



LMDZ outputs

Outline

- Introduction
- «history» files
- «restart» files
- controlling debug printing



LMDZ outputs

Introduction

LMDZ outputs include :

- «**history**» files : they gather instantaneous or averaged diagnostic variables
- «**restart**» files : used to restart or extend a simulation
- the «**output**» file: collects all control and error messages



LMDZ outputs

Introduction:

The «history» and «restart» files in LMDZ are all in **NetCDF** format and written using either the **NetCDF** or **IOIPSL** libraries.

The **IOIPSL** library was developed at IPSL for model I/Os using the **NetCDF** library. For «history» files, **IOIPSL** allows variables to be « manipulated » (e.g. average/max/min) before being written out.

The output of variables consists in 2 steps :

- definition of the variables to output (during initialisation of the run)
- computation and writing of the variables (during the simulation)

To help with the debugging of the code, there is also a mechanism which writes the variables in the **GrADS** format.

Further information : the **XIOS** library can already be used in lieu of the **IOIPSL** library



LMDZ outputs

Introduction:

The partition between the dynamical module and the physical module in the LMDz code and the fact that one can be run without the other implies that both modules need to have their own «**restart**» and «**history**» outputs :

- «**restart.nc**» for the dynamics module and «**restartphy.nc**» for physics module
- «**history**» file for the dynamics module only consists of the variables of state (U, V, T, Q, Ps) at two frequencies : instantaneous and averaged.

When run in parallel mode, each process writes its own history files in its domain. The global file is reconstructed from these various files by using the **rebuild** utility distributed with the **IOIPSL** library



LMDZ outputs

«history» outputs of the physics module :

- Scheme which allows to individually control the output of the variables in 9 files :
 - 5 basic files : *hismth.nc, histday.nc, histhf.nc, histins.nc, histLES.nc*
 - 1 specific file (data sites) : *histstn.nc*
 - 3 files with predefined pressure levels : *hismthNMC.nc, histdayNMC.nc, histhfNMC.nc*
- outputs on standard levels pressures :*
1000., 925., 850., 700., 600., 500., 400., 300., 250., 200., 150., 100., 70., 50., 30., 20., 10.hPa

(names of files can be changed easily)

- There are 3 specific «history» outputs files :
hismthCOSP.nc, histdayCOSP.nc, histhfCOSP.nc
Outputs from the COSP simulator



LMDZ outputs

«history» outputs in the physics module (principle) :

- To each «history» file "histX.nc", an output level, **lev_histX**, is associated
- To each variable, an output level **flag_D** is associated

Then if

flag_D ≤ **lev_histX** ==> variable "D" is defined and written in the file "histX.nc"

- These control keys are defined in the files **config.def/output.def**
- Each output file can be controlled to :
 - activate it
 - define its name,
 - define its output frequency,
 - define the mathematical operation on the output variables (average, max/min ...)
 - output the variables on the whole or a limited domain



LMDZ outputs

«history» outputs in physics module (in practice) :

In the **config.def** or **output.def**, file :

✓ define the control keys of the various files :

```
# Activate or not the output of the file :
phys_out_filekeys=      y      y      n      y      n      n      n      n      n
# Name of files :
phys_out_filenames=  histmth  histday  histhf  histins  .....  histhfNMC.nc
# Output level of files
phys_out_filelevels=   5        2        2        4        .....        5
# Archive operation
phys_out_filetypes=   ave(X)   ave(X)   ave(X)   inst(X)  .....  inst(X)
# Frequence (months, mth, mois, day, days, jour, jours, heure, mn, TS, ... )
phys_out_filetimesteps= 30day   1day    6hr     1TS     .....    6hr
```

✓ define the control keys of each variable :

```
# Vertical wind
flag_vitw   =      2      3      7      6      .....    10
name_vitw   =  vitw
```



LMDZ outputs

«history» outputs of physics module (in practice):

In the **config.def** or **output.def** file :

Names of files :

```
phys_out_filenames = histmth.nc  histday.nc  histhf.nc  histins.nc  .....
```

Activate or note the output on a limited domain

```
reg_out_fkeys =          n          n          y          n          .....
```

Longitude min and max of domain

```
phys_out_lonmin =      -180      -180          0      -180      .....
```

```
phys_out_lonmax =      +180      +180          90      +180      .....
```

Latitude min and max of domain

```
phys_out_latmin =      -90      -90      -30      -90      .....
```

```
phys_out_latmax =      +90      +90      +40      +90      .....
```

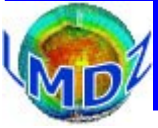
Vertical level min and max

```
phys_out_levmin=        1          1          1          1      .....
```

```
phys_out_levmax=       39          39          3          39      .....
```

›Number of variables controlled by this mechanism (approx.)

300 2D-fields and 150 3D-fields



LMDZ outputs

«history» outputs of physics module

How to add a variable ?

2 routines need to be modified:

- **libf/physlmd/phys_output_ctrlout.F90** :
 - declaration of name, description, unit and output level of the new variable
- **libf/physlmd/phys_output_write.F90** :
 - write the new variable in each file

Example for «t2m_min» variable :

.../libf/physlmd/phys_output_ctrlout.F90

```
type(ctrl_out),save :: o_t2m_min = ctrl_out((/ 1,1,1,5,10,10,5,5,5 /), &  
      't2m_min', 'Temp 2m min', 'K', (/ 't_min(X)', 't_min(X)', 't_min(X)', &  
      't_min(X)', 't_min(X)', 't_min(X)', 't_min(X)', 't_min(X)' /))
```

.../libf/physlmd/phys_output_write.F90

```
CALL histwrite_phy(o_t2m_min, zt2m)
```



LMDZ outputs

«history» outputs of physics module (specific files):

- The same mechanism applies to the COSP simulator

- Routines :

`.../libf/cosp/cosp_output_mod.F90`

`.../libf/cosp/cosp_output_write_mod.F90`



LMDZ outputs

The XIOS library :

IOIPSL is progressively replaced by XIOS

XIOS : XML – IO – SERVER

Library developed at LSCE/IPSL for management of IO of climate codes

Based on client – server principle : IO server manages the outputs so that the climate code does not waste time on its outputs, it just sends them to the IO server

All aspects of the outputs (name, units, post-processing frequencies, operations, ...) are controlled by external xml files.



LMDZ outputs

The XIOS library :

Advantages of **XIOS** library :

- Client – server model : code not slowed down by outputs
- Flexible
- Minimal modification to the simulation code
- No recompilation of code necessary when changing IO definitions (all done in xml files)
- Particularly geared towards parallel code (no more rebuilding output files, asynchronous writes ...)
- XML and its concept of inheritance

« Disadvantage » :

- Another library to install



LMDZ outputs

XIOS in LMDZ :

In working order but not yet a standard. Is the default output mode for the new IPSLCM6 configuration.

Use :

Need to download XIOS from <http://forge.ipsl.jussieu.fr/ioserver/> and compile it

LMDZ compile :

3 options to the makelmdz/makelmdz_fcm command :

- **-io ioipsl** only IOIPSL
- **-io mix** both IOIPSL and XIOS
- **-io xios** only XIOS

ok_all_xml = y in run.def file to get xml to control everything

Sample xml files in DefLists directory



LMDZ outputs

«restart» files:

The dynamical and physical modules each write their own restart file (**restart.nc** et **restartphy.nc**). These files save the state variables that the model needs at each time step so that the model can be restarted without losing continuity (in practical terms this is known as «**1+1=2**»)

Routines involved in this process are :

- **.../libf/dyn3d/dynredem.F** for the dynamical module (the restart state being read in by **.../libf/dyn3d/dynetat0.F**)
- **.../libf/physlmd/phyredem.F** for the physics module (corresponding routine in **.../libf/physlmd/phyetat0.F**)

These routines do not use the **IOIPSL** library and are interfaced directly with the **NetCDF** library.



LMDZ outputs

Controlling output messages :

Most of control outputs and messages are written to standard output (the screen) by the use of commands such as:

```
print*, ... or write(*,*) 'ma variable =',... .
```

A mechanism exists to output these messages to a file rather than the screen.

One just needs :

- to include in any new routine, the `iniprint.h` file which defines and shares 2 parameters :

- **lunout** : a unit number corresponding to the output file
(if `lunout` \neq 6 ==> a `lmdz.out` file is created and assigned to this number)
- **prt_level** : an output level

The value of these two parameters can then be modified in the `run.def` file

- To use them in the routine you then just need to add lines such as :

```
IF (prt_level>9) WRITE(lunout,*) 'pas de convection'
```

While keeping small values of **prt_level** for really important messages