LMDZ outputs

LMDZ Training, December 2024, A. Idelkadi

Introduction

LMDZ outputs include:

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

LMDZ outputs

Outline

- Introduction
- «history» files
- «restart» files
- file of control printing and error messages
- specific output files

LMDZ outputs

Introduction

IOIPSL / XIOS libraries:

- LMDZ «history» and «restart» files are in **NetCDF** format and written using either :
  - directly the **NetCDF** library or
  - **IOIPSL/XIOS** libraries : developped at *IPSL* for model I/Os using the NetCDF library.

- 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 ⇒ both modules need to have their own «restart» and «history» outputs

  - **« restart »** files of dynamical and physical modules are **written by using NetCDF directly**

  - «history» files of dynamical and physical modules are **written by using IOIPSL or XIOS**

    - For **«history» of dynamical** module, it only consists of the variables of state U, V, T, Q, Ps (rarely used)
      ⇒ **« history » of physical part are most often used**

    - For « history » files, 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)

    - In parallel mode,
      - with **IOIPSL :** each process writes its own history file in its domain. Then, global file is reconstructed from these various files by using the ***rebuild*** utility distributed with the IOIPSL library.
      - with **XIOS**, this rebuilding mechanism is automatically provided.

LMDZ outputs

«history» outputs of the physical module:

Generalities:

- **10 « history » output files :**
  same scheme wich allows to control individually the output of the variables in 10 files

  - ➔ **5 basic files :**
    monthly/daily/high frequency - average/instantanous

  - ➔ 5 output files with particular uses :
    - ➢ ***histstn.nc*** ⇒ **data sites** (requested by CMIP)
    - ➢ 3 files : ***histmthNMC.nc, histdayNMC.nc, histhfNMC.nc*** *(requested by CMIP)* ⇒ ***outputs interpolated on 17 standard levels pressures (in hPa):***
      1000., 925., 850., 700., .600., 500., 400., 300., 250., 200., 150., 100., 70., 50., 30., 20., 10**.**
    - ➢ ***histstrataer.nc*** ⇒ **Outputs of stratospheric aerosols**

- **Variables :**
  - ➔ **2D are output on the horizontal grid of the model (longitude, latitude)**
  - ➔ **3D are output on a 3D grid (longitude,latitude,vertical level)**

LMDZ outputs

«history» outputs of the physical module:

Generalities:

- **Vertical coordinate in LMDZ:**

  - **The hybrid coordinate system is used for the vertical in LMDZ.**

  - **Principle ⇒ to define it implicitly by giving the pressure in layer l as: p(l) = ap(l) + bp(l)*ps(1)**

    - **ap and bp coefficients are chosen so that :**
      - the model levels follow the relief near the surface (p(l)=bp(l)*ps : ap~0 and bp~1)
      - and are close to the isobaric levels at higher altitudes (ap~0 and bp~0 with ap>>bp*ps)

    - **distribution of vertical layers is irregular, in order to have**
      - a finer discretization for the atmospheric boundary layer
      - and a coarser discretization for higher altitudes.

⇒ More detail in « Dynamics » presentation (Ehouarn)

LMDZ outputs

«history» outputs of the physical module

Control of output files and variables:

- 4 types of keys :

  1) Flags to control how many and which variables to write in output files

  2) Flags to control time frequency and mathematical operation to write in output files

  3) Flags to control the activation of output of files, to change name of files and variables

  4) Flags to activate and define the output on a limited domain

- Keys can be controlled :

  ➔ with **IOIPSL** in **def** files (**config.def** or **output.def)**

  ➔ with **XIOS** in **xml** files

  ⇒ Sample def and xml files are in DefLists directory :
  ../modipsl/modeles/LMDZ/**DefLists**

LMDZ outputs

«history» outputs of the physical module

Principle of controlling of how many and which variables to write in the output files

- 2 flags

Flag to control the output flow of data in each file
fileN → **outlev_fileN** (integer) N=1, ..,10

variable "V" → **outlev_V_fileN** (integers) N=1, …,10
Flag to control the output of variable V in each file

outlev_V_fileN ≤ outlev_fileN ⇒ variable V is defined and written in fileN

- Default values :

✓ ***outlev_file(1:10)***	⇒	***2***	***1***	***1***	***1***	***1***	***1***	***5***	***5***	***5***	***5***	**(../phylmd/phys_output_mod.F90)**
✓ ***outlev_tsol***	⇒	***1***	***1***	***1***	***5***	***10***	***10***	***11***	***11***	***11***	***11***	**(../phylmd/phys_output_ctrl.F90)**

- Keys can be controlled :
  - ✓ With IOIPSL in **config.def** or **output.def:**

```
phys_out_filelevels  =   5   1   1   1   1   5   5   5   5   5
flag_tsol            =   1   2   3   4   5   11  11  11  11  5
```

  - ✓ With XIOS in file_histday_lmdz.xml :

```
<file_definition>
  …
    <file id="histday" ... output_freq="1d" output_level="2">
        <field field_ref="tsol" level="3" />
```

LMDZ outputs

«history» outputs of the physical module

Control of time frequency and mathematical operation used to write in output files :

- 2 flags :

  Flag to control the time frequency of each output file :
  (mounthly, daily, hourly, physical time step, ...)

  Flag to control mathematical operations used to archive variables :
  (instantaneous, average , min, max)

- Default values for output files in **../phylmd/phys_output_mod.F90** :

✓	***output time frequency*** ⇒	*1month*	*1day*	*3hours*	*30mn*	*3hours*	*30mn*	*1month*	*1day*	*6hours*	*1month*
✓	***archive operation*** ⇒	*ave(X)*	*ave(X)*	*inst(X)*	*inst(X)*	*ave(X)*	*inst(X)*	*inst(X)*	*inst(X)*	*inst(X)*	*ave(X)*

- Keys can be controlled :
  - ✓ With IOIPSL in **config.def** or **output.def:**

```
phys_out_filetimesteps =   1mth      1day       6h       3h       1TS         ...
phys_out_filetypes         =   ave(X)     ave(X)     ave(X)    inst(X)   inst(X)     ...
```

  - ✓ With XIOS in file_histmth_lmdz.xml :

```
<file_definition>
   <file_group id="defile">
     <file id="histmth" output_freq="1m" output_level="2" …
       <field_group operation="average" ...
```

LMDZ outputs

«history» outputs of the physical module

Control of activation of output of files / change the name of output files and variables:

- 3 flags :

  flag used to activate or not the output of files

  flag used to change the name of output files

  flag used to change the name of output variable

- Default values in **../phylmd/phys_output_mod.F90** :
  - *activation of files* ⇒ **y** ***n n n n n n n n n***
  - *name of files* ⇒ ***histmth histday histhf6h histhf3h histhf3hm histstn histmthNMC histdayNMC histhfNMC histstratae***
  - Names of variables ⇒ default values in **../phylmd/phys_output_ctrl.F90** :

- Keys can be controlled :
  - With IOIPSL in **config.def** **or** **output.def:**

```
phys_out_filekeys    =     y         y            n            y            ...
phys_out_filenames =  histmth     histday      histhf       histins        ...
name_tsol                 =      ts           ts            ts           ts            ...
```

  - With XIOS in file_histhf_lmdz.xml

```
<file_definition>
   <file_group id="defile">
      <file id="histhf" name="histhf" output_freq="3h" output_level="2" enabled="TRUE">
          <field_group operation="average" …
                                       <field field_ref="tsol" name="ts" level="1" />
```

LMDZ outputs

«history» outputs of the physical module

Control of output on a limited domain :

- **With IOIPSL in config.def/outputs.def:**
  - Flag to activate or note the output on limited domain for each file :

```
phys_out_regfkey   =       n                 n              y                 n                …
```

  - Flags to define the limited horizontal domain for each file :

```
phys_out_lonmin =        -180            -180          0               -180           ...
phys_out_lonmax =        +180            +180         90               +180           ...
phys_out_latmin =        -90              -90         -30               -90           ...
phys_out_latmax =        +90              +90         +40               +90           ...
```

  - Flag to define the limited vertical axis for each file :

```
phys_out_levmin=          1                1            1                1            ...
phys_out_levmax=         39               39            3               39            ...
```

  - *Default values in* **../phylmd/phys_output_mod.F90**

- **With XIOS in context_lmdz.xml**

```
<domain_definition>
  <domain id="dom_glo" data_dim="2" />
  <domain id="dom_MyRegion"  domain_ref="dom_glo">
    <zoom_domain id="dom_MyRegion" ibegin="12" jbegin="14" ni="8" nj="10" />
  </domain>
 </domain_definition>
```

LMDZ outputs

«history» outputs of the physical module :

In practice, with IOIPSL:

- No output flag in def files, by default ⇒
  - ✓ output of a single ***histmth.nc*** file
  - ✓ with a minimum set of variables
  - ✓ ***monthly averages***

- In **config.def** or **output.def**, file we can define the control keys of output files :

```
# Flag to change the name of files:
phys_out_filenames=histmth  histday  histhfh  histins  histhf3hm  histstn  histmthNMC  histdayNMC  histhfNMC histstrataer
```

basic files ⟷ specific output files

```
#change the name only for 5 first files
phys_out_filenames = histmth  histday  histhf6h  histins  histhf3hm

#change the name only for 3 first files (possibility of separating with , )
phys_out_filenames = histmth,  histday,  histhf6h

#change the name only for the first file
phys_out_filenames = histmth

#change only for seconde file
phys_out_filenames__2 = histday

#control of activation or not of the seconde file
phys_out_filekeys__2   =    y
...
```

LMDZ outputs

«history» outputs of the physical module :

In practice, with IOIPSL:

In **config.def** or **output.def**, file ⇒ we can change the control keys of output files :

# Activate or not the output of the file ⇒ **y (yes), n (no)**

```
phys_out_filekeys    =    y     y     n     y     n
```

# Name of files :

```
phys_out_filenames  =  histmth   histday   histhf6h   histhf   histins
```

# *Time frequency of files* ⇒ ***mth (month), day, hr (hour), TS (time step of physics), ...***

```
phys_out_filetimesteps =  1mth    1day    6hr    3hr    1TS
```

# Archive operation ⇒ ***inst(X)* (instantaneous), *ave(X)* (average), *t_min(X)* (min), *t_max(X)* (max)**

```
phys_out_filetypes   =    ave(X)    ave(X)    ave(X)    inst(X)    inst(X)
```

# Output level of files ⇒ ***0*, *1*, …**

```
phys_out_filelevels   =     5       2       2       1       1
```

# writing of variable in output files : *0*, *1*, …

```
flag_tsol      =            2       3       7       10      10
```

Change the name of variable in output files :

```
name_tsol      =            ts      ts      ts      ts      ts
```

Q1 : In wich output files tsol will be written ?

Q2 : If you want to output tsol in histday.nc file how to do it ?

LMDZ outputs

«history» outputs of the physical module:

In practice with IOIPSL:

Control the output of variables on a limited domain :

```
phys_out_filekeys    =     n          n          y          n
phys_out_filenames  =  histmth    histday    histhf     histins

# Activate or note the output on a lomited domain
phys_out_regfkey =        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 controled by this mechanism (approx.) :

+550 2D-fields and +300 3D-fields

LMDZ outputs

«history» outputs of the physical module

With XIOS library :

- **XIOS** (XML–IO–SERVER) : more developped and progressivly replaces IOIPSL
  - ➔ 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, pot-processing frequencies, …) are **controlled by external xml files**

- To run LMDZ with XIOS, you need to :
  - ➔ Download XIOS from http://forge.ipsl.jussieu.fr/ioserver/XIOS
  - ➔ Compile LMDZ with XIOS :
    3 options to the makelmdz/makelmdz_fcm command :

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

  - ➔ To put : ok_all_xml = y in run.def file to get xml to control everything

LMDZ outputs

«history» outputs of the physical module

With XIOS, in practice :

- **Xml file :** one file, at least must be present (model will crash if this file is absent) ⇒ **iodef.xml**

```
<simulation>
  <context id="xios">
    <variable_definition>
         <variable_group id="server">
            <variable id="using_server2" type="bool">false</variable>
            …..
         </variable_group>
         <variable_group id="buffer">
            …….
         </variable_group>
         <variable_group id="parameters" >
          <variable id="using_server" type="bool">false</variable>
          <variable id="info_level" type="int">1</variable>
          …..
         </variable_group>
       ….
    </variable_definition>
  </context>
  <context id="LMDZ" src="./context_lmdz.xml"/>
</simulation>
```

LMDZ outputs

«history» outputs of the physical module

Xml filesWith XIOS, in practice :

- XML files ⇒ context_lmdz.xml :

```
<! Define available variables >
<field_definition src="./field_def_lmdz.xml"/>

<! Define output files >
  <file_definition src="./file_def_histday_lmdz.xml"/>
  ...

  <! Define domains and groups of domains >
  <domain_definition>
    <domain id="dom_glo" data_dim="2" />
  </domain_definition>

  <! Define groups of vertical axes >
  <axis_definition>
    <axis id="presnivs" standard_name="Vertical levels" unit="Pa"/>
    ...
  </axis_definition>

  <! Define grids >
  <grid id="grid_glo_presnivs">
     <domain domain_ref="dom_glo" />
     <axis axis_ref="presnivs" />
  </grid>

</context>
```

LMDZ outputs

«history» outputs of the physical module

With XIOS in practice :Xml files:

- Xml files ⇒   field_def_lmdz.xml

```
<field_definition level="1" prec="4" operation="average" freq_op="1ts" enabled=".true."
default_value="9.96921e+36">

    <field_group id="fields_2D" domain_ref="dom_glo">
        <field id="phis"    long_name="Surface geop.height"     unit="m2/s2" />
        <field id="ffonte"    long_name="Thermal flux for snow melting"     unit="W/m2" />
        ...
    </field_group>
    <field_group id="fields_3D" domain_ref="dom_glo" axis_ref="presnivs">
        <field id="tke"    long_name="TKE"    unit="m2/s2" />
        ...
    </field_group>

    <field_group  id="fields_NMC" domain_ref="dom_glo" axis_ref="plev">
      <field id="ta" long_name="Air temperature" unit="K" />
      ...
    </field_group>
      ...
    <field_group id="fields_COSP_CALIPSO" domain_ref="dom_glo" freq_op="3h">
      <field id="cllcalipso"    long_name="Lidar Lowlevel Cloud Fraction"   unit="1" />
      ...
    </field_group>
</field_definition>
```

LMDZ outputs

«history» outputs of the physical module

With XIOS, in practice :

- **XML files ⇒ file_def_histday_lmdz.xml :**

```
<file_definition>
    <file_group id="defile">
        <file id="histday" name="histday" output_freq="1d" output_level="2" enabled="TRUE">
            <! VARS 2D >
            <field_group operation="average" freq_op="1ts">
                <field field_ref="phis" level="1" />
                ...
                <field field_ref="ffonte" level="10" />
                ...
                <field_group operation="average" freq_op="1ts" detect_missing_value=".true.">
                    <field field_ref="u850" level="7" />
                    ...
                </field_group>
            </field_group>
            <! VARS 3D >
            <field_group operation="average" freq_op="1ts" axis_ref="presnivs">
                <field field_ref="cldtau" level="5" />
            </field_group>
        </file>
    </file_group>
</file_definition>
```

LMDZ outputs

«history» outputs of the physical module:

How to add a new output variable ?

- 2 routines need to be modified:
  - **libf/phylmd/phys_output_ctrlout.F90** ⇒ declaration of new variable : output level, name, description, unit, archiving operation
  - **libf/phylmd/phys_output_write.F90** ⇒ write the new variable in each file

- Example for «my_new_var» variable :
  - Declaration of variable in **…/libf/phylmd/phys_output_ctrlout.F90**

```
type(ctrl_out),save :: o_my_new_var   =   ctrl_out((/ 1,1,1,5,10,11,11,11,11,11/), &
                                          'my_new_var','My New field', 'K',  &
                                          (/ 'ave(X)', 'ave(X)', 'ave(X)', ave(X), inst(X), &
                                          'inst(X)','inst(X)','inst(X)','inst(X)','ave(X)' /))
```

  - Write a new variable in **…/libf/phylmd/phys_output_write.F90**

```
call histwrite_phy(o_t2m_min, znewvar)
```

znwvar : variable calculated in the code that corresponds to my_new_var

- To add variable to the XIOS output files ⇒ **you also need to add them to the xml files** (necessarily in field_lmdz.xml)

LMDZ outputs

«Restart» files :

- "restarts" files contain the final state of the simulation ⇒
  used **to extend this simulation or to restart an other new one**

- The dynamical and physical modules each write their own restart file (**restart.nc** and **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 :
  - ➔ for the dynamical module :
    - ✓ `…/libf/dyn3d/dynredem.F90` / `…/libf/dyn3dmem/dynredem_mod.F90`
    - ✓ `…/libf/dyn3d/dynetat0.F90` / `…/libf/dyn3dmem/dynetat0_loc.F90`
  - ➔ for the physical module :
    - ✓ `…/libf/phylmd/phyredem.F90` / `…/libf/phylmd/phyredem.F90`
    - ✓ `…/libf/phylmd/phyetat0.F90` / `…/libf/phylmd/phyetat0_mod.F90`

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

LMDZ outputs

files of control printing and error messages:

- **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 cleaner mechanism exists to output these messages to a file rather than the screen.

  ✔ to include in any new routine, the iniprint.h file : **#include iniprint.h**
  in iniprint.h are defined 2 parameters :
  - lunout : a unit number corresponding to the output file
  (if lunout ≠ 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

LMDZ outputs

Output file of control printing and error messages:

- **What use are they ?**

  If the model crashes or does not seem to run properly, these output messages should give you an indication of what’s going on.

  - The first thing to do is : **grep ‘Houston, we have a problem ’ output_file**
    As the model will output this string with an indication of the problem it encountered, when the problem has been anticipated by the developers (might be a configuration problem, a temperature that’s suddenly out of range, …)

  - If the string is not found and no obvious error (**segmentation fault, memory violation, floating point exception**) can be found in the output messages ⇒
    - to recompile the model with the **-debug** option
    - and run it again.

      ⇒ It should now give you an indication of :
      - the line
      - the routine that is causing the crash.

        ⇒ Once found, you can :
        - start debugging the routine or
        - call for help with some vital information.

LMDZ outputs

Output specific files :

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

- **«history» output files of COSP** (CFMIP Observation Simulator Package) :
  3 output files (monthly, daily and HF) with a lot of cloud diagnostics

- **Output files with variables in the GrADS format :**
  to help with the debugging of the code, there is also a mechanism which writes the variables in the GrADS format by using directly NetCDF library

- **Files with all the control parameters used for the simulation :**
  a mechanism to write files with all the control parameters effectivly read and used for the simulation : used_run.def, used_config.def, used_physiq.def, ...

- **CMIP6 IPSL workflow :**
  with XIOS, LMDZ outputs files containing single timeseries of requested variables of CMIP

- **Alert messages in output of control printing file or in file ALERTES.txt :**
  We have introduced a routine called prt_alerte_mod.F90 to print informative alert messages ⇒ see for information: https://lmdz-forge.lmd.jussieu.fr/mediawiki/LMDZPedia/index.php/HowTo:_Print_alert_messages

LMDZ outputs

Output specific files :

« history » output files of COSP

- COSP : CFMIP Observation Simulator Package
  - simulator : a diagnostic code that computes pseudosatellite observations from model variables in order to compare them to real satellite observations
  - Cosp implemented in LMDZ to evaluate the representation of cloud process in the models **(https://lmdz.lmd.jussieu.fr/Members/aidelkadi/cosp)**

  - COSP has simulators for these satellite cloud products:
    ISCCP, CALIPSO, CLOUDSAT, PARASOL, MISR, MODIS

- 2 versions of COSP implemented in LMDZ model :
  - ***cospv1*** (used for CMIP6 exercise)
  - ***cospv2*** (new version of cosp with more diagnostics)

  - Cosp routines in directory :
    - **Cospv1** : ../***phylmd/cosp***
    - ***Cosp*****v2** : ***…/phylmd/cospv2***

    Cosp output routines :

    cospv1: cosp_output_mod.F90 and cosp_output_write_mod.F90
    cospv2: lmdz_cosp_output_mod.F90 and lmdz_cosp_output_write_mod.F90

- To run LMDZ simulation with COSP simulator :
  - Compile LMDZ model with option : **-cosp none/v1/v2**
  - Activate COSP in LMDZ simulation : **ok_cosp=y** in config.def file
  - ***cosp_input.txt*** & ***cosp_output.txt*** : Cosp namelist files to control simulators & outputs

LMDZ outputs

Output specific files :

« history » output files of COSP

The same philosophy, as described before for *LMDZ history files,* is used with IOIPSL and XIOS:

- 3 outputs files for COSP: ***histmthCOSP.nc, histdayCOSP.nc, histhfCOSP.nc***
  Lot of cloud diagnostics : low, mid, hight level, total, vertical distribution of cloud fraction, …

- The control keys of files and variables :
  - **With IOIPSL library:**

cosp_outfilenames = ***histmthCOSP.nc, histdayCOSP.nc, histhfCOSP.nc***			
cosp_outfilekeys =	**y**	**y**	**n**
cosp_ecritfiles =	**1mth**	**1day**	**3h**
cosp_outfiletypes =	**ave(X)**	**ave(X)**	**inst(X)**
cles_cllcalipso =	**y**	**y**	**n**
name_cllcalipso =	**LowCldMth**	**LowCldDay**	**LowCldHf**

  - **With XIOS library:**
    Variables defined in XML file: **field_def_cosp1.xml / field_def_cospv2.xml**
    Output files defined in XML files:
    file_def_histmthCOSP_lmdz.xml / file_def_histmthCOSPv2_lmdz.xml
    file_def_histdayCOSP_lmdz.xml / file_def_histmthCOSPv2_lmdz.xml
    file_def_histhfCOSP_lmdz.xml / file_def_histhfCOSPv2_lmdz.xml

- **Number of Cosp output variables (approx.): +70(cospv1) +100(cospv2)**