Hands on tutorial session 1: First steps with the model

The LMDZ team

December 7, 2015

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/Distrib/TD1_en.pdf
```

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

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.

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
```

2 Running the install.sh script

If you're using a laptop provided for this training courses, you should find the `install.sh` in the main directory (LMDZ) ; otherwise, 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/Distrib/install.sh
chmod +x install.sh
./install.sh
```

Compilation takes around five minutes. Then, a test bench is downloaded (if not already on the disk) and a 5 days 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 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
############################################################
```

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

`install.sh` will check if some archives (LMDZ, NetCDF library, etc.) are on the disk (in the `~/LMDZ/Distrib` folder) and if not, will try to get them through the network using `wget` command. It will create the **LMDZ20151130.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 **LMDZ5** directory.

Once the test bench simulation will be launched (the final step of the **install.sh** script), you will also find a **LMDZ5/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 about eight minutes to complete the 5 days 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.

If you're using a laptop dedicated to the training, the required files/archives are already there. Otherwise, if downloads fail (slow network), please ask for some help to get them from a USB key.

The default behavior of the `install.sh` script is to install the model using the `gfortran` compiler, which is fine on the laptops provided for this training course, but on some local machines, an old version of the compiler only is available, so the script stops with a message that invites you to switch to another compiler. In any case, you can change from compiler by editing the script. For example, on CICLAD cluster, you may set `compilo=ifort` before launching `./install.sh`.

Several other automatic features of the script can be modified or unactivated just by editing few parameters in the file. For example, you can disable the NetCDF download and installation (first operation performed by `install.sh`) in case this library is already present on the computer you are using. The same kind of adjustments is possible for ORCHIDEE and LMDZ [1].

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       4       4
phys_out_filetypes=      ave(X)  ave(X)  ave(X) inst(X) inst(X)
phys_out_filetimesteps=  5day    1day    6hr     6hr     6hr
```

You will obtain 3 files (apart from ORCHIDEE related outputs): `histmth.nc`, `histday.nc` and `histhf.nc`, containing respectively 5 days (1 record for the whole run), daily and hourly averages.

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

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

Before relaunching the model, you can switch the ORCHIDEE model off by editing, in the bench directory, the `config.def` parameters file and setting `VEGET=n`.
Also, you need to remove some output files (in case you relaunch in the sale directory):

```
rm sechiba_out_*.nc sechiba_rest_out.nc
```

You may 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       y       y       y
phys_out_filenames=      histmth histday histhf  histts  histhfm
phys_out_filelevels=     10      5       4       4       4
```

[1] The LMDZ compilation embedded in **install.sh** 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**.

```
phys_out_filetypes=      ave(X)  ave(X)  ave(X)  inst(X) ave(X)
phys_out_filetimesteps=  5day    1day    1hr     1TS     1hr
flag_t2m=                10      10      5       5       5
flag_psol=               10      10      5       5       4
flag_precip=             10      10      4       5       5
```

Note that unlike other frequencies, 1TS (1 Time Step) is written in capital letters.
You can check in particular the **histts.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. (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 overwritting output files) and investigate the differences between simulations (using `diff` or your favorite visualization software).

5 Switching to a different version of the model

The recommended LMDZ version is the latest from the *testing* branch, but the `install.sh` script you have launched is set to download the latest version of the *trunk* branch of the model, containing recent developments some of you may need. Usually, it is safer to use the *testing* branch, much more extensively tested than the *trunk* branch. Your version is dated November 30th 2015, which corresponds to *svn release* 2402, possibly more if commissions have been made since this document revision. This information can moreover be obtained by checking at the last line of the file:

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

or directly from within the LMDZ directory:

```
cd ~/LMDZ/LMDZ20151130.trunk/modipsl/modeles/LMDZ5
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/LMDZ20151130.trunk/modipsl/modeles/LMDZ5'
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 update`. Then try again `svn info`.
Similarly, for *testing* versions, this information is available at the end of the file

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

Alternatively, same information is available on the following web page, dedicated to this branch:

```
http://lmdz.lmd.jussieu.fr/utilisateurs/distribution-du-modele/
versions-intermediaires/versions-intermediaires-ou-testing-en?set_language=en
```

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* 2396 (slightly older), go to directory **LMDZ5** and issue command:

```
svn update --revision 2396
```

From **LMDZ5** directory, run **./makelmdz_fcm -arch local -d 32x32x39 -v true -j 8 gcm** (or alternatively **./makelmdz -arch local -d 32x32x39 -v true gcm**).
Option **-v true** is not mandatory. It only indicates that we compile with Orchidee, but in fact, apart from the first bench, automatically launched by the `install.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 **LMDZ5** 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.
If you used **makelmdz**, it will be **LMDZ5/gcm.e**.
If you used **makelmdz_fcm**, it will be **LMDZ5/bin/gcm_32x32x39_phylmd_seq_orch.e**.

Then launch the run:

```
./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.

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). The two LMDZ versions you used are not very different, as illustrated by the relatively low number of source files that have been updated when you run *svn update*, but you can check it is already enough to explain that the numerical results differ.

Once this test finished, revert to the latest *trunk* 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). Don't forget to recompile the model after any svn update !

6 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 **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 !).