LDMZ tutorial: XIOS

LMDZ team

December 13th, 2017

This tutorial focuses on setting up, compiling and running LMDZ with XIOS.

This document can be downloaded as a pdf file:

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

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

This tutorial is for users who want to learn the basic steps needed to be able to run LMDZ with the XIOS input-output library on their computer. Note that this implies the prerequisite that you can run in parallel on your machine (i.e. that you have already completed the tutorial on running LMDZ in parallel).

1 Running the install_lmdz.sh script

The **install_lmdz.sh** script can download and compile the necessary libraries (NetCDF, IOIPSL, XIOS) and programs (ORCHIDEE and LMDZ), and runs a test simulation. All that is required is to specify the `-io xios` and `-parallel mpi_omp` (XIOS is designed to be used in parallel) options:

```
wget http://www.lmd.jussieu.fr/~lmdz/pub/install_lmdz.sh
chmod +x install_lmdz.sh
./install_lmdz.sh -parallel mpi_omp -xios -d 32x32x39
```

As with previous automated installations, you are encouraged to browse through the contents of sub-directory **LMDZ** (e.g. the **compile.sh**) and **BENCH32x32x39**.

Note that to compile LMDZ with XIOS, one must use the `-io xios` flag, e.g.:

```
./makelmdz_fcm -arch para -mem -parallel mpi -io xios -d 32x32x39 -j 8 gcm
```

As can be seen in the **compile.sh** script in the **LMDZ** directory.

Take the time to browse through the **xml** and **histday.nc** files in **BENCH32x32x39** to identify definitions and settings that were used. Note that the automated processing in the install script has renamed the executable **gcm.e**, but you may still find it in **LMDZ/bin**.

2 Running a first simulation with XIOS

Make a new simulation directory (e.g. by copying over all input **.def** and **.nc** files from the **BENCH32x32x39** directory[1]). In addition you will need the input **.xml** files to manage XIOS outputs.

For this first simulation, we will use XIOS in attached mode (i.e. embedded in LMDZ), so the **using_server** variable in **iodef.xml** must be set to **false**.
Moreover, to enable outputs via XIOS in LMDZ, note that the following flag:

```
ok_all_xml = y
```

must be set in the **config.def** file.

As can be seen in the **context_lmdz.xml** file, many predefined output files (mimicking what is done via the output.def file when using IOIPSL) are defined. By default only **histday.nc** is enabled. Modify file **file_def_histhf_lmdz.xml** by setting

[1] As in this example Orchidee is not used, the flag **VEGET** in **config.def** should be set to **n**

```
<file id="histhf" name="histhf" output_freq="6h" output_level="5"
type="one_file" enabled=".true.">
```

so that the **histhf.nc** file will also be generated (as a single file over the entire domain) when the model is run.

Then run the model "as usual", e.g. in MPI mode using 4 processes:

```
mpirun -np 4 gcm_32x32x39_phylmd_para_mem.e > listing 2>&1
```

And check the contents of the generated **histday.nc** and **histhf.nc** files.

3 Defining an additionnal output domain

One can ouput only a selected subset of the global domain by specifying the appropriate **domain** attributes in the **context_lmdz.xml** file. For example to ontput a 2x3 subdomain starting at grid indexes $i = 20$,$j = 15$ (C convention: index begining at 0):

```
<domain_definition>
    <domain id="dom_glo" data_dim="2" />
    <domain id="domain_zoom" domain_ref="dom_glo">
        <zoom_domain ni="2" ibegin="20" nj="3" jbegin="15" />
    </domain>
</domain_definition>
```

And also, still in the **context_lmdz.xml** file, define the corresponding grid in the `<grid_definition>` section:

```
<!-- Define Scalar grid for GHG, orbital parameters and solar constants -->
    <grid id="grid_scalar">
    </grid>

     <grid id="grid_zoom">
         <domain id="domain_zoom" />
     </grid>

     <grid id="grid_zoom_presnivs">
         <domain id="domain_zoom" />
         <axis id="presnivs" />
     </grid>
```

To test implementing this setup, let's assume you want to output at only one grid point, corresponding to Paris (longitude 49N ,latitude 2E) to compare model ouput to station records.

The first thing to do is to identify the grid coordinates that will have to be specified in the **domain** attributes. This can be done by inspecting the **lat** and **lon** values in the **histday.nc** file from the previous run, either via your favorite visualization software, or simply using the **ncdump** utility:

```
ncdump -fc -v lon histday.nc
ncdump -fc -v lat histday.nc
```

And adapt the **context_lmdz.xml** file accordingly.
Since we are interested in instantaneous values of for instance the **t2m** (temperature at 2m), **precip** (precipitation rates), **psol** (surface pressure) and **temp** (temperature profile) in the zoomed grid, it makes sense to define a new output file. One could either adapt the current **histins** file, or define a new one e.g. a **file_def_histinsParis_lmdz.xml** file:

```
<file_definition>
    <file_group id="defile">
        <file id="histinsParis" name="histinsParis"
              output_freq="1ts" output_level="5"
              type="one_file" enabled=".TRUE.">

            <!-- VARS 2D -->
```

```
            <field_group operation="instant" grid_ref="grid_zoom">
                <field field_ref="t2m" level="5" />
                <field field_ref="precip" level="5" />
                <field field_ref="psol" level="5" />
            </field_group>

            <!-- VARS 3D -->
            <field_group operation="instant"
                         grid_ref="grid_zoom_presnivs">
                <field field_ref="temp" level="5" />
            </field_group>
        </file>
    </file_group>
</file_definition>
```

And add this new definition file to the others specified in **context_lmdz.xml**:

```
    <file_definition src="./file_def_histinsParis_lmdz.xml"/>
```

Run the model and check the produced **histinsParis.nc** file.

4 Running in client-server mode

When running on a small number of cores, it is advised to use XIOS in "attached" mode. In multicore environments (i.e. more than 32) it can be more efficient to run in client-server mode and dedicate some cores to the XIOS server.

To test this setup, make a new directory where to run and copy over input files from previous simulation. Start by copying over the XIOS server **xios_server.exe** from **XIOS/bin**. Then adapt the **iodef.xml** to switch to client-server mode by setting the **using_server** variable to **true**. The executables may now be run, where the number of processes allocated to each is set via the **mpirun** command, for instance to run LMDZ on 3 processes and XIOS on 1:

```
mpirun -np 3 gcm_32x32x39_phylmd_para_mem.e > listing 2>&1 : -np 1 xios_server.exe
```

And check that you get the same output files as before.