble** is only for users who know what an Ensemble is. If you don't, that probably means that you won't need to do ensemble runs.

Here **ensemble** defines a set of several simulations using exactly the same configuration but differing only by changing the initial conditions. LibIGCM could actually help you to create easily two different types of ensembles using a specific card file :

- 1- Ensemble with random perturbations of initial SST
- 2- Ensemble choosing different starting dates from previous simulations

To get more details, please show the dedicated documentation:

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Setup/Ensemble](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Setup/Ensemble)

*We give here an example of config.card and ensemble.card to generate 2 members, starting in 1851 from a piControl simulation of 1850. A white noise (of 0.1K) is applied to the SST of the CPL restart file.*

**Caution:** THIS TOPIC was checked on TGCC Irene machines but could be done on Jean-Zay.

**To configure** an ensemble of simulations with slightly different perturbed initial conditions it is possible to use the `-e` option in `ins_job` script.

To use this option a new configuration card file **ensemble.card** is needed in the current directory.

There are two types of possible ensembles you could create directly with *libIGCM* :

- [Ens\_PERTURB]: configures a set of periodic (annual) simulations from a *Start date* to an *End date*, with a defined number of members (ie random perturbations of the initial state)
- [Ens\_DATE]: configures a set of simulations using several restart dates from one or several simulations

## 11.1 Install and configure ensemble

In this practical we'll use the IPSLCM6.2.2 coupled model. To install it, please follow instructions at the beginning of “[exercise 12: Coupled model](#)” using `./model IPSLCM6.2.2`.

## 11.2 Configure the ensemble

1. Now you will start to create your new ensemble experiment in your model from *decadal* Template:

```
cd modipsl/config/IPSLCM6
cp EXPERIMENTS/IPSLCM/decadal/config.card .
cp EXPERIMENTS/IPSLCM/decadal/ensemble.card .
```

2. Now you will configure an example of a [Ens\_PERTURB] ensemble containing 2 experiments running from January 1st 1851 to December 31th 1851 restarting from the piControl simulation “*CM61-pre-pi-01*”. libGCM will automatically use the previous day as restart (ie 1850-31-12) and perturb it randomly.

→ Start editing `config.card` to common part of all members:

```
vi config.card # Modify using these following lines
JobName=ENS
SpaceName=TEST
DateBegin=1851-01-01
DateEnd=1851-12-31
PeriodLength=1Y

EnvFile=${SUBMIT_DIR}/../../../../ARCH/arch.env # be careful to the number of “/..”
```

### **Important:**

Check that in your `config.card` there is a “[Ensemble]” section as follows (be careful you need to add `EnsembleMergeSave`). Note that `EnsembleRun` option doesn't exist in recent version of libGCM:



```

#=====
[Ensemble]
#D- Ensemble run ? 'y' or 'n'
#D- If 'y', fill in ensemble.card !!
EnsembleRun=y
EnsembleName=
EnsembleDate=
EnsembleType=
EnsembleMergeSave=

```

→ Now configure ensemble properties in `ensemble.card` file as follow (**for Irene users**):

```

vi ensemble.card # (for IRENE only)
# write this following lines in [UserChoices] section
NAME=ENS # Ensemble name
MEMBER=2 # Nb of SST perturbations for each restart
LENGTH=1Y # Simulation length of all members

BEGIN_INIT=18510101 # start date of the first simulation
END_INIT=18511231 # start date of the last simulation
PERIODICITY=1Y # timestep between each simulation

PERTURB_BIN=(AddNoise, CPL, sstoc, O_SSTSST, 0.1)

INITFROM=CM61-pre-pi-01 # Restart simulation name
INITPATH==${R_IN}/RESTART/IPSLCM6/PROD/piControl-spinup

```

**For Jean-Zay users**, use following parameters for this exercise in `ensemble.card`:

```

vi ensemble.card # write this following lines (for Jean-Zay only)
[Ens_PERTURB]
active=y
NAME=ENS
MEMBER=2
LENGTH=1Y

BEGIN_INIT=18510101
END_INIT=18511231
PERIODICITY=1Y

PERTURB_BIN=(AddNoise, CPL, sstoc, O_SSTSST, 0.1)

```

```
INITFROM=CM61-pi-valid.02
INITPATH=$STORE/./././rech/psl/commun/IGCM_OUT/IPSLCM6/DEVT/piControl
```

### !! For Jean-Zay users !!

Check and modify in `ins_job` script that the `workdir` variable is the correct one (ie `$WORK`):

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

# Around line 625
RUN_DIR="${WORK}/ENSEMBLE_TMP"
```

3. Create your ensemble of simulation ENS (if you encounter an error message, have a look to the *TIP* below):

```
../../libIGCM/ins_job -e # answer all questions and ask for 4600s of cputime
```

**NOTE:** for ensemble, **JobName** param in `config.card` will define the name of the global directory containing all simulations, scripts and config files. We advise you to use the same name as in `ensemble.card` **NAME** option as proposed in this example.

**TIP:** Only if you have a problem during ensemble creation, you could check:

```
vi $CCCWORK/MYFIRSTTEST/modipsl/libIGCM/ins_job

If you've got an error like :
"mkdir: cannot create directory '/ENSEMBLE_TMP': Permission
denied"
Check in libIGCM/ins_job value of RUN_DIR:

##### FOR IRENE #####
RUN_DIR="${CCCWORK}/ENSEMBLE_TMP"

##### FOR JEAN-ZAY #####
RUN_DIR="${WORK}/ENSEMBLE_TMP"

(try RUN_DIR="${TMPDIR}/ENSEMBLE_TMP"
```

4. Description of the ensemble organisation

Now that the ensemble configuration is created you could have a look at the organisation before starting ensemble simulations.

First the general directory `ENS` created corresponds to the `config.card` `JobName` param.

Inside the new directory, you'll find :

- the previous `config.card` and `ensemble.card`
- `Job_$ENS` and `run.card.init` files (not used)
- The parameters directories common for all simulations `PARAM`, `DRIVER`, `POST` and `COMP`
- `Qsub.*.sh` and `Qclean.*.sh` scripts allowing to submit or cleaning all members all together
- A `{EnsembleName}{StartDate}` directory containing all members for a single starting date.

#### Informations:

- In `[Ens_DATE]` ensemble, the ensemble name will be only the `NAME` option filled in the `ensemble.card`.
- If you use a `CMIP6` simulation with a member name in `config.card` `[UserChoices]` section like `Member= r1i1p1f1`, the `r` number will be incremented automatically. Don't forget to add `_CMIP6` at the end of `ExpType=` value to be considered as a `CMIP` experiment.

#### 5. Start the ensemble

```
cd ENS #
vi Qsub.ENS1851.sh # script used to launch all sims (only to understand the idea)

chmod 755 Qsub.ENS1851.sh # set to executable
sh Qsub.ENS1851.sh # submit all members

This shell script launches 2 jobs that are running 2 starting
date experiences.

To see if Job are running you can do:

ccc_mstat -u yourusername # for Irene
squeue -u yourusername # for Jean-Zay
```

**On Irene**, this example generates 2 members of simulation starting in 1851 (`BeginDate` in `config.card`), from year 1850 of the restart simulation:

```
${R_IN}/RESTART/IPSLCM6/PROD/piControl/CM61-LR-pi-03
```

**On Jean-Zay**, this example generates 2 members of simulation starting in 1851 (`BeginDate` in `config.card`), from year 1850 of the restart simulation:

```
$STORE/../../../../rech/psl/commun/IGCM_OUT/IPSLCM6/DEVT/piControl/CM
61-pi-valid.02
```

White Noise is applied to SST; you can check perturbed variables here:

**On Irene:**

```
$CCCWORKDIR/IGCM_IN/IPSLCM6/JobNameYEAR/JobNameYEAR-0$member/CPL/R  
estart
```

**On Jean-Zay:**

```
$WORK/IGCM_IN/IPSLCM6/JobNameYEAR/JobNameYEAR-0$member/CPL/Restart
```

In this example JobNameYEAR is “ENS1851”, and subdirectories are ENS1851-01 and ENS1851-02.

The submission directory has been created with the same name as the JobNameYEAR. In this directory there are as many directories as the number of members. Look at JobNameYEAR directory and explore subdirectories.

In JobNameYEAR there is a shell script that can be launched (chmod 755 Qsub.ENS1851.sh; sh Qsub.ENS1851.sh) . With this script all members of all years will be launched.

For more information see documentation :

[https://forge.ipsl.jussieu.fr/igcmg\\_doc/wiki/Doc/Setup/Ensemble](https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Setup/Ensemble)

## SPECIALISED

# 12. Coupled model

The aim of this part is to apply **what you have learned in part 2**: performing extraction, compilation and run of the whole coupled (ocean-atmosphere) model configuration IPSLCM6.2.2. So you have to :

- Extract modipsl
- Extract IPSLCM6.2.2 configuration
- Compile (./compile\_ipslcm6.sh)

If you are working on **Jean-Zay/IDRIS**, don't forget to log in to the preprocessing front-end (otherwise the compilation of XIOS will not work):

```
ssh jean-zay-pp
```

### For January 2023 training course at IDRIS:

Launch the compilation as explained above.

**In case that the compilation duration is too long**, you need to do the 2 following points:  
1- copy the executable in your bin directory:

```
cp  
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2023/IPSLCM6.2  
.2/jean-zay/bin/* $WORK/MYFIRSTTEST/modipsl/bin/.
```

2- create the environment file (it will be used by the simulation to install the environment):

```
cd modipsl/config/IPSLCM6/ARCH  
ln -s arch-X64_JEANZAY.env arch.env
```

If you are working on **Jean-Zay/IDRIS**, don't forget to log out from the preprocessing front-end

```
exit
```

## 12.1 piControl experiment

- Set up a 5 days piControl experiment (IPSLCM/piControl\_TEST experiment)
  - a. SpaceName=TEST
  - b. PackFrequency=NONE
  - c. TimeSeriesFrequency=NONE
  - d. SeasonalFrequency=NONE
- Launch the simulation
- Check output files of the simulation

## 12.2 piControl experiment with CMIP6 workflow

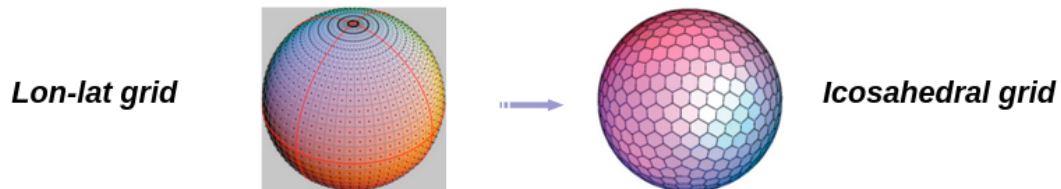
General information on the workflow functionality is available in part 4.4.

If you are working on **Jean-Zay/IDRIS**, you have to modify the script python used to update some XIOS xml files. These xml files allow the activation of CMIP6 workflow. You have to specify you want to use python2 instead of python3 by replacing python by python2 at the first line of the following script  
`.../modipsl/libIGCM/libIGCM_post/xios_parser.py`  
**`#!/usr/bin/env python2`**  
 instead of  
**`#!/usr/bin/env python`**

- Set up a 5 days piControl experiment (IPSLCM/piControl\_TEST experiment)
  - a. SpaceName=TEST
  - b. PackFrequency =NONE
  - c. TimeSeriesFrequency=NONE
  - d. SeasonalFrequency=NONE
  - e. **dr2xmlIPSL=TRUE** (to be added in Post section)
- Launch the simulation
- Check output files of the simulation, especially CMIP6 workflow outputs on  
`$WORKDIR/.../IGCM_OUT/IPSLCM6/TEST/piControl/Name_of_simulation/CMIP6`

## SPECIALISED

### 13. ICOLMDZOR configuration



The aim of this part is to apply to the ICOLMDZOR configuration **what you have learned in part 2**: performing extraction, compilation and run a simulation.

ICOLMDZOR configuration is the offline atmosphere-surface configuration that uses DYNAMICO as dynamical core. With this configuration we can run LMDZ physics and ORCHIDEE land surface on an icosahedral grid or / and a regular grid. The choice is made at the compilation.

The model configuration to use is ICOLMDZOR\_v7\_work. So you have to :

- Extract modipsl
- Extract ICOLMDZOR\_v7\_TP configuration

**Question 13a:** use “-h” option to know all options of the compilation script `compile_icolmdzor.sh`. Which command will you launch to create executables for the regular grid and the icosahedral grid ? Which command will you launch to create only the executable for the icosahedral grid ?

If you are working on **Jean-Zay/IDRIS**, don't forget to log in to the preprocessing front-end (otherwise the compilation of XIOS will not work):

```
ssh jean-zay-pp
```

- Compile icosahedral and regular grid

**For January 2023 training course at IDRIS:**

Launch the compilation as explained above.

**In case that the compilation duration is too long**, you need to done the 2 following points :

1- copy the executable in your bin directory:

```
cp
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2023/ICOLMDZOR
/jean-zay/bin/* $WORK/MYFIRSTTEST/modips1/bin/.
```

2- create the environment file (it will be used by the simulation to install the environment):

```
cd modips1/config/ICOLMDZOR_v7/ARCH
ln -s arch-X64_JEANZAY.env arch.env
```

If you are working on **Jean-Zay/IDRIS**, don't forget to log out from the preprocessing front-end

*exit*

- Set up a 5 days clim (noleap calendar) experiment with the icosahedral grid (ICOLMDZOR/clim\_noleap experiment)
  - a. SpaceName=TEST
  - b. PackFrequency=NONE
  - c. TimeSeriesFrequency=NONE
  - d. SeasonalFrequency=NONE
- Launch the simulation
- Do the same for the regular experiment (LMDZOR/clim\_pdControl experiment - Modify number of OMP thread if you are running on Jean Zay (as in 3.1))

**Question 13b:** Check output files of the two simulations. Compare them, using for example the 5 days average of the temperature at surface.

This exercise allows us to experiment the fact that working with the icosahedral grid is not more complex than with the regular grid. The results are usable as is. The configuration also allows comparison for validation with the regular grid using exactly the same code sources.