HyMeX

REMEMBER

ANR

Introducing LMDZ physical schemes in WRF

L. Fita[1], F. Hourdin, L. Farihead

[1] *Laboratorie de Météorologie Dynamique, IPSL, Paris-6, UPMC, Tr 44-45 2nd fl., Jussieu, 75005, Paris, France*

–

LMD

Contact: **lluis.fita@lmd.jussieu.fr**

Introduction

- LMDZ has a hydrostatic dynamical core

Introduction

- LMDZ has a hydrostatic dynamical core
- LMDZ has a set of physics prepared to run in a GCM

Introduction

- LMDZ has a hydrostatic dynamical core
- LMDZ has a set of physics prepared to run in a GCM
- WRF has a primitive equation non-hydrostatic fully compressible dynamical core

Introduction

- LMDZ has a hydrostatic dynamical core
- LMDZ has a set of physics prepared to run in a GCM
- WRF has a primitive equation non-hydrostatic fully compressible dynamical core
- WRF has a set of physics prepared to run at very high resolutions ($\gtrsim 0.5km$)

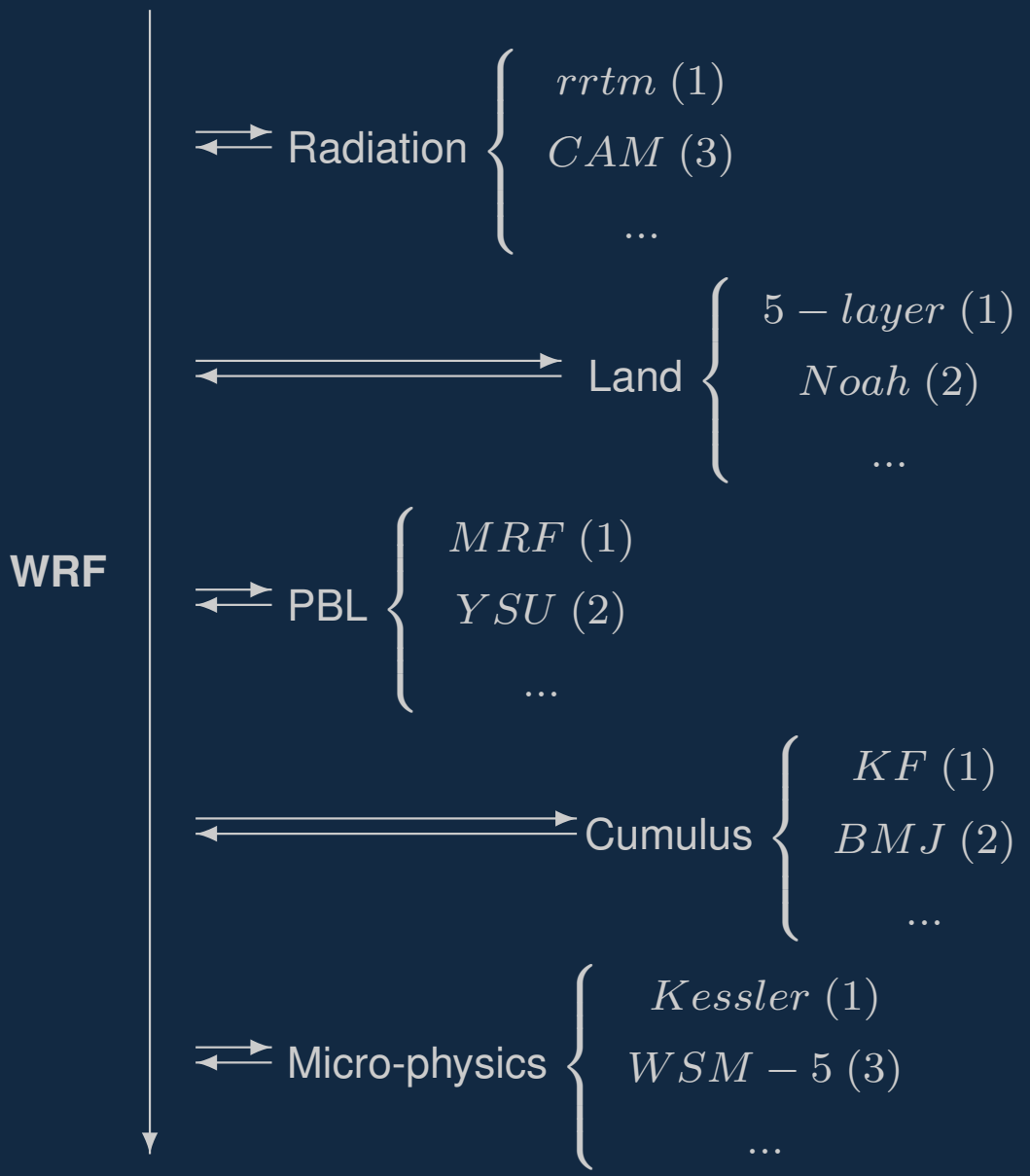
Introduction

- LMDZ has a hydrostatic dynamical core
- LMDZ has a set of physics prepared to run in a GCM
- WRF has a primitive equation non-hydrostatic fully compressible dynamical core
- WRF has a set of physics prepared to run at very high resolutions ($\gtrsim 0.5km$)
- Include LMDZ physical schemes in WRF in order to:

Introduction

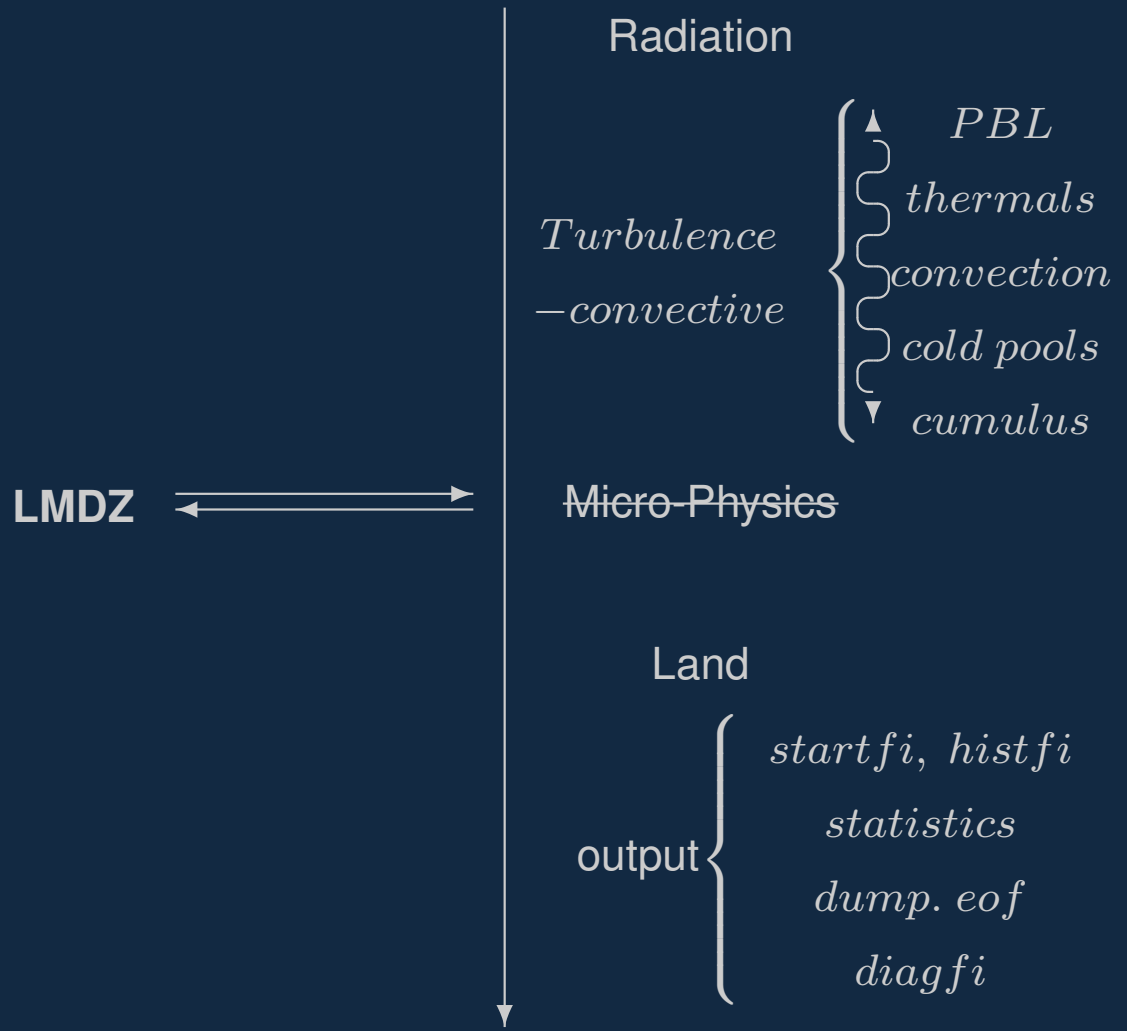
- LMDZ has a hydrostatic dynamical core
- LMDZ has a set of physics prepared to run in a GCM
- WRF has a primitive equation non-hydrostatic fully compressible dynamical core
- WRF has a set of physics prepared to run at very high resolutions ($\gtrsim 0.5km$)
- Include LMDZ physical schemes in WRF in order to:
  - Analyze parameterized/explicitly resolved results with the same platform
  - Analyze the role of a non-hydrostatic dynamics in a GCM/RCM run
  - Analyze the performance of the LMDZ physics at 'high' resolution

Models structure



- Model 'call' each scheme consecutively
- Multiple versions of each scheme
- Flexibility of combination of schemes
- Inconsistent combinations!

Models structure



- Robust call of physics schemes
- Computation of climatic values: daily/monthly means
- Single *'turbulence-convective'* scheme

Technical differences

- WRF code is very flexible:
  - fully modular: no mixture of dynamics, physics, i/o
  - fully F90 code: pointer, data-structures, namelist, ...
  - all variables, domain dimensions and data is kept in a Fortran data structure called `grid`, which is managed with an ASCII file called `Registry`

Technical differences

- WRF code is very flexible:
  - fully modular: no mixture of dynamics, physics, i/o
  - fully F90 code: pointer, data-structures, namelist, ...
  - all variables, domain dimensions and data is kept in a Fortran data structure called `grid`, which is managed with an ASCII file called `Registry`
- LMDZ code is less flexible:
  - not fully modular: mixture of physics & i/o
  - F77 & F90 code: static grid dimensions, use of 'SAVE'

Challenges

1. Perform a flexible introduction of LMDZ physics in WRF

Challenges

1. Perform a flexible introduction of LMDZ physics in WRF
2. Perform an *'easy to use'* implementation

Challenges

1. Perform a flexible introduction of LMDZ physics in WRF
2. Perform an *'easy to use'* implementation
3. Perform an *'easy to update'* (both models) implementation

Challenges

1. Perform a flexible introduction of LMDZ physics in WRF
2. Perform an *'easy to use'* implementation
3. Perform an *'easy to update'* (both models) implementation
4. Technical aspects:
   - Minimal changes in LMDZ code
   - Use of WRF compilation structure/framework

Challenges

1. Perform a flexible introduction of LMDZ physics in WRF
2. Perform an *'easy to use'* implementation
3. Perform an *'easy to update'* (both models) implementation
4. Technical aspects:
   - Minimal changes in LMDZ code
   - Use of WRF compilation structure/framework
5. Usability aspects:
   - Use LMDZ physics as a new WRF set of parameterizations
   - Preserve WRF flexibility and capabilities

Technical details

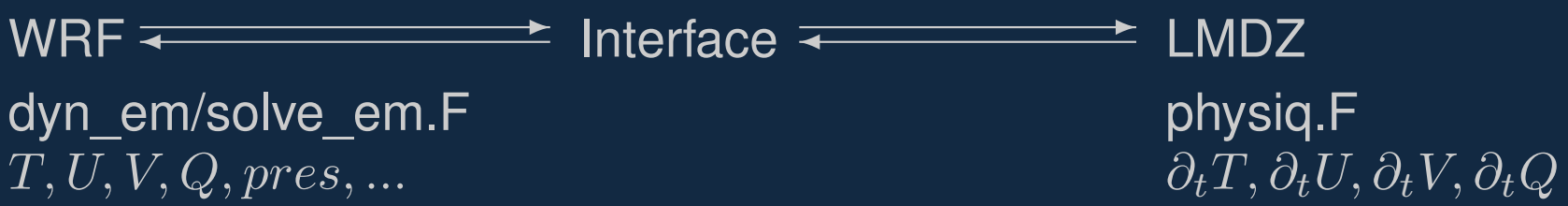
- LMDZ is a full set of physical schemes, deactivation of all WRF schemes!!

Technical details

- LMDZ is a full set of physical schemes, deactivation of all WRF schemes!!
- A WRF$\leftrightarrow$LMDZ interface is introduced in WRF code:

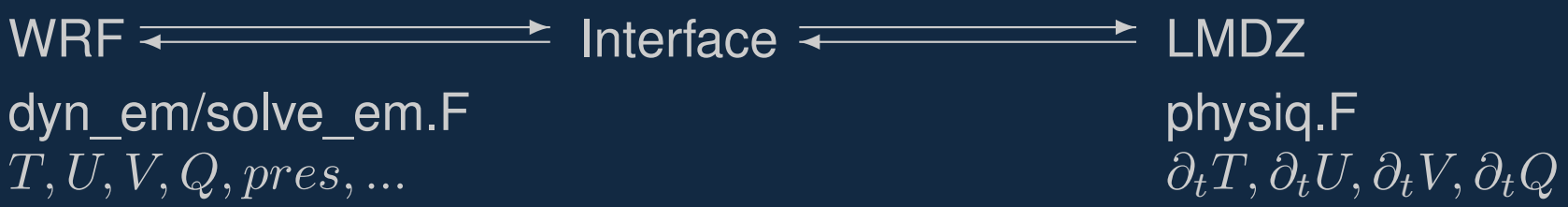
Technical details

- LMDZ is a full set of physical schemes, deactivation of all WRF schemes!!
- A WRF$\leftrightarrow$LMDZ interface is introduced in WRF code:



Technical details

- LMDZ is a full set of physical schemes, deactivation of all WRF schemes!!
- A WRF$\leftrightarrow$LMDZ interface is introduced in WRF code:



1. Provide WRF variables to LMDZ physics
2. Transform WRF variables (dimx,dimz,dimy) to LMDZ variables (klon,nlev) [klon=dimx*dimy]
3. Initialize LMDZ variables
4. Obtain state variable tendencies from LMDZ physics
5. Pass WRF initial/boundary variables to LMDZ (avoiding LMDZ input)
6. Retrieve all LMDZ diagnostic variables and include them in WRF (avoiding LMDZ output)

Technical: Implementation

- All necessary LMDZ code is located in a new folder `'WRFV3/lmdz'` (including also IOIPSL src as another folder inside)

Technical: Implementation

- All necessary LMDZ code is located in a new folder `'WRFV3/lmdz'` (including also IOIPSL src as another folder inside)
- The dependencies of `physiq.F` where followed in order to be able to compile the LMDZphysics source

Technical: Implementation

- All necessary LMDZ code is located in a new folder `'WRFV3/lmdz'` (including also IOIPSL src as another folder inside)
- The dependencies of `physiq.F` where followed in order to be able to compile the LMDZphysics source
- Different modules have been introduced

Technical: Implementation

- All necessary LMDZ code is located in a new folder `'WRFV3/lmdz'` (including also IOIPSL src as another folder inside)
- The dependencies of `physiq.F` where followed in order to be able to compile the LMDZphysics source
- Different modules have been introduced
- LMDZ dimensions are defined as: (content of `'dimensions.h'`):

```
PARAMETER (iim= 1, jjm=[e_we]*[e_sn],llm=[e_vert],ndm=1)
```

Technical: Implementation

- All necessary LMDZ code is located in a new folder `'WRFV3/lmdz'` (including also IOIPSL src as another folder inside)
- The dependencies of `physiq.F` where followed in order to be able to compile the LMDZphysics source
- Different modules have been introduced
- LMDZ dimensions are defined as: (content of `'dimensions.h'`):

```
PARAMETER (iim= 1, jjm=[e_we]*[e_sn],llm=[e_vert],ndm=1)
```

- An effort to map LMDZ variables and `SAVE` attributes for the `Registry` has been done

Technical: Flexible

- WRF has to be able to operate in the same way as if LMDZ was not introduced

Technical: Flexible

- WRF has to be able to operate in the same way as if LMDZ was not introduced
- In a WRF compilation with the LMDZ activated, LMDZ physics will be automatically used

Technical: Flexible

- WRF has to be able to operate in the same way as if LMDZ was not introduced
- In a WRF compilation with the LMDZ activated, LMDZ physics will be automatically used
- All the WRF physical schemes have to be switched off: land, surface, pbl, cumulus, micro-physics and sw/lw radiation

Technical: Flexible

- WRF has to be able to operate in the same way as if LMDZ was not introduced
- In a WRF compilation with the LMDZ activated, LMDZ physics will be automatically used
- All the WRF physical schemes have to be switched off: land, surface, pbl, cumulus, micro-physics and sw/lw radiation
- We could use WRF microphysics... for a further step

Technical: Easy use

- There is not any non-usual thing to do in order to run it

Technical: Easy use

- There is not any non-usual thing to do in order to run it
- Use the regular WRF `namelist.input` file and the LMDZ `*.def`

Technical: Easy use

- There is not any non-usual thing to do in order to run it
- Use the regular WRF `namelist.input` file and the LMDZ `*.def`
- LMDZ `physiq.def` flags: `iflag_pbl`, `lmdz_iflag_con`, `lmdz_iflag_thermals`, `lmdz_iflag_wake`, in order to compute some diagnostic output variables

Technical: Easy update

- It has been tried to keep the `'physiq.F'` as much unchanged as possible

Technical: Easy update

- It has been tried to keep the `'physiq.F'` as much unchanged as possible
- LMDZ code (v1818) is partially in F77 and an effort to 'translate' to F90 has been done

Technical: Easy update

- It has been tried to keep the `'physiq.F'` as much unchanged as possible
- LMDZ code (v1818) is partially in F77 and an effort to 'translate' to F90 has been done
- WRF calls LMDZ from a new subroutine in dyn_em/solve_em.F

Technical: Easy update

- It has been tried to keep the `'physiq.F'` as much unchanged as possible
- LMDZ code (v1818) is partially in F77 and an effort to 'translate' to F90 has been done
- WRF calls LMDZ from a new subroutine in dyn_em/solve_em.F
- Should be easy to update WRF

Technical: Easy update

- It has been tried to keep the `'physiq.F'` as much unchanged as possible
- LMDZ code (v1818) is partially in F77 and an effort to 'translate' to F90 has been done
- WRF calls LMDZ from a new subroutine in dyn_em/solve_em.F
- Should be easy to update WRF
- Is not that easy to update LMDZ. A script arise of importance to automatize the process:

Technical: Easy update

- It has been tried to keep the `'physiq.F'` as much unchanged as possible
- LMDZ code (v1818) is partially in F77 and an effort to 'translate' to F90 has been done
- WRF calls LMDZ from a new subroutine in dyn_em/solve_em.F
- Should be easy to update WRF
- Is not that easy to update LMDZ. A script arise of importance to automatize the process:
  1. Identify if the changed modules had changed in the new updated version
  2. Identify which new modules/subroutines/includes are necessary from the updated version
  3. Update the `lmdz/Makefile`

Technical: Easy update

- It has been tried to keep the `'physiq.F'` as much unchanged as possible
- LMDZ code (v1818) is partially in F77 and an effort to 'translate' to F90 has been done
- WRF calls LMDZ from a new subroutine in dyn_em/solve_em.F
- Should be easy to update WRF
- Is not that easy to update LMDZ. A script arise of importance to automatize the process:
  1. Identify if the changed modules had changed in the new updated version
  2. Identify which new modules/subroutines/includes are necessary from the updated version
  3. Update the `lmdz/Makefile`
- Part of the updating of LMDZ will be simplified once LMDZ code is fully in F90

Interface

- All the interface between WRF and LMDZ is done by `phys/module_lmdz_phys.F`

Interface

- All the interface between WRF and LMDZ is done by `phys/module_lmdz_phys.F`
- Three different steps:

Interface

- All the interface between WRF and LMDZ is done by `phys/module_lmdz_phys.F`
- Three different steps:
  1. Variables initialization: beginning of the simulation, pass WRF input variables to LMDZ

Interface

- All the interface between WRF and LMDZ is done by `phys/module_lmdz_phys.F`
- Three different steps:
  1. Variables initialization: beginning of the simulation, pass WRF input variables to LMDZ
  2. Call to `physiq.F90`: variable transformation from WRF to LMDZ shapes and units

Interface

- All the interface between WRF and LMDZ is done by `phys/module_lmdz_phys.F`
- Three different steps:
  1. Variables initialization: beginning of the simulation, pass WRF input variables to LMDZ
  2. Call to `physiq.F90`: variable transformation from WRF to LMDZ shapes and units
  3. LMDZ variables loading to WRF: passing obtained tendencies and all the LMDZ output variables

Interface

- All the interface between WRF and LMDZ is done by `phys/module_lmdz_phys.F`
- Three different steps:
  1. Variables initialization: beginning of the simulation, pass WRF input variables to LMDZ
  2. Call to `physiq.F90`: variable transformation from WRF to LMDZ shapes and units
  3. LMDZ variables loading to WRF: passing obtained tendencies and all the LMDZ output variables
  0. Special cases on: updating of boundary conditions, final restart variables

Interface

- All the interface between WRF and LMDZ is done by `phys/module_lmdz_phys.F`
- Three different steps:
  1. Variables initialization: beginning of the simulation, pass WRF input variables to LMDZ
  2. Call to `physiq.F90`: variable transformation from WRF to LMDZ shapes and units
  3. LMDZ variables loading to WRF: passing obtained tendencies and all the LMDZ output variables
  0. Special cases on: updating of boundary conditions, final restart variables
- `lmdz/wrf_lmdz_mod.F90`: Is a created module in order to make all the necessary calculations, transformations and actions necessary for the interface

Variables initialization. Aqua-planet

- This is done just at the beginning of the simulation or if it is a re-start

Variables initialization. Aqua-planet

- This is done just at the beginning of the simulation or if it is a re-start
- Using different LMDZ modules from the `gcm.F90` and/or `iniaqua` (or similars): `conf_gcm`, `infotrac_init`, `Init_Phys_lmdz`, `InitComgeomphy`, `phys_state_var_init`, `pbl_surface_init`, `iniphysiq`. Removing these calls from `physiq.F`

Variables initialization. Aqua-planet

- This is done just at the beginning of the simulation or if it is a re-start
- Using different LMDZ modules from the `gcm.F90` and/or `iniaqua` (or similars): `conf_gcm`, `infotrac_init`, `Init_Phys_lmdz`, `InitComgeomphy`, `phys_state_var_init`, `pbl_surface_init`, `iniphysiq`. Removing these calls from `physiq.F`
- WRF variable transformation:

Variables initialization. Aqua-planet

- This is done just at the beginning of the simulation or if it is a re-start
- Using different LMDZ modules from the `gcm.F90` and/or `iniaqua` (or similars): `conf_gcm`, `infotrac_init`, `Init_Phys_lmdz`, `InitComgeomphy`, `phys_state_var_init`, `pbl_surface_init`, `iniphysiq`. Removing these calls from `physiq.F`
- WRF variable transformation:
  - WRF θ temperatures to LMDZ T
  - Computation of LMDZ $area, cu, cv$
  - WRF water species to vapour/liquid LMDZ ones
  - WRF base/perturbation p and z variables to LMDZ total p and z
  - Computation of the LMDZ surface geopotential field (z_{sfc})

WRF input to LMDZ. Real cases

- Using WRF input files (`wrfinput_d01`, `wrfbdy_d01`, `wrflowinput_d01`) as input data for LMDZ (non use of `[re]startphy.nc`)

WRF input to LMDZ. Real cases

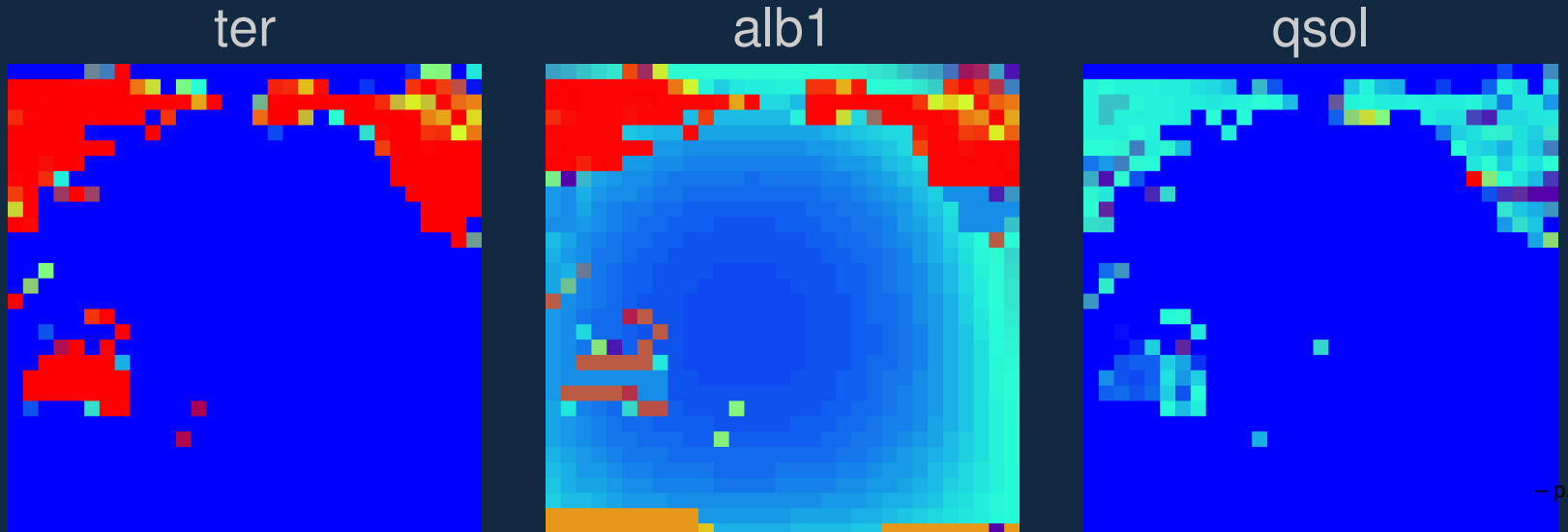
- Using WRF input files (`wrfinput_d01`, `wrfbdy_d01`, `wrflowinput_d01`) as input data for LMDZ (non use of `[re]startphy.nc`)
- WRF pre-process framework (`WPS`) provides all the necessary data highly flexible and from multiple resources: different geographical information (orography, landuse, soiltypes, vegetation,... from 2° to $30''$) and a large variety of GCMs (variables mapping throughout ASCII files), observations, FDDA, 4DVAR, ...

WRF input to LMDZ. Real cases

- Using WRF input files (`wrfinput_d01`, `wrfbdy_d01`, `wrflowinput_d01`) as input data for LMDZ (non use of `[re]startphy.nc`)
- WRF pre-process framework (`WPS`) provides all the necessary data highly flexible and from multiple resources: different geographical information (orography, landuse, soiltypes, vegetation,... from 2° to $30''$) and a large variety of GCMs (variables mapping throughout ASCII files), observations, FDDA, 4DVAR, ...
- WRF inputs transformations: LMDZ land categories: ter, lic, oce, sic, LMDZ total water $qsol$, soil temperatures $tsol, tsoil$, type soil variables of snow, albedo and rugosity

WRF input to LMDZ. Real cases

- Using WRF input files (`wrfinput_d01`, `wrfbdy_d01`, `wrflowinput_d01`) as input data for LMDZ (non use of `[re]startphy.nc`)
- WRF pre-process framework (`WPS`) provides all the necessary data highly flexible and from multiple resources: different geographical information (orography, landuse, soiltypes, vegetation,... from 2° to $30''$) and a large variety of GCMs (variables mapping throughout ASCII files), observations, FDDA, 4DVAR, ...
- WRF inputs transformations: LMDZ land categories: ter, lic, oce, sic, LMDZ total water $qsol$, soil temperatures $tsol, tsoil$, type soil variables of snow, albedo and rugosity



WRF input to LMDZ. Real cases

- Using WRF input files (`wrfinput_d01`, `wrfbdy_d01`, `wrflowinput_d01`) as input data for LMDZ (non use of `[re]startphy.nc`)
- WRF pre-process framework (`WPS`) provides all the necessary data highly flexible and from multiple resources: different geographical information (orography, landuse, soiltypes, vegetation,... from 2° to $30''$) and a large variety of GCMs (variables mapping throughout ASCII files), observations, FDDA, 4DVAR, ...
- WRF inputs transformations: LMDZ land categories: ter, lic, oce, sic, LMDZ total water $qsol$, soil temperatures $tsol, tsoil$, type soil variables of snow, albedo and rugosity
- Initialization of all the LMDZ restart variables mapping first WRF input variables to the LMDZ *[VAR]_rst* and then calling `pbl_surface_init`

WRF input to LMDZ. Real cases

- Using WRF input files (`wrfinput_d01`, `wrfbdy_d01`, `wrflowinput_d01`) as input data for LMDZ (non use of `[re]startphy.nc`)
- WRF pre-process framework (`WPS`) provides all the necessary data highly flexible and from multiple resources: different geographical information (orography, landuse, soiltypes, vegetation,... from 2° to $30''$) and a large variety of GCMs (variables mapping throughout ASCII files), observations, FDDA, 4DVAR, ...
- WRF inputs transformations: LMDZ land categories: ter, lic, oce, sic, LMDZ total water $qsol$, soil temperatures $tsol, tsoil$, type soil variables of snow, albedo and rugosity
- Initialization of all the LMDZ restart variables mapping first WRF input variables to the LMDZ *[VAR]_rst* and then calling `pbl_surface_init`
- `NOread_limit_sub_variables.F90:` Module to create `SAVE` definitions for that variables that were uploaded by `'limit.nc'` or `'startphy.nc'`
- `physiq_limit_variables_mod.F90:` Access restart LMDZ variables in `'physiq'`

LMDZ output to WRF. Real cases

- LMDZ output is avoided and WRF structure is used in replace

LMDZ output to WRF. Real cases

- LMDZ output is avoided and WRF structure is used in replace
- LMDZ output variables introduced in the `Registry` structure. All LMDZ variables start their name with the character `'L'`

LMDZ output to WRF. Real cases

- LMDZ output is avoided and WRF structure is used in replace
- LMDZ output variables introduced in the `Registry` structure. All LMDZ variables start their name with the character `'L'`
- `output_lmdz_NOmodule.F90` gives the `'SAVE'` attribute and allocate all that output variables which are locally used in `physiq.F90` and they do not appear in the LMDZ modules: `phys_state_var_mod`, `phys_output_var_mod`, `phys_local_var_mod`, `comgeomphy`

LMDZ output to WRF. Real cases

- LMDZ output is avoided and WRF structure is used in replace
- LMDZ output variables introduced in the `Registry` structure. All LMDZ variables start their name with the character `'L'`
- `output_lmdz_NOmodule.F90` gives the `'SAVE'` attribute and allocate all that output variables which are locally used in `physiq.F90` and they do not appear in the LMDZ modules: `phys_state_var_mod`, `phys_output_var_mod`, `phys_local_var_mod`, `comgeomphy`
- A series of different subroutines upload the output variables to the WRF `grid` structure: `get_lmdz_out2D_[i/v]`, `get_lmdz_out3D_z_[i/v]`

Boundary & restart variables

- WRF files `wrfbdy_d01` and `wrflopwbdy_d01` provide boundary and updating values for certain variables (usually from the GCMs, at every 6h). An interface to the LMDZ *[var]_lim* variables is needed to be created

Boundary & restart variables

- WRF files `wrfbdy_d01` and `wrflopwbdy_d01` provide boundary and updating values for certain variables (usually from the GCMs, at every 6h). An interface to the LMDZ *[var]_lim* variables is needed to be created
- At the end of the simulation (or at the given time that a restart file is required), LMDZ *[var]_rst* would be updated

Work in progress...

1. WRF using LMDZ physics: working on aqua-planet configuration

Work in progress...

1. WRF using LMDZ physics: working on aqua-planet configuration
2. LMDZ receiving WRF initial conditions: working
   - Orography variables for the orographic scheme: not done
   - GHG gases? $ozone, CO_2, CH_4, NO_2, CFC-11, CFC-12$, aerosols?: not done nor thought
   - WRF does not support *p_top = 0*. Add an extra vertical level used only by LMDZ? : not done nor thought

Work in progress...

1. WRF using LMDZ physics: working on aqua-planet configuration
2. LMDZ receiving WRF initial conditions: working
   - Orography variables for the orographic scheme: not done
   - GHG gases? $ozone, CO_2, CH_4, NO_2, CFC-11, CFC-12$, aerosols?: not done nor thought
   - WRF does not support *p_top = 0*. Add an extra vertical level used only by LMDZ? : not done nor thought
3. LMDZ output in WRF output: working ?
   - Statistic variables: not done nor thought
   - COSP outputs: not done nor thought

Work in progress...

1. WRF using LMDZ physics: working on aqua-planet configuration
2. LMDZ receiving WRF initial conditions: working
   - Orography variables for the orographic scheme: not done
   - GHG gases? $ozone, CO_2, CH_4, NO_2, CFC-11, CFC-12$, aerosols?: not done nor thought
   - WRF does not support *p_top = 0*. Add an extra vertical level used only by LMDZ? : not done nor thought
3. LMDZ output in WRF output: working ?
   - Statistic variables: not done nor thought
   - COSP outputs: not done nor thought
4. Updating of the boundary conditions: not done

Work in progress...

1. WRF using LMDZ physics: working on aqua-planet configuration
2. LMDZ receiving WRF initial conditions: working
   - Orography variables for the orographic scheme: not done
   - GHG gases? $ozone, CO_2, CH_4, NO_2, CFC-11, CFC-12$, aerosols?: not done nor thought
   - WRF does not support *p_top = 0*. Add an extra vertical level used only by LMDZ? : not done nor thought
3. LMDZ output in WRF output: working ?
   - Statistic variables: not done nor thought
   - COSP outputs: not done nor thought
4. Updating of the boundary conditions: not done
5. restart outputs: not done

Work in progress...

1. WRF using LMDZ physics: working on aqua-planet configuration
2. LMDZ receiving WRF initial conditions: working
   - Orography variables for the orographic scheme: not done
   - GHG gases? $ozone, CO_2, CH_4, NO_2, CFC-11, CFC-12$, aerosols?: not done nor thought
   - WRF does not support *p_top = 0*. Add an extra vertical level used only by LMDZ? : not done nor thought
3. LMDZ output in WRF output: working ?
   - Statistic variables: not done nor thought
   - COSP outputs: not done nor thought
4. Updating of the boundary conditions: not done
5. restart outputs: not done
6. WRF+LMDZ compiled in parallel: not done

Work in progress...

1. WRF using LMDZ physics: working on aqua-planet configuration
2. LMDZ receiving WRF initial conditions: working
   - Orography variables for the orographic scheme: not done
   - GHG gases? $ozone, CO_2, CH_4, NO_2, CFC-11, CFC-12$, aerosols?: not done nor thought
   - WRF does not support *p_top = 0*. Add an extra vertical level used only by LMDZ? : not done nor thought
3. LMDZ output in WRF output: working ?
   - Statistic variables: not done nor thought
   - COSP outputs: not done nor thought
4. Updating of the boundary conditions: not done
5. restart outputs: not done
6. WRF+LMDZ compiled in parallel: not done
7. Different WRF projections? Lambert Conformal, Mercator, Polar stereographic, Regular latitude-longitude, or cylindrical equidistant: not done nor thought

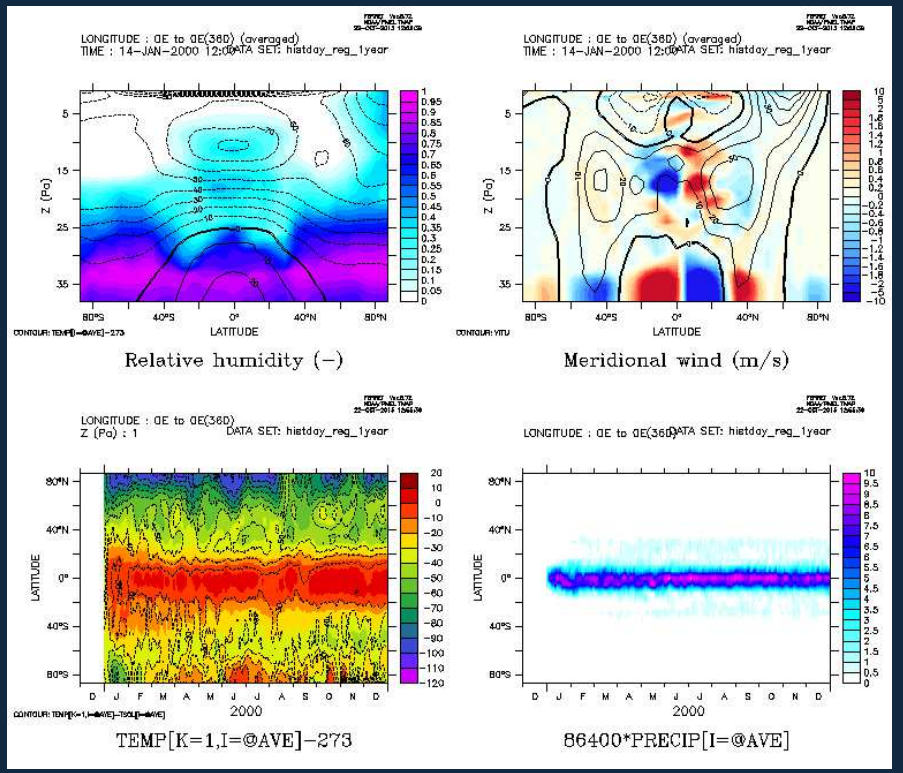
Work in progress...

1. WRF using LMDZ physics: working on aqua-planet configuration
2. LMDZ receiving WRF initial conditions: working
   - Orography variables for the orographic scheme: not done
   - GHG gases? $ozone, CO_2, CH_4, NO_2, CFC-11, CFC-12$, aerosols?: not done nor thought
   - WRF does not support *p_top = 0*. Add an extra vertical level used only by LMDZ? : not done nor thought
3. LMDZ output in WRF output: working ?
   - Statistic variables: not done nor thought
   - COSP outputs: not done nor thought
4. Updating of the boundary conditions: not done
5. restart outputs: not done
6. WRF+LMDZ compiled in parallel: not done
7. Different WRF projections? Lambert Conformal, Mercator, Polar stereographic, Regular latitude-longitude, or cylindrical equidistant: not done nor thought
8. couplings with ORCHIDEE, NEMO, INCA: not done nor thought

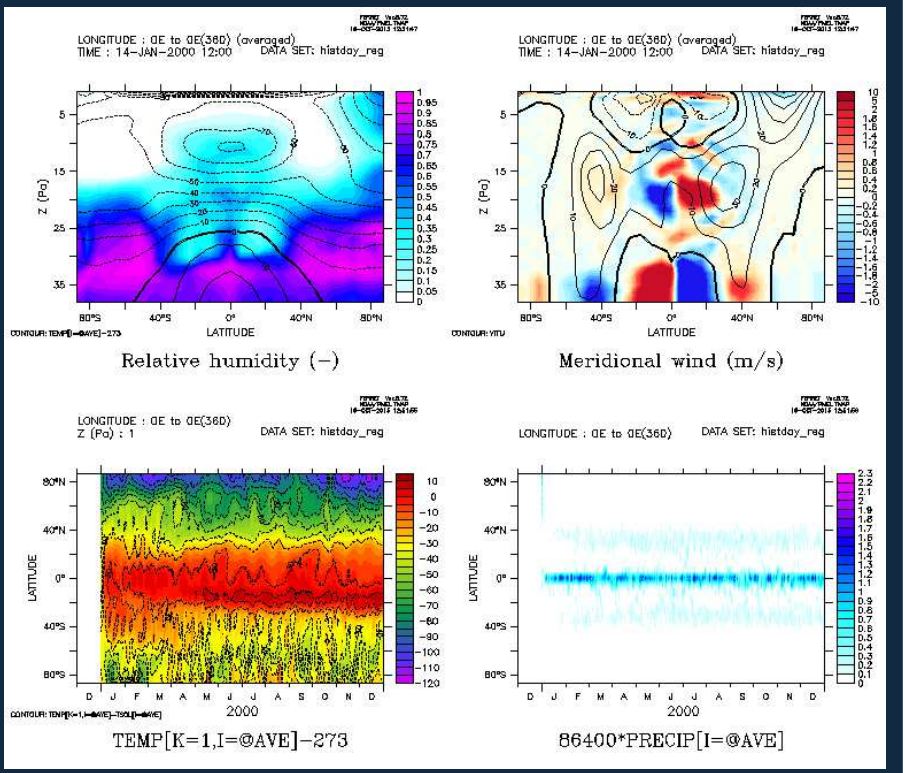
Preliminar results

- Global aqua-planet runs

WRF+LMDZ AR40.0

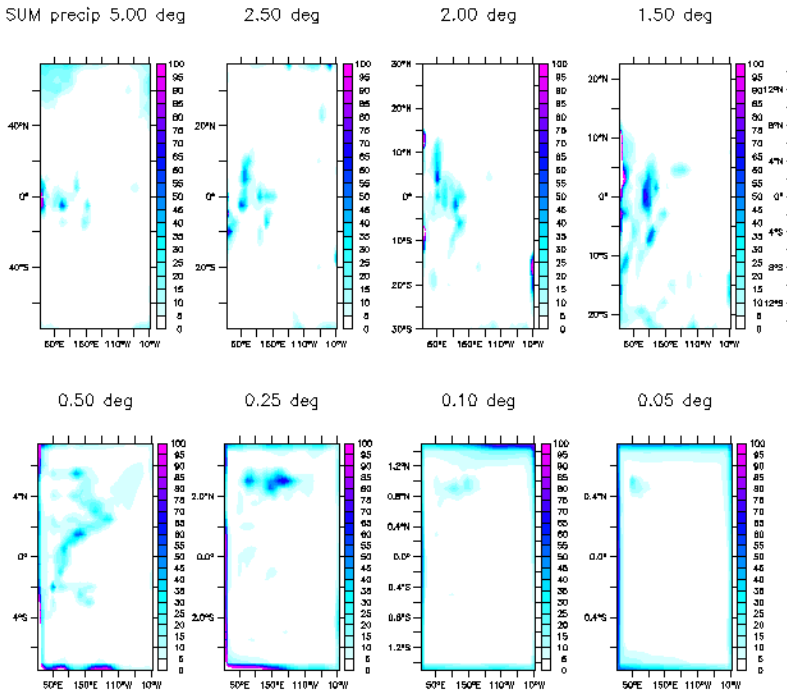


WRF+LMDZ NPv3.0



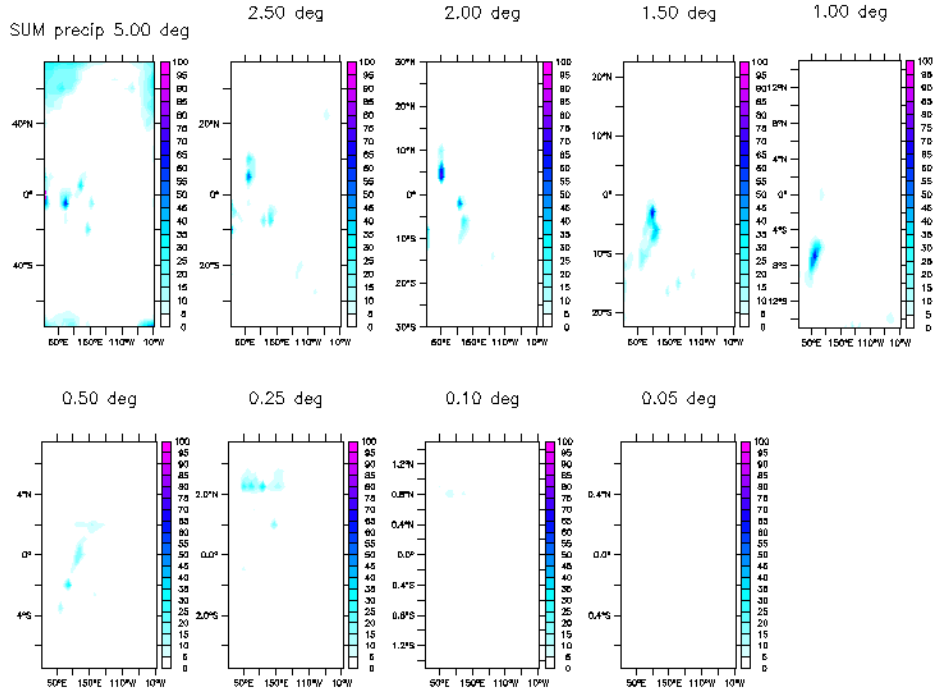
Preliminar results

- Regional aqua-planet runs: Zoom on point $N\ 0°, 171.56°W$



WRF+LMDZ AR40.0

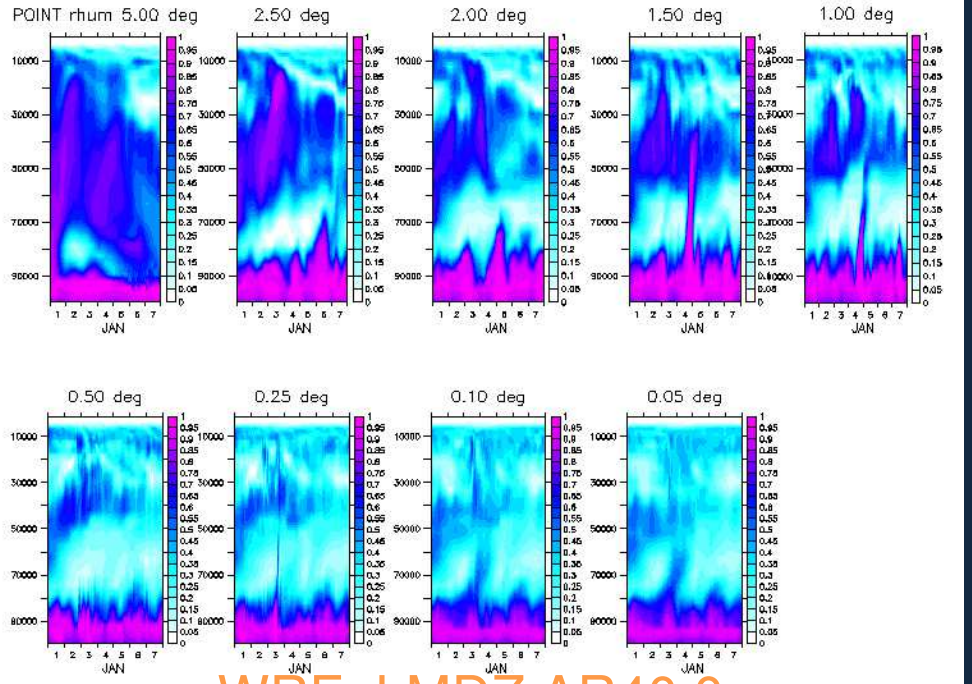
Precipitation



WRF+LMDZ NPv3.0

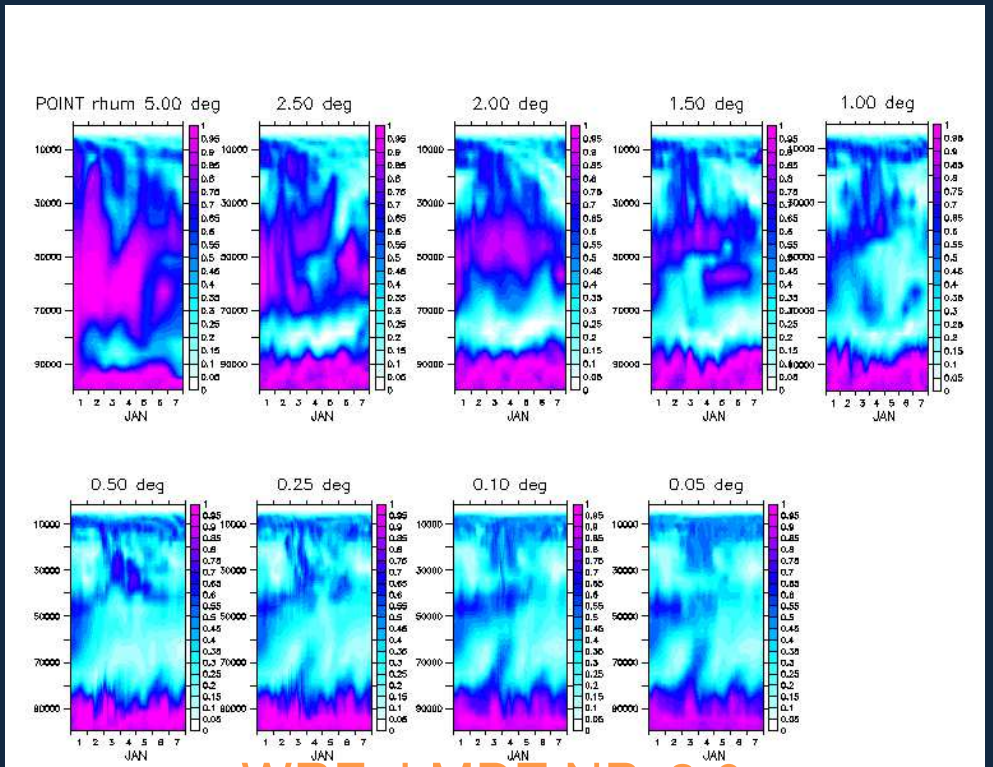
Preliminar results

- Regional aqua-planet runs: Zoom on point $N\ 0°, 171.56°W$



WRF+LMDZ AR40.0

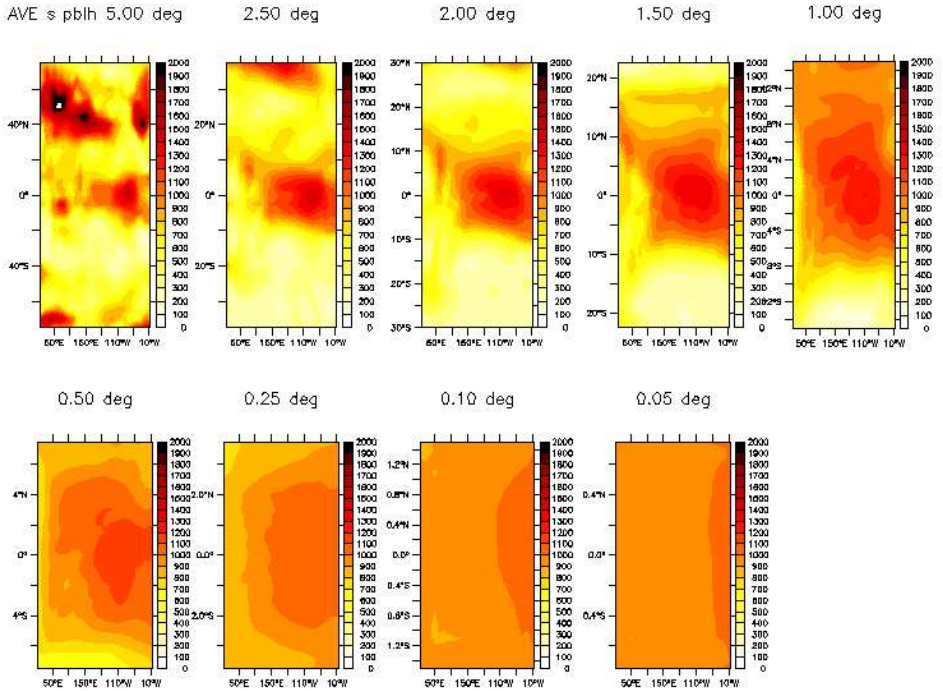
Vertical relative humidity



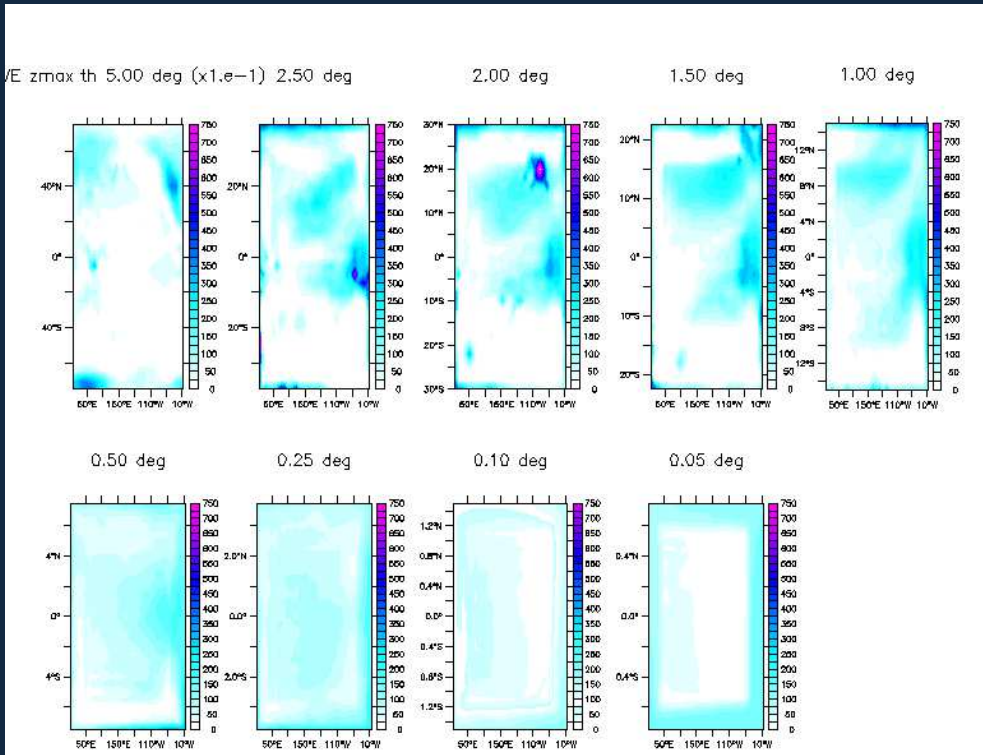
WRF+LMDZ NPv3.0

Preliminar results

- Regional aqua-planet runs: Zoom on point $N\ 0°, 171.56°W$



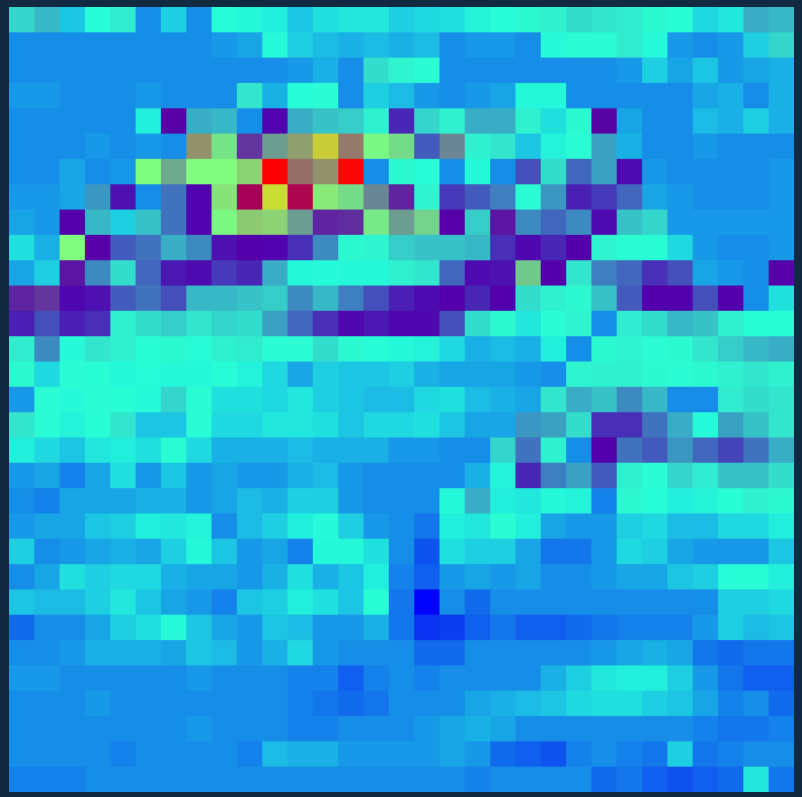
WRF+LMDZ NPv3.0 pbl height



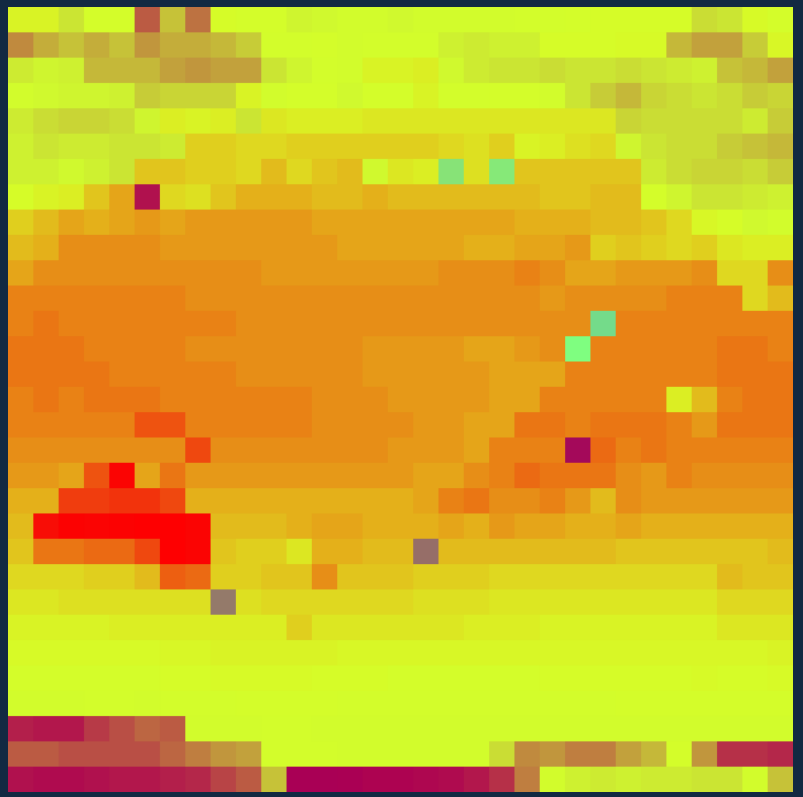
WRF+LMDZ NPv3.0 thermal h_{max}

Preliminar results

- Regional real 6h runs res $5°deg$: Centered on point $N\ 0°, 171.56°W$



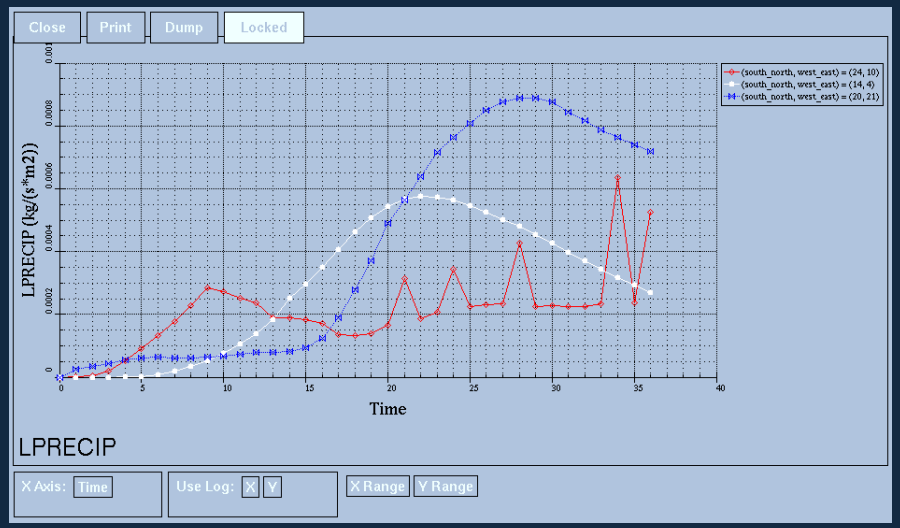
Surface evaporation



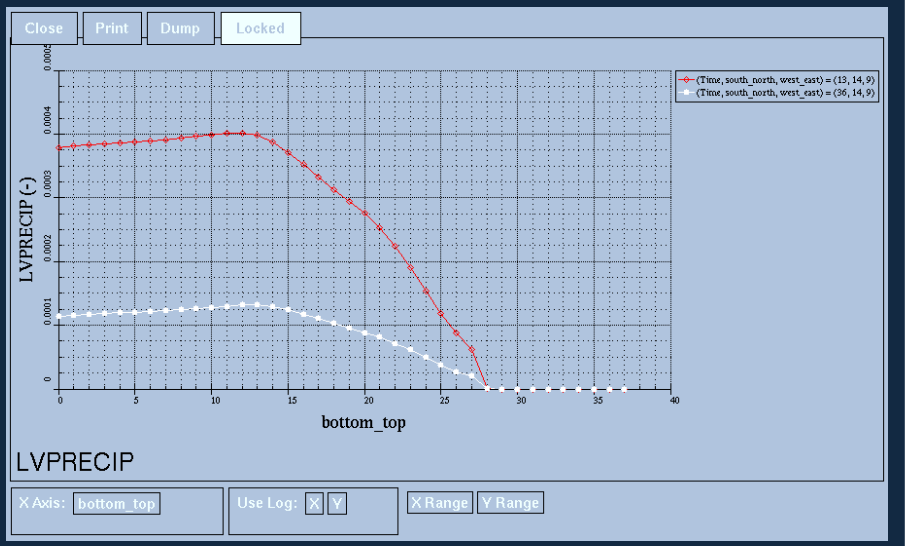
soil temperature

Preliminar results

- Regional real 6h runs res $5°deg$: Centered on point $N\ 0°, 171.56°W$



Precipitation evolution at different grid points



Vertical profile of precipitation at different times

Extra

soil categories computation from WRF

1. WRF land vegetation are retrieved from different geographical data-sets and resolutions

soil categories computation from WRF

1. WRF land vegetation are retrieved from different geographical data-sets and resolutions
2. An ASCII file `VEGPARM.TBL`. This information is kept in WRF `wrfinput_d01` as percentage of each type at each grid point in the variable `LANDUSE`

Extra

soil categories computation from WRF

1. WRF land vegetation are retrieved from different geographical data-sets and resolutions
2. An ASCII file `VEGPARM.TBL`. This information is kept in WRF `wrfinput_d01` as percentage of each type at each grid point in the variable `LANDUSE`
3. The vegetation types are grouped in the four LMDZ categeories: ter, lic, oce, sic and the total sum for each one is computed from `LANDUSE`

Extra

soil categories computation from WRF

1. WRF land vegetation are retrieved from different geographical data-sets and resolutions
2. An ASCII file `VEGPARM.TBL`. This information is kept in WRF `wrfinput_d01` as percentage of each type at each grid point in the variable `LANDUSE`
3. The vegetation types are grouped in the four LMDZ categeories: ter, lic, oce, sic and the total sum for each one is computed from `LANDUSE`
4. WRF does not distinguish between lic/sic, thus taking that WRF ice/glacier types and ter, the equivalents lic/sic are obtained:

$$lic = \sum_{ice}^{glacier} WRF_{soil} \times \frac{ter}{ter + oce}$$

$$sic = \sum_{ice}^{glacier} WRF_{soil} \times \frac{oce}{ter + oce}$$

Water soil content from WRF

1. WRF soil types are retrieved from different geographical data-sets and resolutions

Water soil content from WRF

1. WRF soil types are retrieved from different geographical data-sets and resolutions
2. An ASCII file `SOILPARM.TBL`. This information is kept in WRF `wrfinput_d01` as percentage of each type at each grid point in two variables `SOILCTOP` (top soil layer), `SOILCBOT` (bottom soil layer)

Water soil content from WRF

1. WRF soil types are retrieved from different geographical data-sets and resolutions
2. An ASCII file `SOILPARM.TBL`. This information is kept in WRF `wrfinput_d01` as percentage of each type at each grid point in two variables `SOILCTOP` (top soil layer), `SOILCBOT` (bottom soil layer)
3. Soil moistures (usually at 4 different depths) are provided in $m^3 m^{-3}$

Water soil content from WRF

1. WRF soil types are retrieved from different geographical data-sets and resolutions
2. An ASCII file `SOILPARM.TBL`. This information is kept in WRF `wrfinput_d01` as percentage of each type at each grid point in two variables `SOILCTOP` (top soil layer), `SOILCBOT` (bottom soil layer)
3. Soil moistures (usually at 4 different depths) are provided in $m^3 m^{-3}$
4. Soil percentages are interpolated at each layer

Water soil content from WRF

1. WRF soil types are retrieved from different geographical data-sets and resolutions
2. An ASCII file `SOILPARM.TBL`. This information is kept in WRF `wrfinput_d01` as percentage of each type at each grid point in two variables `SOILCTOP` (top soil layer), `SOILCBOT` (bottom soil layer)
3. Soil moistures (usually at 4 different depths) are provided in $m^3 m^{-3}$
4. Soil percentages are interpolated at each layer
5. Dry density of each soil type is read from the `SOILPARM.TBL` file.

Water soil content from WRF

1. WRF soil types are retrieved from different geographical data-sets and resolutions
2. An ASCII file `SOILPARM.TBL`. This information is kept in WRF `wrfinput_d01` as percentage of each type at each grid point in two variables `SOILCTOP` (top soil layer), `SOILCBOT` (bottom soil layer)
3. Soil moistures (usually at 4 different depths) are provided in $m^3 m^{-3}$
4. Soil percentages are interpolated at each layer
5. Dry density of each soil type is read from the `SOILPARM.TBL` file.
6. Taking the soil moistures and soil density at each level, the total soil water content at each grid point is computed as follows:

$$qsol = \sum_{k=1}^{4} \sum_{s=i}^{Nsoil} \delta x \delta y \delta z(k) \rho_s smois(k)$$