# Hands on tutorial #1: First steps with the model

The LMDZ team

December 14-18, 2020

This first tutorial focuses on installing and making basic first runs using the LMDZ model.
This document can be downloaded as a pdf file:

```
wget http://www.lmd.jussieu.fr/~lmdz/pub/Training/Tutorials/Tutorial_1.pdf
```

which should ease any copy/paste of command lines to issue.

## 1 Prerequisits

To run LMDZ, you will need a significant amount of memory, so first ensure this is true. You can use the following command line (that you can also, for convenience, put in your ∼/.bashrc file):

```
ulimit -Ss unlimited
```

If you are working on a laptop provided by the LMD and dedicated to these training sessions, you have to log in as the default (and sole) user util1 (the associated password will be given during the training session). On those machines, a working folder, named LMDZ, contains all the required archives ; if it's not already there, you can create it:

```
mkdir LMDZ
cd LMDZ
```

You may want to copy this full folder on a USB key at the end of the sessions.

## 2 Running the install_lmdz.sh script

The first step consists in downloading it from the LMD website and *blindly* running it (after having first set the access permissions to make it executable):

```
wget http://www.lmd.jussieu.fr/~lmdz/pub/install_lmdz.sh
chmod +x install_lmdz.sh
./install_lmdz.sh -d 32x32x39 -v 20201109.trunk
```

In the present case, option `-v 20201109.trunk` implies that the version built on the 9th of November 2020 will be installed, and option `-d 32x32x39` that a bench testcase at resolution 32×32-L39 will be run.
Compilation should take around five minutes. The test bench is then downloaded (or copied over if already on the disk) and the 1 day long test simulation on a regular 32×32-L39 grid is run ; messages about various downloads (via **wget**) and/or compiler informations are displayed.

The script should then run smoothly (if it isn't the case, immediately ask for some assistance) and end with messages of the likes of:

```
############################################################
Simulation finished in ...
You have compiled with:
```

```
./makelmdz...
You may re-run it with : cd  ...
or ./bench.sh
############################################################
```

You can take advantage of the installation time to open a second terminal window and explore the downloaded directories and files.

`install_lmdz.sh` will check if some archives (LMDZ, NetCDF library, etc.) are on the disk (in the `~/LMDZ` directory and subdirectories) and if not, will try to retrieve them through the network using `wget` command. It will create the **LMDZ20201109.trunk** directory ; inside, you will find subdirectories **modipsl**, which contains the model, and **netcdf-4.0.1**, which contains the NetCDF library.

In **modipsl**, you will find directory **modeles**, containing the **LMDZ** directory.

Once the test bench simulation has been launched (the final step of the **install_lmdz.sh** script), you will also find a **LMDZ/BENCH32x32x39/** directory from where you will be able to list the outputs of the run (even if the simulation is still running: it indeed takes a few minutes to complete the 1 day-long run on a single processor). Check out the contents of this directory and use your favorite software (Grads, Ferret,...) to browse the contents of the **histday.nc** file.

The default behavior of the `install_lmdz.sh` script is to install the model using the `gfortran` compiler, which is fine on the laptops provided for this training course. In any case, you can switch to another compiler using the `-compiler` option of the `install_lmdz.sh` script.

Several other features of the script can be modified or unactivated just by editing few parameters in the file or specifying the appropriate option (run `./install_lmdz.sh -h` to learn about these). For example, you can disable the NetCDF download and installation (first operation performed by `install_lmdz.sh`) in case this library is already present on the computer you are using.

# 3 Changing the outputs

The model outputs are controled in file **config.def** by the following lines:

```
phys_out_filekeys=       y       y       y       n       n
phys_out_filenames=      histmth histday histhf  histins histLES
phys_out_filelevels=     10      5       4       3       4
phys_out_filetypes=      ave(X)  ave(X)  ave(X) inst(X) inst(X)
phys_out_filetimesteps=   5day   1day    6hr     6hr     6hr
```

Note the `phys_out_filekeys` flags (y/n) which set which file is generated. In the present case, you will obtain 3 files (apart from ORCHIDEE related outputs): `histmth.nc`, `histday.nc` and `histhf.nc`, containing respectively 0 days (should contain 1 record for the whole run, but the output frequency set for `histmth.nc` above is `5day`, whereas the run was only 1 day long), daily and 6-hour averages.

You can find out which variables have been written in a given file by running the command:

```
ncdump -h histday.nc | grep long_name
```

To get a `histmth.nc` with data, you would need to run the model over 5 days. This means you would need to change the value of **nday** in file **run.def** to 5. To run a new simulation, issue command `./gcm.e`; if all goes well, it should end with the message `"Everything is cool"`.

You may also re-run the simulation with a higher output frequency for few variables (e.g. atmospheric temperature at 2m above the surface: **t2m**, surface pressure: **psol**, precipitation: **precip**, etc.), and other fields types (for instance instantaneous fields) by modifying **config.def**:

```
phys_out_filekeys=       y       y       n       y       n
phys_out_filenames=      histmth histday histhf  histins histLES
phys_out_filelevels=     10      5       4       3       4
phys_out_filetypes=      ave(X)  ave(X)  ave(X) inst(X) inst(X)
phys_out_filetimesteps=   5day   1day    6hr     6hr     6hr
flag_t2m=                10      10      5       3       5
flag_psol=               10      10      5       3       4
flag_precip=             10      10      4       3       5
```

Note that unlike other frequencies, 1TS (1 Time Step) is written in capital letters.
You can check in particular the **histins.nc** file and explain the special results.

# 4 Making some sensitivity test runs

You can now move on to changing parameters in a .def file. One may for instance deactivate the subgrid orography parametrizations by changing the **ok_orodr** and **ok_orolf** flags in **physiq.def**. One could also choose to change the value of the cloud water to rain conversion factor **cld_tau_lsc** (in **physiq.def**), or the atmospheric $CO_2$ concentration, etc. (ask around to find out the meaning of the various available parameters).

Just run the simulation (in a different directory to avoid overwritting output files) and investigate the differences between simulations (using `diff` or your favorite visualization software).

# 5 Preparing things for tomorrow's course

Tomorrow you will need to run the model (at 32×32-L39 resolution) and investigate the outputs. To prepare things for that you must, in a dedicated directory, run the model for 5 days.

- go to `LMDZ20201109.trunk/modipsl/modeles/LMDZ` and use the file called:

  ```
  bench_lmdz_32x32x39.tar.gz
  ```

  to create a new experiment:

  ```
  mv BENCH32x32x39 BENCH32x32x39_old
  tar -xf bench_lmdz_32x32x39.tar.gz
  cd BENCH32x32x39
  ```

- make sure that `nday=5` in `run.def`
- to avoid recompiling the code, just create a link to the executable you have already compiled before:

  ```
  ln -s ../BENCH32x32x39_old/gcm.e .
  ```

- open the config.def files and after the section where the outputs are defined:

  ```
  phys_out_filekeys=       y       y       y       n       n
  phys_out_filenames=      histmth histday histhf   histins histLES
  phys_out_filetimesteps=  5day    1day    1hr      6hr     6hr
  phys_out_filelevels=     10      5       5        4       4
  phys_out_filetypes=      ave(X)  ave(X)  inst(X) inst(X) inst(X)
  ```

  add these lines:

```
flag_vitu= 0 0 0 0 0
flag_vitv= 0 0 0 0 0
flag_du_gwd_hines= 0 0 0 0 0
flag_dv_gwd_hines= 0 0 0 0 0
flag_duoro= 0 0 0 0 0
flag_dvoro= 0 0 0 0 0
flag_dulif= 0 0 0 0 0
flag_dvlif= 0 0 0 0 0
flag_dthin= 0 0 0 0 0
flag_dtoro= 0 0 0 0 0
flag_dtlif= 0 0 0 0 0
flag_dteva= 0 0 0 0 0
flag_dtlsc= 0 0 0 0 0
flag_dtvdf= 0 0 0 0 0
flag_dtcon= 0 0 0 0 0
flag_dtwak= 0 0 0 0 0
flag_dtthe= 0 0 0 0 0
flag_dtajs= 0 0 0 0 0
flag_dtswr= 0 0 0 0 0
flag_dtlwr= 0 0 0 0 0
flag_dqeva= 0 0 0 0 0
flag_dqlsc= 0 0 0 0 0
flag_dqvdf= 0 0 0 0 0
flag_dqcon= 0 0 0 0 0
flag_dqwak= 0 0 0 0 0
flag_dqthe= 0 0 0 0 0
flag_dqajs= 0 0 0 0 0

ok_hines=y
```

- Run the model using `./gcm.e > listing`.

# 6 Optional: Running an aquaplanet configuration

The simulations run in the previous example included initial and boundary conditions (**start*.nc** and **limit.nc** files). It is also possible to run the model in "aquaplanet" configuration, with idealized initial and boundary conditions are used (no topography and imposed surface temperatures).

Make an **AQUAPLANET** directory where you will run the model. Copy over the **gcm.e** and ***.def** files from another simulation to this directory. Edit the **gcm.def** file to set

```
read_start=n
iflag_phys=101
```

Note that there are various possibilities for flag **iflag_phys**, values between 101 and 114 correspond to different choices of imposed SSTs (see routines **iniaqua** and **profil_sst** in file **phyaqua_mod.F90** in the **LMDZ/libf/phylmd** directory).

Then run the model:

```
./gcm.e > listing 2>&1
```

Inspect the various outputs (e.g. zonal averages and deviations of meteorological quantities such as surface pressure *psol*) and their temporal evolution, and compare to previously obtained simulations. Experiment changing the length of the run (variable **nday** in the **run.def** file to investigate

the evolution of the system from initial settings to a "converged" state.

If time permits, experiment running at a different resolution (e.g. 96×95-L39 ; which means you will have to recompile the model). Note that the number of time steps per day depends on the model resolution, and has to be higher for this resolution than it was strictly required for the 32×32-L39 case. The gcm.def file you've used has already **day_step=480**, which is enough, but you can check that the model will crash if you reduce it significantly (try it !).

# 7 Optional: Switching to a different version of the model

By using the installation script with option `-v 20201109.trunk`, the version dated November 9th 2020, which corresponds to *svn release* 3784, has been installed. This information can moreover be obtained directly from within the LMDZ directory:

```
cd ~/LMDZ/LMDZ20201109.trunk/modipsl/modeles/LMDZ
svn info
```

Note: you may encounter the following error when typing *svn info*:

```
svn: E155036: Voir la commande 'svn upgrade'
svn: E155036: The working copy at '/home/util1/LMDZ/LMDZ20201109/modipsl/modeles/LMDZ'
is too old (format 10) to work with client version '1.8.8 (r1568071)' (expects format 31).
```

If so, simply follow the suggested cure. Enter: `svn upgrade`. Then try `svn info` again.

Using svn, it is possible to change from model version and select and older or newer `svn release`. To change from current version to *svn release* 3776 (a previous version), go to directory **LMDZ** and issue command:

```
svn update --revision 3776
```

In the **LMDZ** directory, recompile LMDZ. To do so, run[1] **./makelmdz_fcm -arch local -rrtm true -d 32x32x39 -v orchidee2.0 -cpp ORCHIDEE_NOZ0H gcm**
Options **-v orchidee2.0 -cpp ORCHIDEE_NOZ0H** is not mandatory. They only indicate that we compile with Orchidee, but in fact, apart from the first bench, automatically launched by the `install_lmdz.sh` script, it won't be used in this tutorial. Only routines modified by the svn update or depending on them will be recompiled.

Once the model has been successfully recompiled, run a new simulation. To do so, create a new subdirectory in **LMDZ** (e.g. **BENCH32x32x39.3776**) and copy boundary conditions, initial conditions and parameters files (**limit.nc, start*nc, *.def**) over from directory **BENCH32x32x39**, along with the newly created gcm executable.
As you used **makelmdz_fcm**, it will be **LMDZ/bin/gcm_32x32x39_phylmd_seq_orch.e** and you should copy it over to the directory where you will run the model, e.g.:

```
cp LMDZ/bin/gcm_32x32x39_phylmd_seq_orch.e BENCH32x32x39.3776/gcm.e
```

Then launch the run[2]:

```
./gcm.e > listing 2>&1
```

Redirection of the model outputs in a text file is usually the best way to keep a trace of the run.

[1] To avoid typing such a long line over and over, it is more convenient to have it in a script, e.g. **compile.sh** to execute each time a compilation is required.

[2] As for the compilation step, this step can be put in a script, e.g. **bench.sh**, as is the case in **BENCH32x32x39**

Once the simulation is finished, you can compare, using the **diff** command and visualisation software, the simulation results (compare files **listing**, **hist*.nc restart.nc** and **restartphy.nc** from the two simulations)

Once this test finished, revert to the *trunk* 3784 version of the model by going to directory **LMDZ** and using:

```
svn update --revision 3784
```

Note that using **svn update** without any specific revision number implies updating to the latest version on the branch (which is something you should do regularly if you want to keep up with model updates). Don't forget to recompile the model after any **svn update**!