

Getting started with the IPSL tools: modipsl and libIGCM

Exercises for training course

Revised for 2024 training sessions.

A. Caubel, 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. 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 IPSLCM7 configuration. But everything you learn will be usable with all model configurations (IPSLCM6, LMDZOR_v6.2, LMDZORINCA, etc...).

You will find several questions in all this document. Answers are in a separate file that you can find in the [training part](#) of IGCMG online documentation.

Table of content

0. Introduction	5
0.1 For TGCC users	5
0.2 Essentials notes on today's training	5
0.3 How to correctly install your environment?	5
0.4 Subscribe to the platform-users mailing list	7
1. Check your quota	8
1.1 For IDRIS	8
1.2 For TGCC	8
1.3 For MESO-IPSL/Spirit and Spiritx	9
1.4 For LSCE/Obelix	9
2. Installing and compiling	10
2.0 Install modipsl	10
2.1 Extract sub configuration ICOLMDZOR from main IPSLCM7 configuration	12
2.2 Compile with the resolution 144x142x79	15
3. Basic simulations	18
3.1 Create the first experiment directory	18
3.2 Define and launch your first simulation of 1 day	20
3.3 How to clean up and relaunch (if needed)	27
3.4 Continue the simulation 4 more days	28
3.5 Create another simulation with pack	32
3.6 Use different forcing files	33
3.7 CREATE_clim and CREATE_amip: Experiments to create initial state files and boundary conditions for LMDZ	34
3.8 Summary on how to extract, compile and launch a simulation	36
4. Debug	38
4.0 How can you analyse the Job Output: Script_Output?	38
4.1 Debug: setup error	38
4.2 Debug: error during the simulation	39
4.3 Compilation in debug mode	40
4.4 Use of RUN_DIR directory to debug without libIGCM infrastructure	45
5. Post_processing : how to quickly visualise the main diagnostics of a simulation	47
5.1 Launch 2 years with default Time Series and default C-ESM-EP atlas	47
5.2 Add variables to Time Series and relaunch with the TimeSeries_Checker.job	51
6. Monitoring and Inter-monitoring	53
6.1 Monitoring	53
6.2 Inter-monitoring	53
6.2.1 from web interface tool "Inter-monitoring application"	53
6.2.2 from web interface tool "simu finder application"	54
7. How to REDO part of a simulation	55
7.1 Launch a 6 days simulation of LMDZOR experiment	55

7.2 Remove daily output file for ATM component of the days 3 and 4 (i.e. 1980-01-03/04)	57
7.3 Apply the method to redo days 3 and 4 of the simulation (to recover missing output file)	57
8. Modify output using XIOS	59
8.1 Create a new output file for ORCHIDEE	59
8.2 Enable a new output file in LMDZ	61
8.3 XIOS in other models	63
8.4 XIOS and CMIP workflow	63
9. Output files manipulations	64
9.0 Protocol and environment	64
9.0.1 Protocol	64
9.0.1 Environment	65
9.1 Network Common Data Form (NetCDF) format	66
9.2 NetCDF Operator (NCO)	67
9.3 Climate Data Operators (CDO)	68
9.4 NetCDF Visual browser (NCView)	69
9.5 Ferret	69
9.6 NCAR Command Language (NCL)	70
9.7 Python	72
9.7.1 NetCDF4 / Numpy	72
9.7.2 XArray	74
10. Install and run NEMO-PISCES	76
10.1 Run a 1 month online experiment of NEMO-PISCES	76
10.2 Run a 1-year offline experiment of NEMO-PISCES	78
11. Ensembles	82
11.1 Install and configure ensemble	82
11.2 Configure the ensemble	83
12. Coupled model	88
13. ICOLMDZOR configuration	91

BEGINNER

0. Introduction

0.1 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. This message can be sent during the training even if we are using JeanZay for this specific day. Don't forget to specify your login in this email.

If you are a TGCC user, don't wait to do it.

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/LSCÉ and spirit or spiritx at meso-IPSL**, 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 2024 training course at IDRIS"**.

0.3 How to correctly install your environment?

Before working with `modipsl` and `libIGCM` on IDRIS or TGCC machines, you need to install a specific 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

For 2024 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/paste 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
```

Besides, take some time to read the file system information for the machine you are using:

→ For IDRIS

https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters/IDRIS#Thingstoknowaboutfilesystems

→ For TGCC

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

→ Working on Obelix/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...). If you are using another machine, replace them with the machine-specific environment variables. For example use \$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).

→ Install your environment if it's not already done on your account.

0.4 Subscribe to the platform-users mailing list

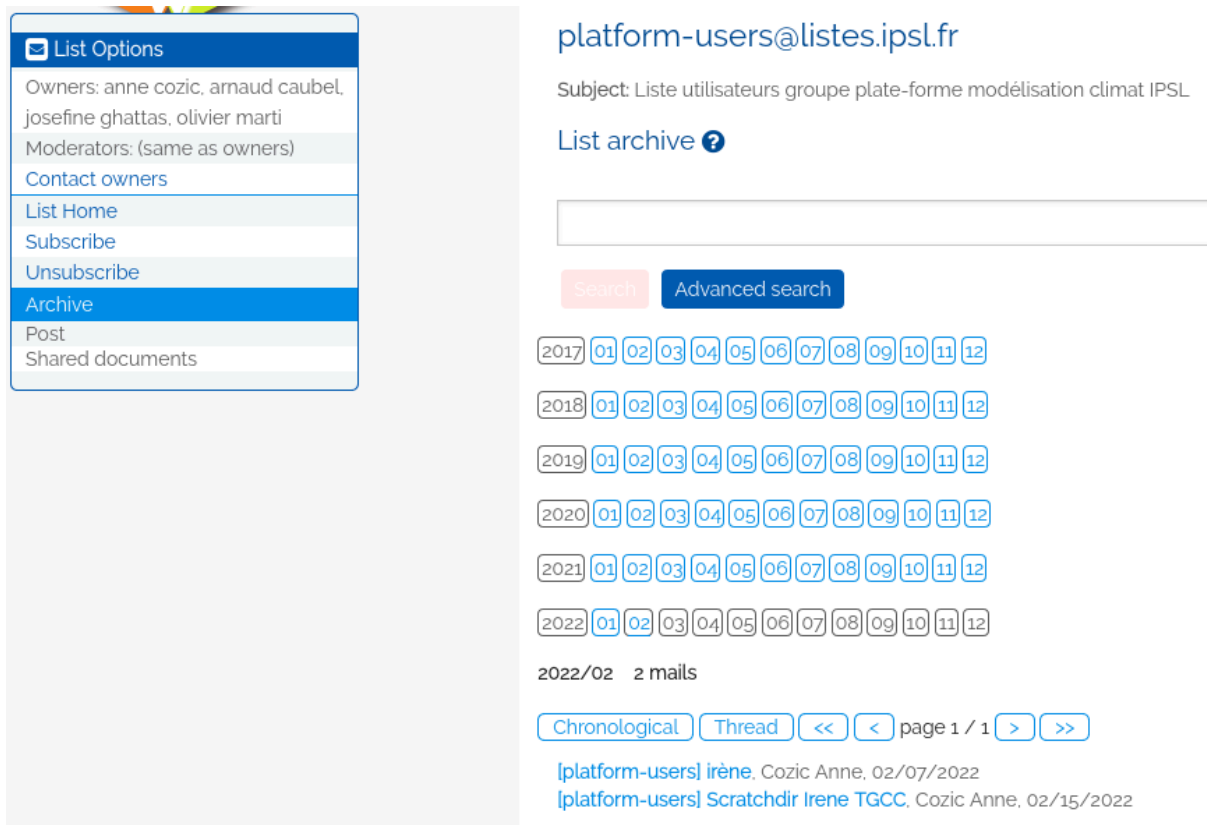
Before working with `modipsl`, `libIGCM` and IPSL models, you should subscribe to the **platform-users mailing list**. Do this by following the link:

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

The purpose of this list is to share information about IPSL tools 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

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



The screenshot shows the web interface for the mailing list `platform-users@listes.ipsl.fr`. On the left is a sidebar menu with options: List Options (selected), Owners: anne cozic, arnaud caubel, josefine ghattas, olivier marti, Moderators: (same as owners), Contact owners, List Home, Subscribe, Unsubscribe, Archive (highlighted), Post, and Shared documents. The main content area displays the list name, subject 'Liste utilisateurs groupe plate-forme modélisation climat IPSL', and a 'List archive' link with a help icon. Below is a search bar and buttons for 'Search' and 'Advanced search'. The archive is organized by year from 2017 to 2022, with each year having 12 numbered links (01-12). The current view is for February 2022, showing '2 mails'. Navigation controls include 'Chronological', 'Thread', and page navigation buttons (page 1 / 1). Two email entries are listed: '[platform-users] irène. Cozic Anne, 02/07/2022' and '[platform-users] Scratchdir Irene TGCC. Cozic Anne, 02/15/2022'.

BEGINNER

1. Check your quota

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

Do the exercises below on the computing centre 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 MESO-IPSL/Spirit and Spiritx

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

→ Use the `quotas` command to check the quota

Warning: note that on Spirit, there is a big problem on `/data`. Do not install the model at this disk space. Install it in your home or in `scratchu`.

If you do this training on Spirit, you can install the model in `/scratchu/$username`.

If you do this training on SpiritX, you can install the model in `/scratchx/$username`.

1.4 For LSCE/Obelix

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

→ 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/$username` 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` file or change it in `modips1/libIGCM/libIGCM_sys_obelix.ksh`.

If you do this training on Obelix, you can install the model in `/home/scratch01/$username`.

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**. At spirit(x) and obelix choose an appropriate folder, see also the introduction.

```
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
```

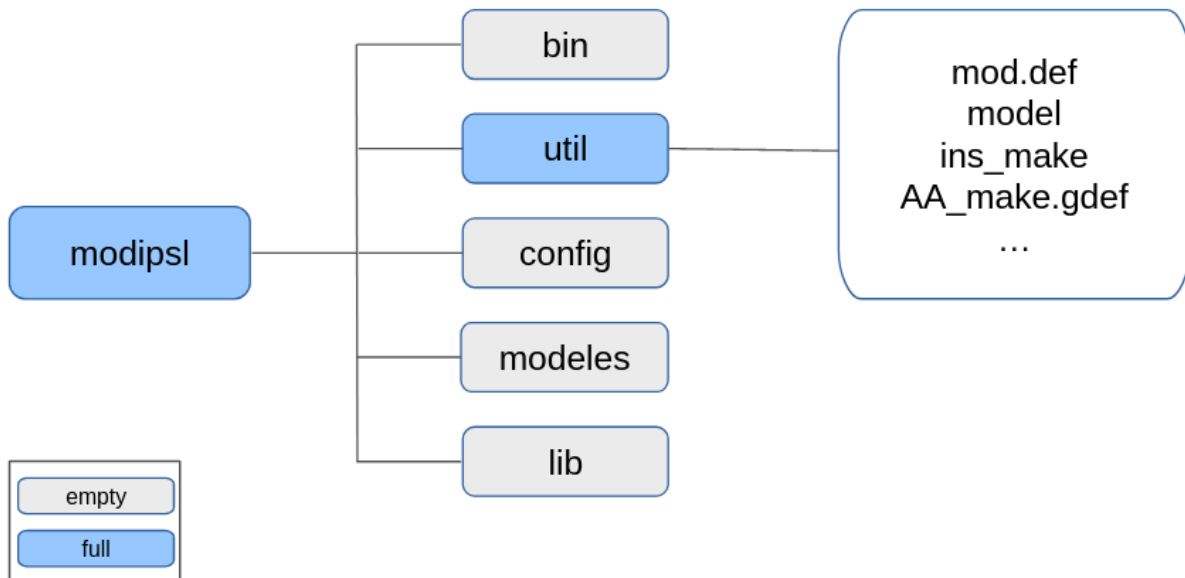
The passwords must be asked to a Platform group member.

Note: You might encounter the following message “Error validating server certificate for 'https://forge.ipsl.jussieu.fr:443'”. We are working on it, and to not have this message anymore you can answer `p` to accept the certificate permanently.

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 defined in the 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

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

- Review the models configurations to download (`mod.def`)
- Download the chosen configuration (`model`)

→ Explore the `util/` directory:

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

Note that some files (`ins_make`, `AA_make.gdef`) were used to create a makefile adapted to the downloaded configuration and the supercomputer on which you work. They are useful only if you are extracting a configuration older than v6.2 (for more recent configuration versions, we no longer create makefiles).

2.1 Extract sub configuration ICOLMDZOR from main IPSLCM7 configuration

Note for Spirit(X) and Obelix users:

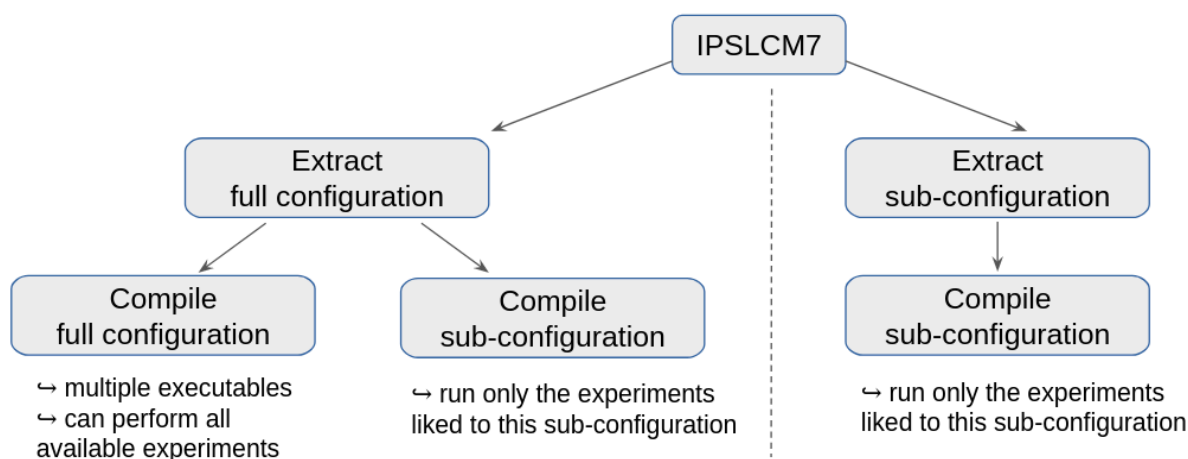
DYNAMICO has not yet been installed on Spirit(X) or Obelix. Therefore you should install a LMDZ-ORCHIDEE configuration without DYNAMICO. You'll also need to use a lower number of MPI processors and only 1OMP thread. This will be set in the text further below.

Preliminary notes: during this training course we'll be working with v7 model configurations. These configurations enable experiments to be carried out with both regular and icosahedral grids.

All v7-type configurations can be extracted from the [IPSLCM7 configuration](#). There are several solutions:

- Extract the “full” IPSLCM7 configuration, compile it in its entirety and with these executables perform all available experiments (coupled and forced).
- Extract the “full” IPSLCM7 configuration, compile only a sub-configuration, and then run only the experiments linked to this sub-configuration.
- [Extract a sub-configuration](#) from the IPSLCM7 configuration, and compile this sub-configuration.

Summary on how to use IPSLCM7 configuration



Note that if you are still working with a “v6” family configuration, the extraction of a sub-configuration is not possible yet.

Description of how to extract a chosen configuration: In `util/` directory, 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 `IPSLCM7_TP2024`) 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

Question 2a: Using the `./model -h` command, find out which version of LMDZ, ORCHIDEE and libIGCM are currently defined in the `IPSLCM7_TP2024` 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. (NB: this configuration offers two versions of ORCHIDEE: version `2_2` is used by default, but `ORCHIDEE_4` is also defined in the configuration, and can be compiled if needed)

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 IPSLCM7_TP2024
vi mod.def
```

In our case we don't want to extract all the coupled model, but just the sub-configuration `ICOLMDZOR`. The procedure is described on this page:

https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Install#v7family

- Download the sub-configuration `ICOLMDZOR` (IDRIS and TGCC users) or `LMDZOR` (Spirit(X) and Obelix users) from the configuration `IPSLCM7_TP2024` 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:

```
./model IPSLCM7_TP2024 ICOLMDZOR
```

For spirit(x) and obelix users:

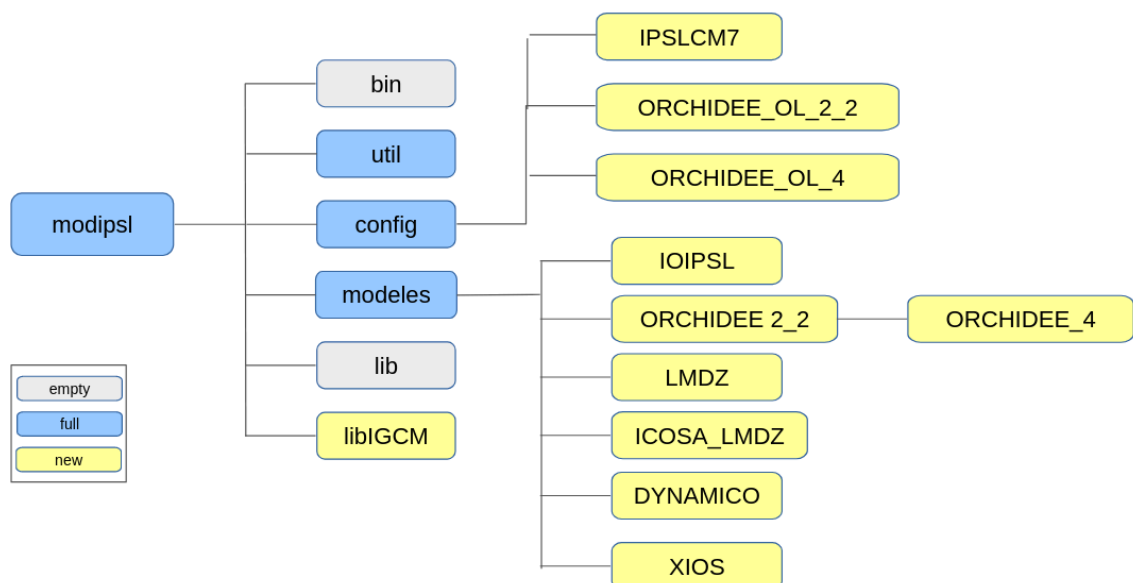
```
./model IPSLCM7_TP2024 LMDZOR
```

→ Explore the directories in `modipsl`.

You can see in `modipsl/modeles` that you have one directory per model. You also have the `modipsl/config/IPSLCM7` directory, and two directories to run ORCHIDEE model in offline mode: `modipsl/config/ORCHIDEE_OL_*` and the `modipsl/libIGCM` directory.

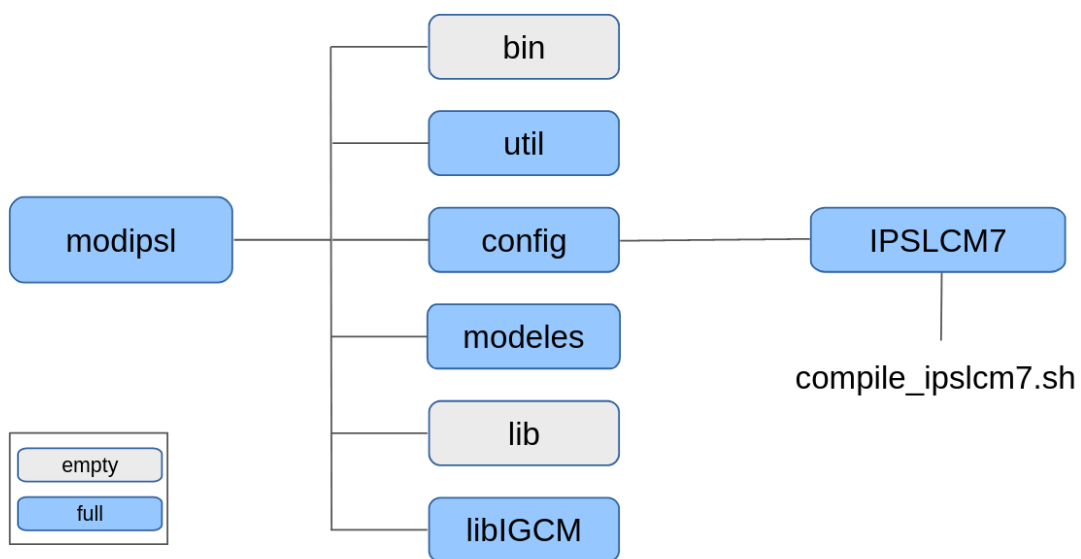
→ Type `svn info` in each model directory to get information about the extracted version and compare them with your answers to question 2.a.

Question 2b: Have all the components of the IPSLCM7 model been extracted? If not, which ones are missing?



2.2 Compile with the resolution 144x142x79

We use a specific script to launch the configuration compilation. This script is located in the `config/IPSLCM7` directory. The `config/ORCHIDEE_OL_*` directories are specific to run ORCHIDEE in offline configuration.



For this training course we have chosen to compile the model configuration ICOLMDZOR so that it can be used on both the regular grid ("beginner" exercises) and the icosahedral grid ("advanced" exercises). We will use the default version for the ORCHIDEE model, which is ORCHIDEE_2_2, but there's a compile option to switch for compilation of ORCHIDEE_4 if required.

Question 2c: Launch the help of the compilation script and try to find all available options for the compilation using `./compile_ipslcm7.sh -h`. You can also open the file to search for options that might not be documented in the help. Find out which resolution is the default one, then launch the compilation for both grids "regular" (with the resolution "144x142x79") and "icosahedral" (in this case the resolution is chosen when we launch the experiment). You can use these pages to help you to understand the script syntax:

- https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Compile#Scriptforconfigurations_v6.2andnewer

- https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/Compile#Compilationoptionsforconfigurations_v7andnewer

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
```

→ Launch the compilation script to compile the model with the regular grid (with default resolution 144x142x79 for the atmosphere) and the icosahedral grid

To compile, use the `compile_ipslcm7.sh` compilation script (compilation can take between 1h30 and 2h30 depending on the computer):

```
cd $WORK/MYFIRSTTEST/modipsl/config/IPSLCM7
./compile_ipslcm7.sh -h
./compile_ipslcm7.sh
```

For 2024 training course at IDRIS:

The compilation can take more than one hour for the regular and the icosahedral grid. We will stop it (only for the training day) You need to do the 2 following points:

- 1- stop the compilation (ctrl+c) after few seconds
- 2- copy the executables in your bin directory (be careful, the command is on a single line):

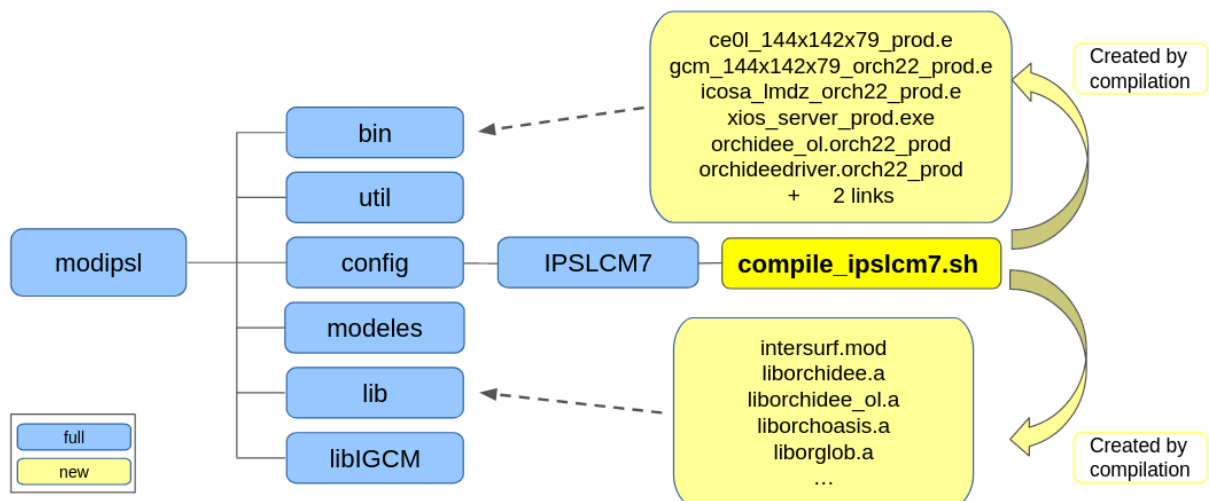
```
cp
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2024/ICOLMDZOR
/jean-zay/bin/* $WORK/MYFIRSTTEST/modipsl/bin/.
```

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.

When the compilation is over, you will find the executables in the `modipsl/bin` directory. 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 ICOLMDZOR configuration, the `icoso_lmdz_orch22_prod.e` executable is specific to the icosahedral grid, and `orchidee_ol.orch22_prod` and `orchideedriver.orch22_prod` executables are specific to run ORCHIDEE model in offline mode.

In the following exercises we will use the coupled configuration, which will only use `ce01`, `gcm`, and `xios_server` executables.

Question 2d: What if you want to recompile the whole code? Use the option `-h` or 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
```

BEGINNER

3. Basic simulations

In a configuration we can choose between several types of experiments, and we can run them *from the same executable*. All the available experiments are stored in the `EXPERIMENTS/` directory.

Now that you have chosen which model configuration you want to work with, and you have downloaded it and compiled it, you can choose which type of experiment you want to use.

For obelix users:

When using LMDZ at obelix, one parameter needs to be changed for the use of the filter at the poles. The default fft filter can not be used. Therefore set `use_filtre_fft=n` in the file `run.def` read by LMDZ.

This change can be done in the file `config/IPSLCM7/GENERAL/LMDZ/run.def` before creating the experiment folder or in the file `PARAM/run.def` inside the specific experiment folder.

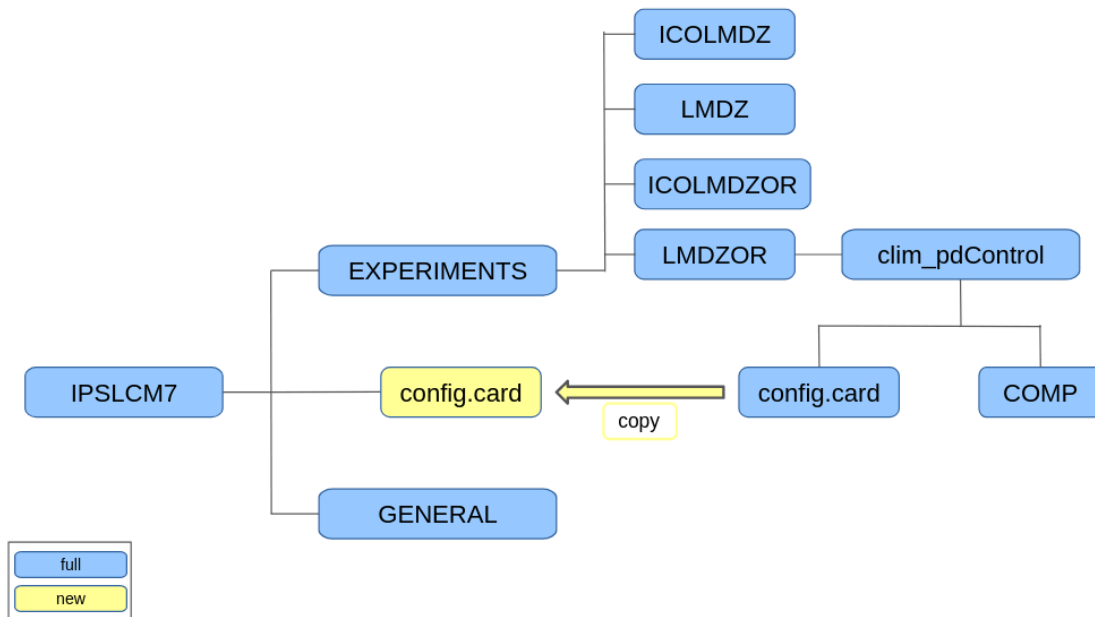
Read more about using LMDZ on obelix here:

https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/ComputingCenters/LSCE

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 IPSLCM7/ICOLMDZOR case, you can choose between the ICOLMDZOR, LMDZOR, LMDZ and ICOLMDZ types of experiments.

For this exercise we will create an experiment from `LMDZOR/clin_pdControl` (default experiment to run a control simulation of type present day). To do this, we will copy the `config.card` file located in `EXPERIMENTS/LMDZOR/clin_pdControl` into the `config/IPSLCM7/` 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/IPSLCM7
# At obelix first change to have use_filtre_fft=n in
# GENERAL/PARAM/LMDZ/run.def then continue

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_${ResolAtm}_orch22_${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 spirit(x) and obelix, in ATM: replace 71MPI,8OMP by 15MPI,1OMP
# On Irene skl or amd, change nothing for parallelization
  
```

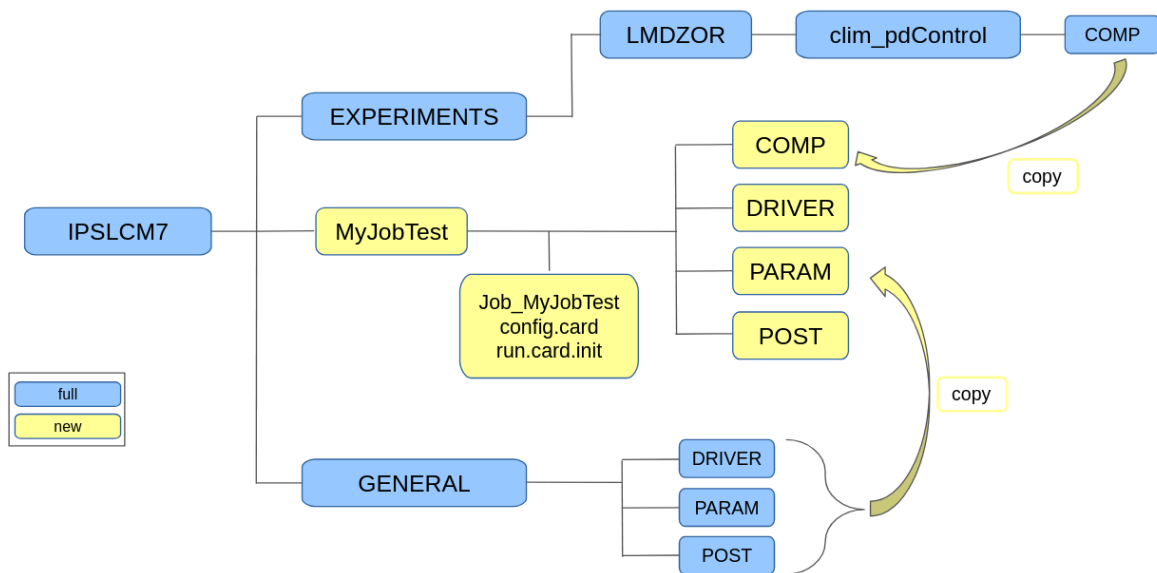
```

../../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.



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 the 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

3. Which calendar will you use?

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 except time-series. 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 the scratch file system of the computer (`$SCRATCH` for JeanZay, `$CCCSCRATCHDIR` for Irene).

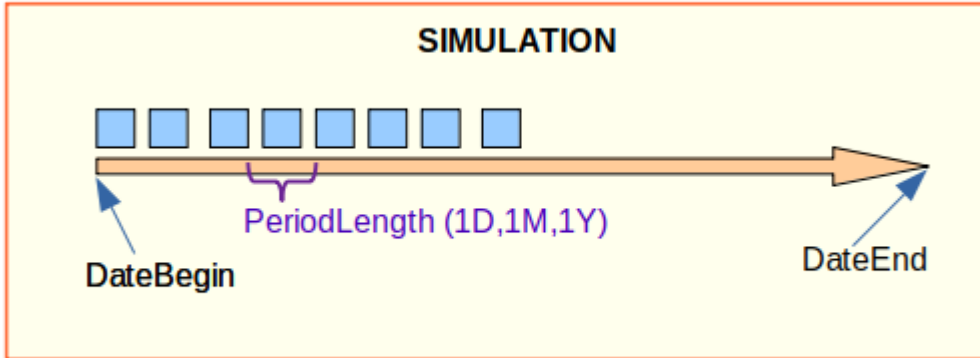
```
JobName=MyJobTest
#----- Short Name of Experiment
ExperimentName=clim
#----- DEVT TEST PROD
SpaceName=TEST
LongName="LMDZOR configuration"
TagName=LMDZOR
#D- Choice of experiment in EXPERIMENTS directory
ExpType=LMDZOR/clim_pdControl
```

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

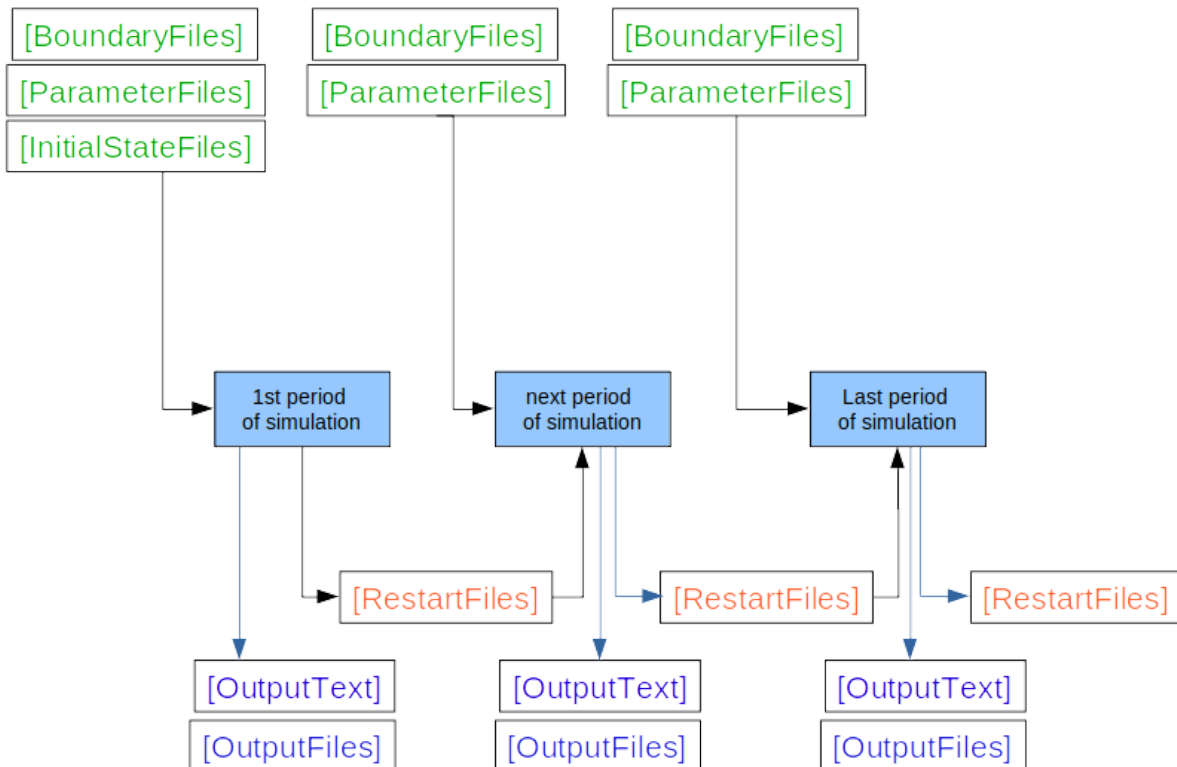
PeriodLength=1D

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- Activate C-ESM-EP atlas
Cesmep=FALSE
#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 = 10800s

```

- `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.

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=16 # Number of MPI tasks
#SBATCH --hint=nomultithread # 1 processus MPI par par physical core (no
hyperthreading)
#SBATCH --time=30 # Wall clock limit (minutes)

```

- **Header for obelix:**

This should come up as default:

```

#####
## OBELIX LSCE ##
#####
#PBS -N MyJobTest
#PBS -m a
#PBS -j oe
#PBS -q mediump
#PBS -o Script_Output_MyTest.000001
#PBS -S /bin/ksh
#PBS -v BATCH_NUM_PROC_TOT=16
#PBS -l nodes=1:ppn=16

```

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/IPSLCM7/MyJobTest/
```

```
JeanZay: sbatch Job_MyJobTest
```

Note: for 1 day of simulation, the job should run no more than 1 or 2 minutes.

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 and 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_mdcl (TGCC) / qdel (Obelix)
```

followed by your job ID:

On JeanZay or spirit(x)

```
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 obelix
qstat -u $user
>> Job ID        Username Queue   Jobname   SessID NDS    (...)
>> -----
>> 3653407.obelix0 jghattas mediump  MyJobnameO 133961  (...)
qdel 3653407.obelix0

```

→ 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 analyse the file `Script_Output`. It will give you important information on your simulation.

https://forge.ipsl.jussieu.fr/igcmg_doc/wiki/Doc/CheckDebug#AnalyzingtheJoboutput:ScriptOutput

→ 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

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: `/home/scratch01/$USER/`
- Spirit : `/data/$USER/`
- SpiritX : `/homedata/$USER/`

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

To ease the cleaning you can use either one of two scripts: `clean_or_continue.job` or `purge_simulation.job` stored in `libIGCM` directory.

- **Script `clean_or_continue.job`**

The script `clean_or_continue.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/IPSLCM7/MyJobTest:` (do not do it now if you want to keep results of your first simulation)

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

- **Script `purge_simulation.job`**

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/IPSLCM7/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

In this exercise, you will continue your simulation for 4 more days.

To continue a simulation, you will have to:

- Change the `DateEnd` in `config.card`, based on the last date you want to calculate. **Do not change** `DateBegin` **nor** `PeriodLength`.
- Use the script `clean_or_continue.job` which will update the parameter `PeriodState` in `run.card` and the job header:

```
../../../../libIGCM/clean_or_continue.job
```

Note that the script `clean_or_continue.job` will:

- modify the parameter `PeriodState` in `run.card` (currently it's equal to "Completed" and we want it to be "OnQueue");

- change the suffix value of lines "#MSUB -o / -e Script_Output_" in the job header (Job_* file), regarding the value of the parameter CumulPeriod in run.card.

If you don't want to use clean_or_continue.job script, you can modify run.card and Job_* file by yourself. The example below illustrates a simulation of 1 month which is extended for one more month.

Current situation:

```
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.

→ For the exercise, you have to **adapt this case to continue your simulation for 4 more days** (remember that `DateEnd` = last day of simulation).

Continue your simulation for 4 more days:

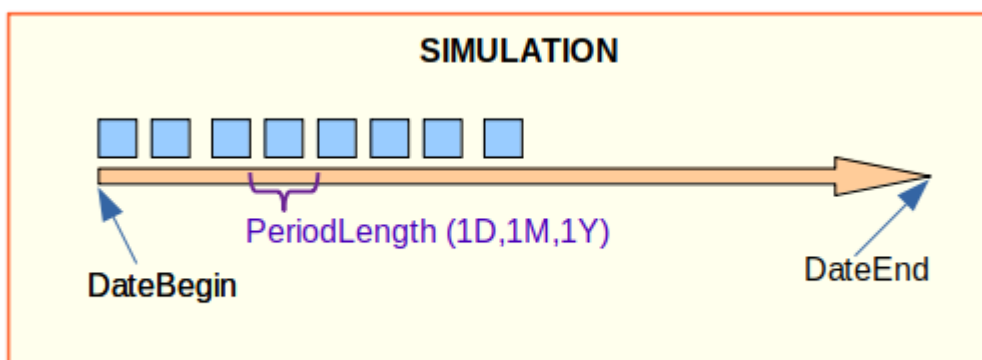
```

vi config.card # → Modify DateEnd
../../../../libIGCM/clean_or_continue.job

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 can modify the parameter `NbPeriodsPerJob` in the main Job. `NbPeriodsPerJob` is the number of Periods launched by the job for the given

CPU time. Note that if the CPU time is too small compared to the duration of the calculation, the job might stop before the end of the simulation.

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

→ Create a new simulation of 5 days

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 our 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.

→ Check your output files

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

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 computer centers.

→ 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.

```
cd $WORK/MYFIRSTTEST/modipsl/config/IPSLCM7/
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 Cesmep, TimeSeries and Seasonnal 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.
# you can modify NbPeriodsPerJob to calculate several periods per job.

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

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` files (`COMP/lmdz.card`, `COMP/orchidee.card`, etc.).

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

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

- 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 `IPSCLM7/` directory ; change some parameters as the `JobName`, `DateBegin` and `DateEnd` ; use `ins_job` command to initialise the simulation directory. Don't forget to modify the output frequencies for LMDZ, ORCHIDEE, and Stomate for the case of daily simulation.

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/clim #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.

- 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

`EXPERIMENT/LMDZ/CREATE_clim`, `EXPERIMENT/LMDZ/CREATE_clim_360d` and `EXPERIMENT/LMDZ/CREATE_amip` are two experiments set-up that launch the program `ce01.e` (stored as `ce01_144x142x79_prod.e` in `modips/bin/` directory), 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 IPSLCM7/ICOLMDZOR configuration is to first run the experiment `CREATE_clim`, `CREATE_clim_360d` 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/IPSLCM7
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`.

→ Submit the job as before:

```
sbatch Job_ELC-144x142x79 (Jean-Zay, spirit(x))
ccc_msub Job_ELC-144x142x79 (Irène)
qsub Job_ELC-144x142x79 (Obelix)
```

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/$USER` 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 does a year contain? 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 of 5 days 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:IPSLCM7 with sub-configuration ICOLMDZOR)

```
cd $WORK/MYFIRSTTEST/modipsl/util
./model IPSLCM7_TP2024 ICOLMDZOR
(+usernames and passwords)
```

3. Compile

```
cd $WORK/MYFIRSTTEST/modipsl/config/IPSLCM7
./compile_ipslcm7.sh [options]
```

4. Create experiment directory

```
cd $WORK/MYFIRSTTEST/modipsl/config/IPSLCM7
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/modipsl/config/IPSLCM7/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_2024/LMDZOR
```

- On jean-zay, IDRIS :

```
$WORK/../../../../rech/psl/commun/TRAINING/MODIPSL_HandsOn_2024/LMDZOR
```

- On spirit/spiritx:

```
/projsu/igcmg/TRAINING/MODIPSL_HandsOn_2024/LMDZOR
```

- On obelix:

```
/home/orchideeshare/igcmg/TRAINING/MODIPSL_HandsOn_2024/LMDZOR
```

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. Get more information here about the structure of the Script_Output file:

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 the beginner part.

- Create a new experiment for 1 day. Then copy the file `lmdz.card_1` from the directory above into the `lmdz.card` file in the COMP sub-directory in your new submit directory.

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_or_continue.job` to do this.

Question 4a: What was the error?

Copy the `lmdz.card_2` and debug again.

Question 4b: What was the error now?

Copy the `lmdz.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

```
cd modipsl/modeles/LMDZ/libf/phyImd/  
vi conf_phys_m.F90
```

Look for line

```
CALL getin('type_ocean', type_ocean_omp)
```

And add just after

```
write(lunout,*) "Debug in conf_phys : type_ocean = ", type_ocean_omp
```

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

→ Re-compile your models.

For 2024 training course at IDRIS:

The compilation can take more than one hour for the regular and the icosahedral grid. We will stop it (only for the training day) You need to do the 2 following points:

0- stop the compilation (ctrl+c)

1- copy the executables in your bin directory (be careful, the command is on a single line):

```
cp
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2024/LMDZOR/je
an-zay/Debug4.2/bin/* $WORK/MYFIRSTTEST/modips1/bin/.
(you need to accept all overwriting)
```

→ launch a 1 day test.

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. These files will be :

- If the simulation runs without a bug : in the SCRATCH directory if you are running in TEST, or STORE directory if you are running in DEVT or PROD. For a debug simulation we advise you to run in TEST.
- If the simulation stops with a bug : in the `Debug/` directory created by your simulation in your experiment directory.

```
cd $SCRATCH/IGCM_OUT/LMDZOR/TEST/clim/MySimulation/ATM/Debug
vi MySimulation_datebegin_dateend_out_lmdz.x.out
```

Or

```
cd $WORK/.../modips1/config/IPSLCM7/MySimulation/Debug
vi MySimulation_datebegin_dateend_out_lmdz.x.out
```

In both case look for “Debug in conf_phys : type_ocean”

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

We will still work with an experiment like “MyJobTest” from the **beginner** part. We will modify the LDMZ code to create a bug during the simulation. For this you will replace the file :

`modips1/modeles/LMDZ/libf/dyn3dmem/gcm.F90` by the one store on
TRAINING/MODIPSL_HandsOn_2024/LMDZ

→ As you modify the code you need to recompile.

```
./compile_ipslcm7.sh
```


You can also indicate more options to the script of compilation to indicate that you just want the lmdzor regular grid executable (the compilation will take few minutes)

```
./compile_ipslcm7.sh -regular_latlon yes -subconfig LMDZOR
```

For 2024 training course at IDRIS:

The compilation can take more than one hour for the regular and the icosahedral grid. We will stop it (only for the training day) You need to do the 2 following points:

0- stop the compilation (ctrl+c)

1- copy the executables in your bin directory (be careful, the command is on a single line):

```
cp
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2024/LMDZOR/je
an-zay/Debug4.3/bin/* $WORK/MYFIRSTTEST/modipsl/bin/.
```

(you need to accept all overwriting)

→ then launch a new experiment. Your simulation will not finish successfully
(PeriodState= Fatal in run.card)

In the file Script_Output you will find this message :

```
=====
EXECUTION of : /usr/bin/time srun --cpu-bind=none --distribution=arbitrary --multi-prog ./run_file
Return code of executable : 143
IGCM_debug_Exit : EXECUTABLE

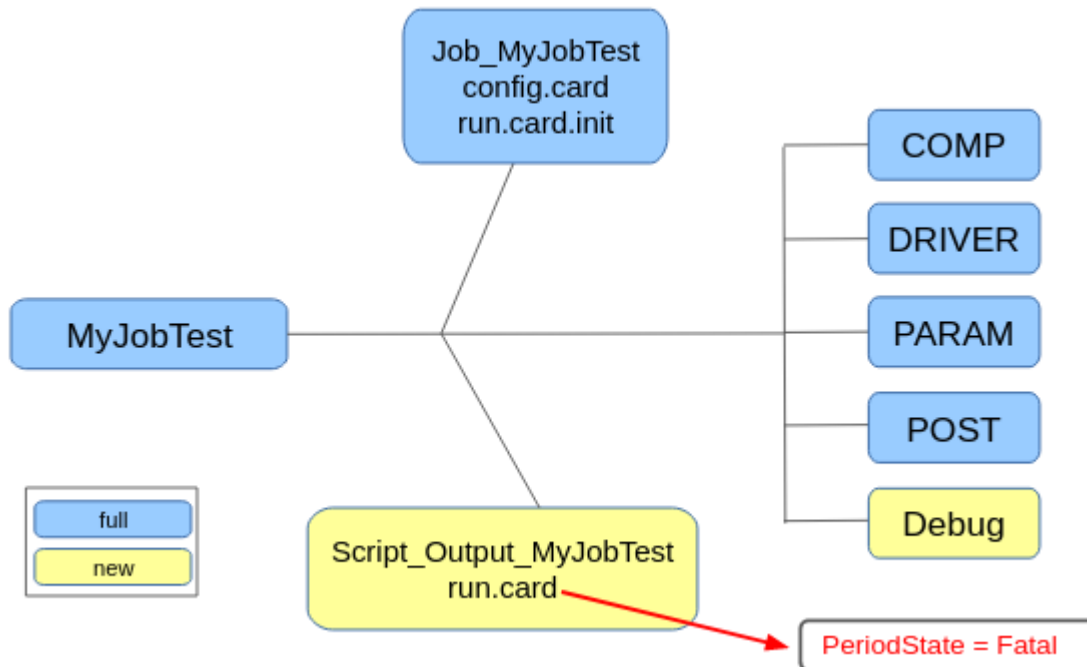
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! ERROR TRIGGERED !!
!! EXIT FLAG SET !!
!-----!

0 - IGCM_debug_Exit (_0_)
IGCM_sys_Mkdir : ****/modipsl/config/IPSLCM7/****/Debug
IGCM_sys_Cp : out_execution ****/modipsl/config/IPSLCM7/****/Debug/****_****_****_out_execution_error

to debug your experience, you can analyze the various files (*.err, *.out, and parameters files) available in the Debug/ directory
these files are managed by the list [OutputText] defined in componant's card (COMP/*.card)

=====
```

A new directory called `Debug` is created in your experiment directory.



You can read the description of this directory here

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.

You will have something like that: (this is a copy from JeanZay, but it will be similar on other computers)

```

| 0 out_lmdz.x.err.0
-----
Ofortrl: severe (71): integer divide by zero
0lmdz.x      PC          Routine      Line   Source
0lmdz.x      0000000002908BF4  Unknown          Unknown Unknown
0libpthread-2.28.s 0000154935359CE0  Unknown          Unknown Unknown
0lmdz.x      000000000041F781  Unknown          Unknown Unknown
  
```

0lmdz.x	00000000041D542	Unknown	Unknown	Unknown
0libc-2.28.so	00001549320A9CF3	__libc_start_main	Unknown	Unknown
0lmdz.x	00000000041D42E	Unknown	Unknown	Unknown

Actually we know that there is a division by zero, but we don't where. 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_ipslcm7.sh -debug
```

You can also indicate more option to the script of compilation to indicate that you just want the lmdzor regular grid executable

```
./compile_ipslcm7.sh -regular_latlon yes -subconfig LMDZOR -debug
```

→ Re-compile in debug mode (this compilation can take 50 minutes, if you compile only LMDZOR sub configuration)

For 2024 training course at IDRIS:

The compilation can take more than one hour for the regular and the icosahedral grid. We will stop it (only for the training day) You need to do the 2 following points:

0- stop the compilation (ctrl+c)

1- copy the executables in your bin directory (be careful, the command is on a single line):

```
cp /gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2024/LMDZOR/jean-zay/Debug4.4/bin/* $WORK/MYFIRSTTEST/modipsl/bin/.
```

→ Purge your experiment with `libIGCM/purge_simulation.job` script and modify the `config.card` file to indicate that we will use the debug executable.

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

→ Launch one more time your simulation.

This 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).

In our case :

```
-----  
| 0 out_lmdz.x.err.0  
-----  
Ofortrl: severe (71): integer divide by zero  
OImage      PC          Routine      Line   Source  
Olmdz.x      000000000886DC54 Unknown          Unknown Unknown  
Olibpthread-2.28.s 00001546FB867CE0 Unknown          Unknown Unknown  
Olmdz.x      0000000000427209 MAIN__          390 gcm.f90  
Olmdz.x      000000000041FF42 Unknown          Unknown Unknown  
Olibc-2.28.so 00001546F85B7CF3 __libc_start_main Unknown Unknown  
Olmdz.x      000000000041FE2E Unknown          Unknown Unknown  
^
```

is telling you that there is a problem at line 390 of gcm.f90.

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
```

→ Open the gcm.f90 file in pre-compiled directory, and find the line of code producing the bug.

Question 4d: what causes the bug ?

→ Remove the line in `modipsl/modeles/LMDZ/libf/dyn3dmem/gcm.F90`

→ Re-compile in prod mode (without the option -debug)

```
./compile_ipslcm7.sh
```

You can also indicate more options to the script of compilation to indicate that you only want the lmdzor regular grid executable

```
./compile_ipslcm7.sh -regular_latlon yes -subconfig LMDZOR
```

For 2024 training course at IDRIS:

The compilation can take more than one hour for the regular and the icosahedral grid. We will stop it (only for the training day) You need to do the 2 following points:

0- stop the compilation (ctrl+c)

1- copy the executables in your bin directory (be careful, the command is on a single line):

```
cp  
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2024/ICOLMDZOR  
/jean-zay/bin/* $WORK/MYFIRSTTEST/modipsl/bin/.
```

(you need to accept all overwriting)

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 IPSLCM7 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

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

```
cd $WORK/MYFIRSTTEST/modips1/config/IPSLCM7

cp EXPERIMENTS/LMDZOR/clim_pdControl/config.card .

vi config.card   ### Modify JobName=MyJobTest.debug
                  ### prepare for a 1 day simulation in TEST

../../../../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
```

→ Explore this new directory.

Question 4e: Do you understand what these files are ?

→ Launch your simulation from this new directory

INTERMEDIATE

5. Post_processing : how to quickly visualise the main diagnostics of a simulation

With the libGCM workflow, several types of post-processing are available to quickly visualise the main diagnostics of a simulation. LibGCM will create [Time Series](#), will monitor them in a [Monitoring](#), and use them to create [C-ESM-EP](#) atlas to compare them to observation or another simulation.

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.

In this section you will launch a simulation of 2 years using Time Series and associated C-ESM-EP atlas.

Note for spirit(x) and obelix users:

The C-ESM-EP is not yet installed at obelix and not fully working at spiritx. You need to deactivate it by setting `Cesmep=FALSE` in `config.card`.

At spirit, C-ESM-EP is working.

5.1 Launch 2 years with default Time Series and default C-ESM-EP atlas

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.

In the folder `modipsl/config`, you'll find 2 folders `ORCHIDEE_OL_2_2` and `ORCHIDEE_OL_4` which contain experiments to run ORCHIDEE (`ORCHIDEE_2_2` or `ORCHIDEE_4`) in offline mode. You can use one of those corresponding to the compilation done. Default compilation is done with version `ORCHIDEE_2_2` so you should use the corresponding `config/ORCHIDEE_OL_2_2` folder.

As you didn't compile the model in this config folder, you need to set a link to the arch.env file which corresponds to the one used for the compilation as in the example below.

```
cd ORCHIDEE_OL_2_2
cd ARCH
ln -s arch-****.env arch.env
## for JeanZay : ln -s arch-X64_JEANZAY.env arch.env
```

For 2024 training course at IDRIS:

Link ORCHIDEE directory to ORCHIDEE_2_2 (usually done by compilation script)

```
cd modeles
ln -s ORCHIDEE_2_2 ORCHIDEE
```

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_2_2/ 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).

```
cd $WORK/MYFIRSTTEST/modipsl/config/ORCHIDEE_OL_2_2/ARCH
ln -s arch-X64_JEANZAY.env arch.env
# at jean-zay: ln -s arch-X64_JEANZAY.env arch.env
# at irene: ln -s arch-X64_IRENE.env arch.env
# at meso-ipsl: ln -s arch-ifort_MESOIPSL.env arch.env
# at obelix: ln -s arch-ifort_LSCE.env arch.env

cd ..
cp -r OOL_SEC_STO_FG2 MyPostExp
cd MyPostExp
```


→ Modify `config.card`, `run.def` and `COMP/stomate.card`.

How to modify `run.def`:

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

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

How to modify `config.card`:

- Just like every time you configure a new experiment, you have to change `JobName` (your experiment name = `MyPostExp` in this exercise)
- Modify `DateEnd` for a simulation of 2 years (Remember : `DateEnd` is the last day of the simulation)
- 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`.
- 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`).
- Calculate the `Seasonal Means` over 2 years by setting `SeasonalFrequency=2Y`.
- On JeanZay you need to specify your `DataProject` to avoid a bug in libGCM for Cesmep Atlas post-treatments
- Check that `Cesmep=TRUE`

For 2024 training course at IDRIS:

Unfortunately, C-ESM-EP atlas do not work on training accounts. They must therefore be deactivated.

In `config.card`: `Cesmep=FALSE`

You can, however, view the final result

For a simulation done with ORCHIDEE_offline model :

https://thredds-su.ipsl.fr/thredds/fileServer/idris_thredds/work/rpsl592/C-ESM-EP/OL2/DEVT/secsto/OOLTP2024.2/Analyse/OOLTP2024.2_standard_comparison/C-ESM-EP_OOLTP2024.2_standard_comparison.html

For a simulation done with IPSLCM6 coupled model :

https://thredds-su.ipsl.fr/thredds/fileServer/tgcc_thredds/work/p24cozic/C-ESM-EP/IPSLC_M6/DEVT/piControl/ESMGES.20231218.01/Analyse/ESMGES.20231218.01_run_comparison/C-ESM-EP_ESMGES.20231218.01_run_comparison.html

```
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=2Y
                # => Add : DataProject=*** #your project
                # Cesmep=TRUE
```

At obelix and spiritx:

vi config.card => Change to Cesmep=FALSE

At spirit: C-ESM-EP is working, you can keep Cesmep=TRUE

At obelix: when libGCM launches the post-treatment job for the time-series and seasonal averages, an error message will appear. You need to launch these jobs using TimeSeries_Checker.job and SE_Checker.job afterwards as explained in the next section.

How to modify COMP/stomate.card:

- To avoid a bug due to the short length of the simulation you need to modify the type of Time Series for stomate_ipcc_history output file

```
vi COMP/stomate.card
[Post_1M_stomate_ipcc_history]
(...)
ChunckJob2D=100Y
```

→ Create the job with `ins_job`

Question 5a: What is different from previous experiment directories ?

→ In the main Job, set `NbPeriodsPerJob=2` and `#SBATCH --time=30` or `#SBATCH --time=00:30:00` (IDRIS, Spirit(x)) or `#MSUB -T 1800` (TGCC).

→ If you have not already done it, if you are working on Jean Zay you need to [install your environment for C-ESM-EP](#). Nothing needs to be done for Irene or spirit. The C-ESM-EP is not yet installed at obelix and currently not working on spiritx.

```
# At JeanZay
module load singularity
container=/gpfswork/rech/psl/commun/Tools/cesmep_environment/20230611_V3.0_IPSL8.s
if
idrcontmgr cp $container
```

→ Submit the job (using `sbatch`, `ccc_msub` or `qsub` depending on the machine).

If you follow your simulation with `squeue` (IDRIS, spirit) / `ccc_mpp` (TGCC) / `qstat` (Obelix) you can see that several jobs will be launched. First your simulation job, then jobs for the pack, then for the Time Series, and finally for the Monitoring and the C-ESM-EP atlas.

Question 5b: Once the simulation and post-processing are complete (~15 minutes), check Time Series in the archive directory (see the following directories: `IGCM_OUT/[...]/JobName/***/Analyse/TS_MO`) and the seasonal averages (see `IGCM_OUT/[...]/JobName/***/Analyse/SE`). If now files were found (case at obelix), launch manually the `TimeSeries_Checker.job` as described in the following section.

Now change directory to the `cesmep_lite` directory. Use a command to view files creation dates.

```
cd cesmep_lite
ls -lrt
```

→ Open `libIGCM_post.out` and find the path of the html web page created to visualize the atlas

```
-- The CliMAF ESM Evaluation Platform atlas is available here:
--
--
https://thredds-su.ipsl.fr/thredds/fileServer/idris_thredds/work/***/C-ESM-EP/OL2/DEVT/secsto/***/Analyse/***_standard_comparison/C-ESM-EP_***_standard_comparison.html
--
--
-- The html file is here:
--
/gpfswork/rech/***/***/C-ESM-EP/OL2/DEVT/secsto/***/Analyse/***_standard_comparison/C-ESM-EP_***_standard_comparison.html
```

→ Open this web page

5.2 Add variables to Time Series and relaunch with the TimeSeries_Checker.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 5c: How to check that "z0h" and "z0m" variables exist in the simulation file?

Question 5d: How to use TimeSeries_Checker.job?

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

INTERMEDIATE

6. Monitoring and Inter-monitoring

The monitoring is a web-interface tool that visualises 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

6.1 Monitoring

For example, you can visualise 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

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: <https://webservices.ipsl.fr/interMonitoring/>

For this exercise choose the 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

- CM61-pi-valid.02.JZ:

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 <https://webservices.ipsl.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" (<https://webservices.ipsl.fr/simuFinder/>). In the left window of this application you can write "tags" to describes simulations you want to visualise. 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 <https://webservices.ipsl.fr/simuFinder/> 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)
 1. click on the path on the right window and drag it on the bottom one
 2. 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.
 3. click on the path and drag it to the bottom window
 4. 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

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/IPSLCM7
cp EXPERIMENTS/LMDZOR/clim_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_${ResolAtm}_orch22_${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 NbPeriodsPerJob to not relaunch the simulation between two periods of
simulation.
NbPeriodsPerJob=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

```


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

Check that this file exists (\$CCCSTOREDIR on TGCC):

```
ls
$STORE/IGCM_OUT/LMDZOR/DEVT/clim/MyJobTest-6D/ATM/Output/DA/MyJobTest-
6D_19800103_19800104_1D_histday.nc
```

then remove it:

```
rm -f
$STORE/IGCM_OUT/LMDZOR/DEVT/clim/MyJobTest-6D/ATM/Output/DA/MyJobTest-
6D_19800103_19800104_1D_histday.nc
```

7.3 Apply the method to redo days 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/IPSLCM7
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 did the first
two 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/IPSLCM7/MyJobTest-6D-REDO
# remove lines for periods 3 to 6 at the end of the file (because in our REDO
simulation, these periods don't exist yet)
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/MyJobT
est-6D_19800103_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/.
```

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 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`. To see where they are written in the model, you have to 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 done 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.card` (section `[UserChoices]`). To save a variable at a specific frequency, the corresponding 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, you have to :

- Continue in the same modipsl where you ran 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="2">

  <field_group group_ref="remap_1h" >

    <field_group grid_ref="grid_out" >

      <field field_ref="slp" level="5" />

    </field_group>

  </field_group>

</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 line is in the header of your job file
# for jean-zay
#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 `hsthf.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/NEMO (note that directory  
PARAM 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

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 libIGCM 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 (`t2m`) 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/x.x          2) nco/x.x
3) ferret/x.x          4) netcdf/4.7.2-mpi
5) ncview/x.x          6) cdo/1.9.7.1
7) ncl/x.x             8) python/3.x
```

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 ferret
module load nco
module load cdo
module load ncview
module load ncl
module load python/3.7.6
module load netcdf-c/4.7.2
```

Irene

```
module purge
module load ferret
module load nco
module load cdo
module load python3
module load ncl_ncarg
module load ncview
```

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 9.0.1 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 with only 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 -W 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 -W fldmean t2m_TS_CDO.nc t2m_glob_mean_CDO.nc
# Zonal average
cdo -W 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

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@tmXLabelStride = 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 4 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 9.0.1 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_TP2024` 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_TP2024
```

→ 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 :

```
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. Here, we will also 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 :

```
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 .
```

→ Set the following informations for your experiment in the `config.card` file :

```
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` file 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 the variables NO3, PO4, Si, Fe, DCHL, NCHL to the specific file group "5d" in the `file_def_nemo-top.xml` file :

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

→ Check that in the `<!--5d files-->` section the `enabled` parameter is set to `"_AUTO_"`

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

→ Add the 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.nc` 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, NO3, Si and Fe.

→ Open the `pisces.card` file :

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

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

→ Open the file :

```
vi ../../../../modeles/NEMO/cfgs/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`

Here you will 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 from the `namelist_pisces_ref` the lines below :

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

→ Paste them 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

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 modips1/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=n
```

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

```
vi ensemble.card # (for IRENE only)
# write this following lines in [Ens_PERTURB] section
active=y
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
```

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="${CCCWORKDIR}/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/CM61-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/Restart
```

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

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 in IPSLCM7 configuration. So you have to :

- Extract modipsl
- Extract IPSLCM7_TP2024 full configuration
- Compile (./compile_ipslcm7.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 2024 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 (beware of the long size of the line) :

```
cp  
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2024/IPSLCM7/j  
ean-zay/bin/* $WORK/MYFIRSTTEST_COUPLED/modipsl/orbin/.
```

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

```
cd modipsl/config/IPSLCM7/ARCH  
ln -s arch-X64_JEANZAY.env arch.env
```

3- (if the link doesn't already exist) link ORCHIDEE directory to ORCHIDEE_2_2 (usually done by compilation script)

```
cd modeles  
ln -s ORCHIDEE_2_2 ORCHIDEE
```

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

exit

At this stage, you have two possibilities :

- Run **IPSLCM-reg** configuration : this configuration is LMDZ-ORCHIDEE on regular grid 144x143 coupled to NEMO4 at ORCA1 resolution.
- Run **IPSLCM-ico** configuration : this configuration is DYNAMICO-LMDZ-ORCHIDEE on icosahedral grid nbp 40 (200 km horizontal resolution) coupled to NEMO4 at ORCA1 resolution.

12.1 IPSLCM-reg configuration (piControl experiment)

- Set up a 5 days piControl experiment (IPSLCM-reg/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 IPSLCM-ico configuration (piControl experiment)

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

12.3 IPSLCM-reg configuration (piControl experiment) with CMIP6 workflow

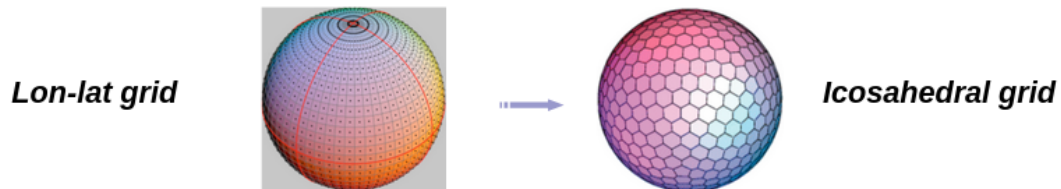
General information on the workflow functionality is available in part 8.4. Note that the workflow functionality is only available for LMDZ and ORCHIDEE component in IPSLCM7 configuration and for IPSLCM-reg experiments.

- Set up a 5 days piControl experiment (IPSLCM-reg/piControl_TEST experiment)
 - a. SpaceName=TEST
 - b. PackFrequency =NONE
 - c. TimeSeriesFrequency=NONE
 - d. SeasonalFrequency=NONE

- e. **dr2xmlIPSL=TRUE** (to be added in Post section)
- Please, refer to “*Definition of Output files in COMP/*.card*” in section 3.2 in order to activate daily outputs
- Launch the simulation
- Check output files of the simulation, especially CMIP6 workflow outputs on \$WORKDIR/.../IGCM_OUT/IPSLCM7/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.

You already extract ICOLMDZOR configuration in the first part of this training course. If you start with this exercise you need to extract it now.

For users starting out with this exercise (without doing the beginner part) :
→ extract modipsl

```
cd modipsl/util
./model IPSLCM7_TP2024 ICOLMDZOR
```

ICOLMDZOR configuration is the offline atmosphere-surface configuration that uses DYNAMICO as dynamical core. It's a subconfiguration to IPSLCM7 configuration. With this configuration we can run LMDZ physics and ORCHIDEE land surface on an icosahedral grid. We can also run LMDZOR configuration as we done in the beginner part of this training.

Question 13a: use “-h” option to know all options of the compilation script `compile_ipslcm7.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 ?

For users starting out with this exercise (without doing the beginner part) :

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 2024 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 (beware of the long size of the line):

```
cp
/gpfswork/rech/psl/commun/TRAINING/MODIPSL_HandsOn_2024/ICOLMDZOR/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/ICOLMDZOR_v7/ARCH
ln -s arch-X64_JEANZAY.env arch.env
```

3- (if the link doesn't already exist) link ORCHIDEE directory to ORCHIDEE_2_2 (usually done by compilation script)

```
cd modeles
ln -s ORCHIDEE_2_2 ORCHIDEE
```

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_pdControl experiment)
 - a. SpaceName=TEST
 - b. PackFrequency=NONE
 - c. TimeSeriesFrequency=NONE
 - d. SeasonalFrequency=NONE
 - e. Cesmep=FALSE
- Please, refer to “*Definition of Output files in COMP/*.card*” in section 3.2 in order to activate daily outputs
- 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 in `histday.nc` output file.

This exercise allows us to experiment with 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.