

Hands on tutorial session 1: First steps with the model

The LMDZ team

December 15, 2014

This first tutorial essentially 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/Distrib/TD1_en.pdf
```

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

If you are working on a machine which is part of the LMD local network, then you have to log in as "tdlmdz" or something similar (the associated password will be given during the training session). Log in and go to the local machine a directory where you will work:

```
cd /home/tdlmdz
```

If on the local LMD machines, it is important to work in the **/home** directory (and not the **/home/users/tdlmdz** login directory) as it is physically present on the machine you logged on (and thus minimizes disk access latencies).

1 Running the install.sh script

The first step consists in downloading the install.sh script 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/Distrib/install.sh
chmod +x install.sh
./install.sh
```

The default behavior of the `install.sh` script is to install the model using the `gfortran` compiler. On some local machines only an old version is available, so the script stops with a message that invites you to switch to another compiler. Set `compile=ifort` in the script and then run `./install.sh` again.

The script should then run smoothly without errors. If it isn't the case, immediately ask for some assistance. The script will end with messages of the likes of:

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

As the script runs, which takes a dozen minutes¹, you will see messages corresponding to the download of various elements via `wget` or informational messages from the compiler. The script ends by running a 5 days long test simulation on a regular 48×36-L19 grid.

¹Compilation LMDZ will be faster if one uses the `makelmdz_fcm` script rather than the default `makelmdz` script. To change this behaviour in the `install.sh` script find the instruction line `compile_with_fcm=0` and change it to `compile_with_fcm=1`.

As the automated installation proceeds, you can open a different terminal window and explore the downloaded directories and files. When `install.sh` runs, it creates a **LMDZtesting** directory in which you will find subdirectories **modipsl**, which contains the model, and **netcdf-4.0.1**, which contains the NetCDF library installed by `install.sh`. In **modipsl**, you will find directory **modeles**, which contains the **LMDZ5** directory. Go to this directory. Once the test bench simulation has been launched (the final step of the `install.sh` script), you will find a **BENCH48x36x19/** directory in which you can visualize the outputs of the run (even if the simulation is still running). Check out the contents of the directory and use your favorite software (Grads, Ferret,...) to browse the contents of the **histday.nc** file.

2 Switching to a different version of the model

The bench run that has been made used the latest *testing* version of the model, dated December 3rd 2014, which corresponds to *svn release* 2168. This information can moreover be obtained by checking at the last line of file

```
http://www.lmd.jussieu.fr/~lmdz/Distrib/LISMOI.testing
```

or on the web page about the testing versions

```
http://lmd.lmd.jussieu.fr/utilisateurs/distribution-du-modele/versions-intermediaires-en
```

or directly from within the LMDZ directory

```
cd LMDZtesting/modipsl/modeles/LMDZ5
svn info
```

With `svn`, one can change the model version by setting it to a more recent or older *svn release*, which we will now do here as an exercise since using the latest *testing* (the one you just installed) is the recommended way to proceed. To change version, go to directory **LMDZ5** and issue command:

```
svn update --revision 2116
```

to revert to *svn release* 2116 (a previous *testing*).

Still from the **LMDZ5** directory, run command `./makelmdz_fcm -arch local -d 48x36x19 -v true -j 8 gcm` (or alternatively `./makelmdz -arch local -d 48x36x19 -v true gcm`. Option `-v true` is not mandatory. It only indicates that we compile with Orchidee, but in fact it won't be used for the runs in this tutorial. Only some routines (those modified by the `svn update` and routines which depend thereupon) will be recompiled.

Once the model has been successfully recompiled, run a new simulation. To do so, create a new subdirectory in **LMDZ5** and copy boundary conditions, initial conditions and parameter files (**limit.nc**, **star*.nc**, ***.def**) over from directory **BENCH48x36x19**, along with the newly created **gcm.e** (if using `makelmdz`, the executable will be called **gcm.e** and created in the **LMDZ5** directory; if using `makelmdz_fcm` it will be called `gcm_48x36x19_phy_lmd_seq_orch.e` and will be found in the **LMDZ5/bin** directory).

Then run `./gcm.e > listing 2>&1` (the redirection of the model outputs in a text file is usually the best way to keep a trace of the run). Once the simulation is finished, you can compare, using the `diff` command, the simulation results (compare files **listing**, **hist*.nc** **restart.nc** and **restartphy.nc** from the two simulations). At the time of this tutorial, the changes between the latest *testing* versions impact on the results; fields will be numerically different from one case to the next.

Once this test finished, revert to the latest *testing* version of the model by going to directory **LMDZ5** and using:

```
svn update
```

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!). And don't forget to recompile the model after any `svn update`!

3 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 CO₂ concentration, etc. (check the LMDZ documentation and/or ask around to find out the meaning of the various available parameters).

Just run the simulation (in a different directory to avoid overwriting output files) and investigate the differences between simulations.

4 Changing the outputs

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

```
phys_out_filekeys=      y      y      n      y      n
phys_out_filenames=    histmth histday histhf  histins histLES
phys_out_filetimesteps= 5day   1day   1hr    6hr    6hr
phys_out_filelevels=   10     5     0     4     4
phys_out_filetypes=    ave(X)  ave(X)  ave(X) inst(X) inst(X)
```

With these settings, you have outputs in a `histmth.nc` file, which contains the average over the 5 days of the run and a `histday.nc` file which contains the daily averages (a 5 elements long time series).

You can find out which variables have been outputed by running the command

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

in the directory where the simulation was run.

You may also rerun the simulation but with a higher frequency of outputs for given variables (e.g. atmospheric temperature at 2m above the surface: `t2m`, surface pressure: `psol`, precipitation: `precip`, etc.), for instance by using the following settings in `config.def`:

```
phys_out_filekeys=      y      y      y      y      y
phys_out_filenames=    histmth histday histhf  histts  histhfm
phys_out_filetimesteps= 10day  1day   1hr    1ts    1hr
phys_out_filelevels=   10     5     0     0     0
phys_out_filetypes=    ave(X)  ave(X)  inst(X) inst(X)  ave(X)
flag_t2m=              10     10     1     1     1
flag_psol=             10     10     1     1     0
flag_precip=          10     10     0     1     1
```

5 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, where 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
and
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.F** in the **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-L19 ; which means you will have to recompile the model). Note that if you keep the same settings as for the 48×36-L19, case, your run will crash (try it!); at such resolution, one needs to use more time steps per day (set **day_step=480** and **iphysiq=10** in **gcm.def**), as can be seen in the sample **gcm.def** file in the **DefLists** directory.