

CORDEX-WRF: creation of a module for WRF to provide the extra CORDEX output

L. Fita¹, R. Pennel², J. Polcher², K. Béranger^{2,5}, S. Bastin³, and T. Arsouze^{2,4}

¹Centro de Investigaciones del Mar y la Atmósfera (CIMA), CONICET-UBA, CNRS UMI-IFAECI, Buenos Aires, Argentina

²Laboratoire de Météorologie Dynamique (LMD), IPSL, CNRS, École Polytechnique, Palaiseau, France

³Laboratoire Atmosphères, Milieux, Observations Spatiales (LATMOS), IPSL, CNRS, Guyancourt, France

⁴École Nationale Supérieure de Techniques Avancées (ENSTA ParisTech) , 828, Boulevard des Maréchaux , 91762 Palaiseau Cedex , France

⁵Université Grenoble Alpes (UGA), CNRS, IRD, IGE, Grenoble, France

March 23, 2018

Abstract

CORDEX requirements of data for stake holders and decision making community, push the output of the atmospheric models, which demands that usually require time consuming post-process of the standard model output. In order to avoid this time and effort consuming post-processing task, here is presented the implementation of a new module into the Weather and Forecasting Model (WRF, <http://www.mmm.ucar.edu/wrf/users/> Skamarock et al., 2008) module called `module_diag_cordex` with which is expected to substantially limit the need of post-processing.

A WIKI version of this document can be found at:

<http://wiki.cima.fcen.uba.ar/mediawiki/index.php/CDXWRF>

Contents

1	CORDEX requirements	4
2	module_diag_cordex	5
2.1	Additional: pressure levels interpolation	5
2.2	Additional: water budget	5
2.3	Installation	5
2.4	Usage	6
2.5	Pressure interpolation	7
3	Variables	8
3.1	clt: total cloudiness	10
3.2	cllmh: low, medium and high cloudiness	10
3.3	wsgsmax: Maximum Near-Surface Wind Speed of Gust	10
3.4	wsgsmax100: Daily Maximum Near-Surface Wind Speed of Gust at 100 m	12
3.5	prw: precipitable water or water vapor path	15
3.6	clwvi: condensed water path	15
3.7	clivi: ice water path	15
3.8	clgvi: graupel water path	15
3.9	clhvi: hail water path	16
3.10	psl: sea level pressure	16
3.11	cape: convective available potential energy	16
3.12	cin: convective inhibition	18
3.13	sund: duration of sunshine	18
3.14	hur: relative humidity	18
3.15	hus: specific humidity	18
3.16	zg: geopotential height	20
3.17	press: air-pressure	20
3.18	ta: air-temperature	20
3.19	ua/va: air-wind Earth oriented	20
3.20	cdgen	20
3.21	tauvv	20
3.22	evspsblpot	21
3.23	rsus	21
3.24	rlus	21
4	Additional variables	22
4.1	Water vapor balance terms	22
4.2	zmlagen: generic boundary layer height	29
5	Work done	31
6	Missing variables	32
6.1	wsgsmax100: Daily Maximum Near-Surface Wind Speed of Gust at 100 m	32
6.2	ic_lightning, cg_lightning, tot_lightning: intra-cloud, ground and total lightning flashes	32
6.3	praccmov	32
7	Others	33
7.1	New variables	33
7.2	CF-compilant file	33
7.3	Optimization	33
A	clt	35
A.1	code from LMDZ	35

B	cllmh	35
	B.1 cllmh: code from LMDZ	35
C	wsgsmax	36
	C.1 afwa	36
	C.2 lmdz	37
D	psl	37
	D.1 afwa	37
	D.2 lmdz	38
	D.3 p_interp	38
E	cape	40
	E.1 afwa	40
	E.2 lmdz	45
F	cin	45
	F.1 afwa	45
	F.2 lmdz	46
G	WRF CORDEX variables definition	46

1 CORDEX requirements

CORDEX requirements of data must cover all the possible needs of stake holders, and scientists working on the adaptation and mitigation strategies. They are grouped in different levels of frequency and priority. A working copy of this list is available here: <https://www.hymex.org/cordexfps-convection/wiki/doku.php?id=protocol> from the CORDEX convection permitting Flag Ship Pilot study.

Some of the variables are not directly computed in the WRF model which require to extend the model output in order to provide enough variables to post-process the variables.

The implementation of the `module_diag_cordex` module should allow to avoid the post-processing by computing the CORDEX-required (Core & Tier) variables during model integration

NOTE: Be aware that any systematic checking process has been applied to this module. Use it and the variables therein at your own risk !! It has been tested on a 2-nested domain configuration with the inner domain at cloud resolving resolution ($< 5\text{ km}$, without cumulus scheme), making use of restarts and on pure distributed memory parallel environment

2 module_diag_cortex

The module is basically based on two modules:

- `phys/module_diag_cortex.F`: Main module which manages the calls to the variables and the accumulations for the means, ...
- `phys/module_diagvar_cortex.F`: Module with the computation of all the variables

This module is accompanied with a new `Registry/registry.cortex` where the variables and a new section in the `namelist.inpt` labeled `cortex` are defined. There are other necessary complementary modifications on `phys/module_diagnostics_driver.F` encompassed by the pre-compilation flag `CORDEXDIAG`, as well some modifications in the `main/depend.common` and `phys/Makefile`.

Output is provided by the auxiliary history output `#9` with a provisional file name: `wrfcortex_d<domain>_<date>`

All that variables which are only required at output time step, are computed only at that exact time.

2.1 Additional: pressure levels interpolation

At the same time, WRF can output on pressure levels while integration. However, initial version of the module does not include required CORDEX variables: `wa` (vertical wind speed) and `hus` (specific humidity). Thus, code has also been modified and now, WRF output at pressure levels also provides `wa` and `hus`.

It has been accomplished after modifying the codes: `Registry/registry.diags`, `phys/module_diagnostics_driver.F`, `phys/module_diag_pld.F` and `dyn_em/start_em.F`. The three latest modifications are also encapsulated within pre-compilation flag `CORDEXDIAG`.

See more details in how to activate this option in WRF users web-page over the namelist section `diags&`.

2.2 Additional: water budget

It has been added also the components of the atmospheric water budget. They are accumulated internally and vertically integrated all over the column. In order to provide this capability, a series of modifications have been introduced in `dyn_em/solve_em.F`

2.3 Installation

These steps must be followed prior the re-compilation of the WRF model and assuming that the process is started where the code resides (WRFV3)

1. Untar the file

```
$ tar xvfz WRF_CORDEX.tar.gz
```

2. It deflates all the required files and the modified original WRF files

```
main/depend.common
dyn_em/solve_em.F
dyn_em/start_em.F
phys/module_diagnostics_driver.F
phys/module_diag_cortex.F
phys/module_diagvar_cortex.F
phys/module_diag_pld.F
phys/Makefile
README.cortex
Registry/registry.cortex
Registry/registry.diags
```

3. On the `Registry/Registry.EM` add the following line (after the line with `include registry.new3d_wif`)

2.5 Pressure interpolation

Remember to activate section `&diags` in order to get pressure-level vertical interpolation of state variables (g.e.: assuming 6 levels only and output every 3 hours)

```
&time_control
(...)
auxhist23_outname="wrfpress_d<domain>_<date>"
io_form_auxhist23 = 2,
auxhist23_interval = 180, 180,
frames_per_auxhist23 = 100, 100,

(...)
/
(...)

&diags
p_lev_diags = 1,
num_press_levels = 6,
press_levels = 100000, 92500, 85000, 70000, 50000, 20000
use_tot_or_hyd_p = 1
p_lev_missing = -999.
/
```

3 Variables

These are the different variables added and their implementations from the WRF point of view. There might be necessary to revise some of them, or even decide which version to use

In case of accumulation/mean they are also be included

These variables are:

- Instantaneous diagnostics (only computed on output times)
 - **prw**: Total water path
 - **clwvi**: Total liquid water path (QCLOUD + QRAIN)
 - **clivi**: Total ice water path (QSNOW+QICE+GRAUPEL+QHAIL)
 - **ua**: 3D earth-rotated eastward wind [ms-1]
 - **va**: 3D earth-rotated northward wind [ms-1]
 - **ws**: 3D wind speed [ms-1]
 - **ta**: 3D air-temperature [K]
 - **press**: 3D air pressure [Pa]
 - **zg**: 3D geopotential height [m]
 - **hur**: 3D relative humidity [1]
 - **hus**: 3D specific humidity [1]
 - **uas**: 10m earth-rotated eastward wind [ms-1]
 - **vas**: 10m earth-rotated northward wind [ms-1]
 - **wss**: 10m wind speed [ms-1]
 - **hurs**: 2m relative humidity [1]
 - **huss**: 2m specific humidity [1]
 - **psl**: sea level pressure [Pa] (three different ways)
 - **cape**: Convective Available Potential Energy [Jkg-1]
 - **cin**: Convective inhibition [Jkg-1]
 - **zlfrc**: Height at the Level of free convection [m]
 - **plfrc**: Pressure at the Level of free convection [Pa]
 - **li**: Lifted index [1]
 - **mrso**: total soil moisture content [kgm-2]
 - **slw**: total liquid water content [kgm-2]
 - **zmla**: pbl height following a generic method [m]
 - **ws100**: 100m wind speed [ms-1]
 - **uz100**: 100m wind x-direction [ms-1]
 - **vz100**: 100m wind y-direction [ms-1]
 - **tauu**, **tauuv**: components of the downward wind stress at 10 m [m2s-2]
 - **cdgen**: generic drag coefficient [-]
- Accumulated or similar time dependency (computed at every time-step). They are initialized after each output time-step. Thus, they represent statistics (mean, accumulation) only from between output time-steps.
 - **clt**: total cloud cover [1]¹

¹**NOTE:** CLDFRAC is computed by the radiative scheme thus, bear in mind to configure the `namelist.input` that: `auxhist9_interval` > `radt` otherwise one obtains repeated values of `clt`, `c1l`, `c1m`, `c1h`

- `c11`: low-level cloud cover [1]
- `c1m`: mid-level cloud cover [1]
- `c1h`: high-level cloud cover [1]
- `cltmean`: mean `clt`
- `c1lmean`: mean `c1l`
- `c1mmean`: mean `c1m`
- `c1hmean`: mean `c1h`
- `wsgsmax`: maximum surface wind gust [ms-1] (two different methods)
- `ugsmax`: eastward maximum surface gust wind direction [ms-1]
- `vgsmax`: northward maximum surface gust wind direction [ms-1]
- `wsgspercen`: percentage of times when grid point got gust wind [%]
- `totwsgsmax`: maximum surface wind gust [ms-1] (addition of different methods)
- `totugsmax`: eastward maximum surface gust wind direction [ms-1]
- `totvgsmax`: northward maximum surface gust wind direction [ms-1]
- `totwsgspercen`: percentage of times when grid point got total gust wind [%]
- `wsz100max`: maximum 100m wind [ms-1] (two different methods)
- `uz100max`: eastward maximum 100m wind direction [ms-1]
- `vz100max`: northward maximum 100m wind direction [ms-1]
- `sund`: sunshine length [s]
- `rsds`: mean surface Downwelling Shortwave Radiation [Wm-2]
- `rlds`: mean surface Downwelling Longwave Radiation [Wm-2]
- `hfls`: mean surface Upward Latent Heat Flux [Wm-2]
- `hfss`: mean surface Upward Sensible Heat Flux [Wm-2]
- `rsus`: mean surface Upwelling Shortwave Radiation [Wm-2]
- `r1us`: mean surface Upwelling Longwave Radiation [Wm-2]
- `evspsbl`: mean evaporation [kgm-2s-1]
- `evspsblpot`: mean potential evapotranspiration [kgm-2s-1]
- `snc`: mean snow area fraction [
- `snd`: mean snow depth [m]
- `mrros`: mean surface Runoff [kgm-2s-1]
- `mrro`: mean total Runoff [kgm-2s-1]
- `mrsol`: mean total water content of soil layer [kgm-2]
- `pr`: precipitation flux [kgm-2s-1]
- `pr1`: large scale precipitation flux [kgm-2s-1]
- `prc`: convective precipitation flux [kgm-2s-1]
- `snw`: accumulated snow [ksm-2]
- Additionally added referred to the water budget in the atmosphere (not required by CORDEX):
 - * `wbacdiabh`: Water-budget vertically integrated accumulated of diabatic heating from microphysics [K]
 - * `wbacpw`, `wbacpw[c/r/s/i/g/h]`: Water-budget vertically integrated accumulated total tendency for water vapour, cloud, rain, snow, ice, graupel, hail [mm]
 - * `wbacf`, `wbacf[c/r/s/i/g/h]`: Water-budget vertically integrated accumulated horizontal advection for water vapour, cloud, rain, snow, ice, graupel, hail [mm]

- * **wbacz**, **wbacz[c/r/s/i/g/h]**: Water-budget vertically integrated accumulated vertical advection for water vapour, cloud, rain, snow, ice, graupel, hail [mm]
- * **wbacdiabh{1/m/h}**: Water-budget vertically integrated accumulated of diabatic heating from micro-physics at low, medium and high levels (same as cloudiness) [K]
- * **wbacpw[v/c/r/s/i/g/h]{1/m/h}**: Water-budget vertically integrated accumulated total tendency for water vapour, cloud, rain, snow, ice, graupel, hail at low, medium and high levels (same as cloudiness) [mm]
- * **wbacf[v/c/r/s/i/g/h]{1/m/h}**: Water-budget vertically integrated accumulated horizontal advection for water vapour, cloud, rain, snow, ice, graupel, hail at low, medium and high levels (same as cloudiness) [mm]
- * **wbacz[v/c/r/s/i/g/h]{1/m/h}**: Water-budget vertically integrated accumulated vertical advection for water vapour, cloud, rain, snow, ice, graupel, hail at low, medium and high levels (same as cloudiness) [mm]

- Pressure interpolation

- **hus_pl**: specific humidity [1]
- **w_pl**: vertical wind speed [ms-1]

3.1 clt: total cloudiness

This variable computes the total cloudiness above a grid point taking as input the cloud fraction of a given grid cell and level.

NOTE:

*cloud fraction in WRF is computed by the radiative scheme, which is called at a frequency given by **radt**. It should be taking into account when one gets any accumulation of any value retrieved from it. Otherwise, one could compute the cloud fraction at every time-step (using any of the subroutines from module_radiation_driver.F: **cal_cldfra1**, **cal_cldfra2**, **cal_cldfra3**), but then it will not be consistent in what was already considered whilst model integration*

The most common implementation assumes ‘*random overlapping*’ and its implemented in most of the global climate models. Here is considered to take the implementation from the GCM LMDZ (http://lmdz.lmd.jussieu.fr/?set_language=en) Hourdin et al., 2006). Calculation of the total cloudiness is done inside the subroutine **newmicro.f90**.

Specific variable computation has already been extracted and implemented as a subroutine for the python utils PyNCplot (<http://www.xn--1lusfb-5va.cat/python/PyNCplot>). The code is provided in the appendix A.1

3.2 cllmh: low, medium and high cloudiness

This variable computes the total cloudiness above a grid point at different vertical intervals (low: $p \geq 680hPa$, medium: $680 < p \leq 400 hPa$, high: $p < 400 hPa$) taking as input the cloud fraction of a given grid cell.

As in the case of the ‘clt’ calculation from LMDZ has already been implemented as an independent subroutine. See appendic B.1 for a snapshot of the code. See in figure 1 the result of the implementation

3.3 wsgsmax: Maximum Near-Surface Wind Speed of Gust

The wind gust accounts for the wind from upper levels that is projected to the surface due to instability within the boundary layer. It can have different implementations. Winds are Earth-rotated.

- Brasseur01: An implementation of a wind gust following Turbulent Kinetic Energy ($\mathcal{TK}\mathcal{E}$) estimates and stability by virtual temperature (θ_v , see mainly equation 1) reproducing Brasseur (2001) from the clWRF (clWRF, <http://www.meteo.unican.es/wiki/cordexwrf/SoftwareTools/ClWrf> Fita et al., 2010) [**wsgs_diag** = 1]

$$\frac{1}{z_p} \int_0^{z_p} \mathcal{TK}\mathcal{E}(z) dz \geq \int_0^{z_p} g \frac{\Delta\theta_v(z)}{\Theta_v(z)} dz \quad (1)$$

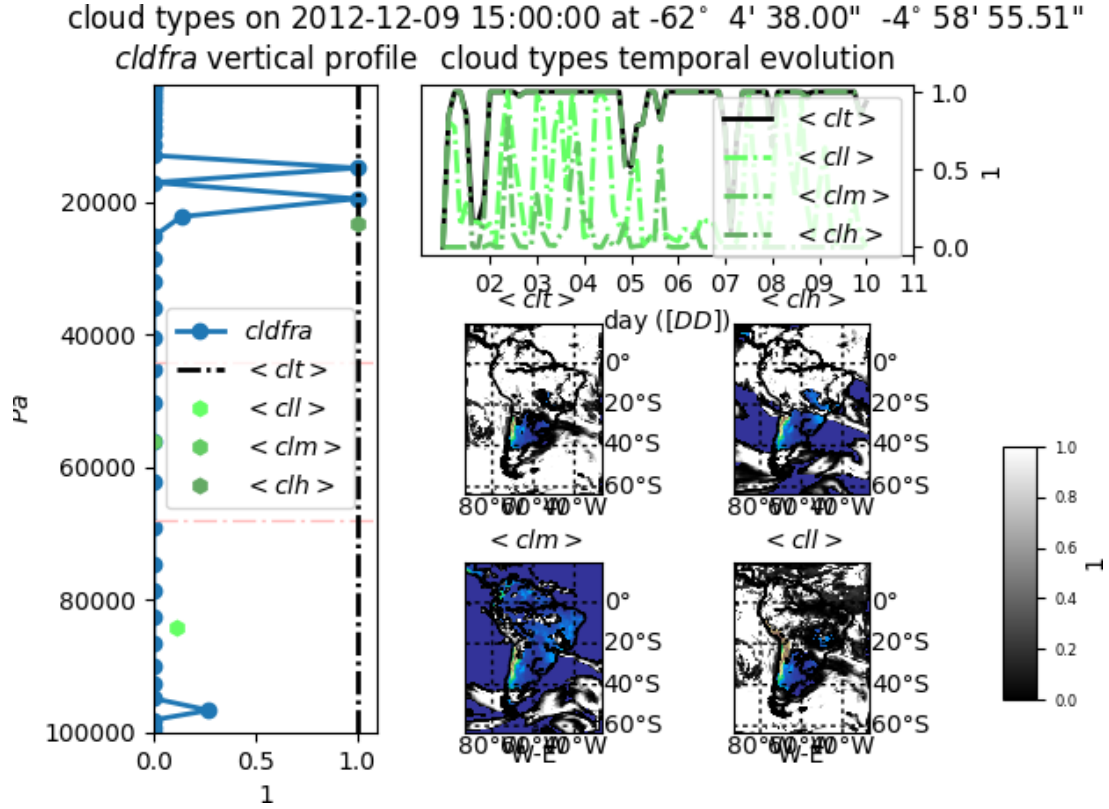


Figure 1: Vertical distribution of cloud fraction and the different cloud types at a given point (top left): cloud fraction (*cldfra*, full circles with line in blue), mean total cloud fraction (*cldmean*, vertical dashed line), mean low-level cloud fraction (*cldmean* $p \geq 680$ hPa, dark green hexagon), mean mid-level (*clmmean* $680 < p \leq 440$ hPa, green hexagon), mean high-level (*clhmean* $p < 440$ hPa, clear green hexagon). Temporal evolution of cloud types at the given point (top right). Map of *cldmean* with colored topography beneath to show-up cloud extent (middle middle), map of *clhmean* (middle right), map of *clmmean* (bottom middle) and map of *cldmean* (bottom right)

- WRF afwa diagnostics: Inside the WRF module `module_diag_afwa.F` there is an implementation of the calculation of the wind gust which only occurs as a blending of upper-level winds (around 1km above ground $zagl$; $zagl(k_{1000}) \geq 1000$ m, see equation 2) above a given maximum precipitation intensity of $prate_{hr}^{mm} \geq 50$ mmh^{-1} (see the code in appendix C.1) [`wsgs_diag` = 2]

$$\begin{aligned}
\vec{v}_{a_{1km}} &= \vec{v}_a(k_{1000} - 1) + [1000 - zagl(k_{1000} - 1)] \frac{\vec{v}_a(k_{1000}) - \vec{v}_a(k_{1000} - 1)}{zagl(k_{1000})} \\
\gamma &= \frac{150 - prate_{hr}^{mm}}{100} \\
\vec{v}_{a_{blend}} &= \vec{v}_a s \gamma + \vec{v}_{a_{1km}} \times (1 - \gamma)
\end{aligned} \tag{2}$$

These two methodologies have been implemented and can be switched by a new `namelist.input` parameter labeled `wsgs_diag` (in `cordex` section). Its default value is 1

It comes out, that both methodologies provide wind gust estimation (WGE) from two different perspectives: mechanic and convective. In order to take into account both winds gusts, another variable as the addition of both estimations is provided as `totwsgsmax`, `totugsmax`, `totvgmax`, `totwsgspercen`. On figure 2 is shown the different outcomes applying each approximation

3.4 wsgsmax100: Daily Maximum Near-Surface Wind Speed of Gust at 100 m

The wind gust at 100 m is understood that should follow a similar implementation than for the `wsgsmax`, but at 100 m, since is understood than an extrapolation of such turbulent phenomena it would require a complete new set of equations. This one is let to open discussion.

Instead as a way to overcome it, the estimation of maximum wind speed at 100 m is provided. Winds are Earth-rotated. After PhD thesis of Jourdier (2015), two different methodologies are implemented to estimate the wind at 100 m above ground:

- Following power-law wind vertical distribution, as it is depicted in equation 3 using the upper-level atmospheric wind speed below ($k_{100}^<$) and above ($k_{100}^>$) the height above ground of 100 m ($zagl$) [`wsz100_diag` = 1]

$$\begin{aligned}
\vec{v}_{a_{100}} &= \vec{v}_a(k_{100}^>) \left(\frac{100.}{zagl(k_{100}^>)} \right)^{\alpha_{x,y}} \\
\alpha_{x,y} &= \frac{\ln(\vec{v}_a(k_{100}^>)) - \ln(\vec{v}_a(k_{100}^<))}{\ln(zagl(k_{100}^>)) - \ln(zagl(k_{100}^<))}
\end{aligned} \tag{3}$$

- Following logarithmic-law wind vertical distribution, as it is depicted in equation 4 using upper-level atmospheric wind speed below ($k_{100}^<$) and above ($k_{100}^>$) the height above ground of 100 m ($zagl$) [`wsz100_diag` = 2]

$$\begin{aligned}
\ln(z_0) &= \frac{\vec{v}_a(k_{100}^>) \ln(zagl(k_{100}^<)) - \vec{v}_a(k_{100}^<) \ln(zagl(k_{100}^>))}{\vec{v}_a(k_{100}^>) - \vec{v}_a(k_{100}^<)} \\
\vec{v}_{a_{100}} &= \vec{v}_a(k_{100}^>) \frac{\ln(100.) - \ln(z_0)}{\ln(zagl(k_{100}^>)) - \ln(z_0)}
\end{aligned} \tag{4}$$

- Following Monin-Obukhov theory is implemented and was tested, but it is not useful for heights larger than few decameters ($z > 80$. m). However, the necessary code to extrapolate the wind at given height is left commented just in case someone wants to use it.

These two methodologies have been implemented and can be switched by a new `namelist.input` parameter labeled `wsz100_diag` (in `cordex` section). Its default value is 1. Even one can select another height for the estimation by providing the new parameter `z100m_wind` with a different value than 100 m (default value)

On figure 3 is shown the different outcomes applying each approximation. There are some problems on Monin-Obukhov application under certain stable conditions (too small Obukhov length)

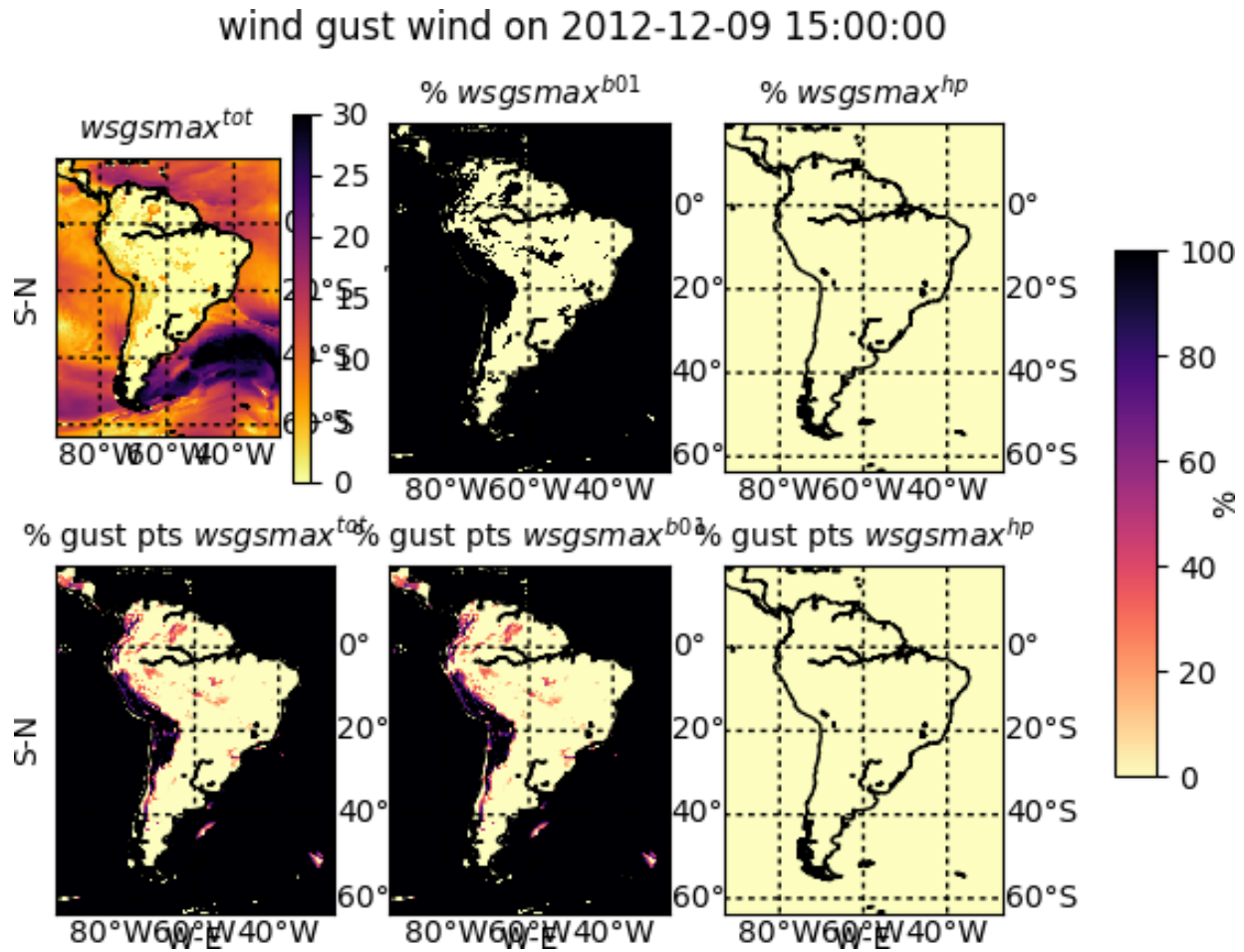


Figure 2: near surface wind gust estimates. 3h-maximum total wind gust strength ($wsgsmax^{tot}$, top left), percentage of $wsgsmax^{tot}$ due to Brasseur's application ($wsgsmax^{b01}$, top middle), percentage due to AFWA-heavy precipitation implementation ($wsgsmax^{hp}$, top right), percentage of time-steps where grid point got total wind gust (bottom left), percentage of time-steps where grid point got $wsgsmax^{b01}$ (bottom middle), percentage due to $wsgsmax^{hp}$ (bottom right)

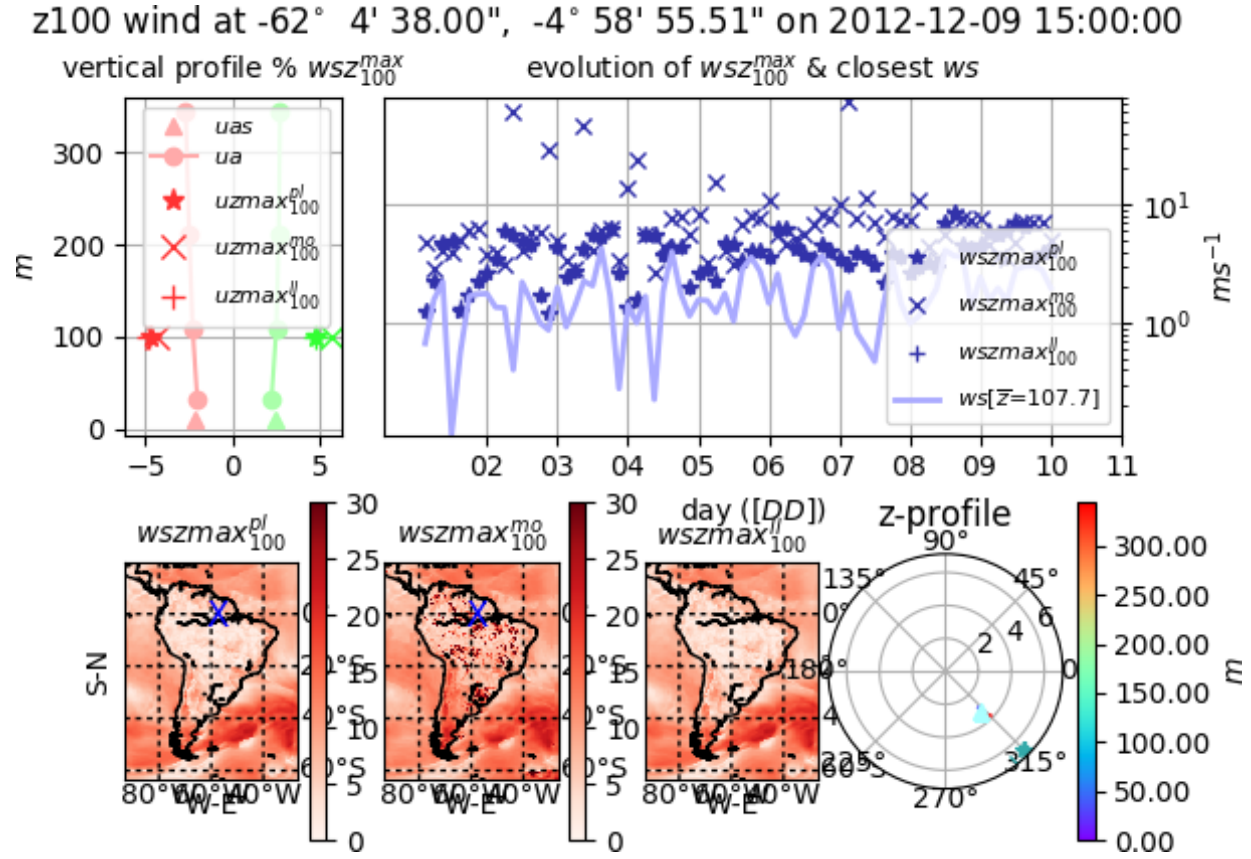


Figure 3: 100 m wind estimates. Comparison between upper-level winds and estimation at a given point and moment (upper left): 3h-maximum eastward wind (red) at 100 m by power-law ($uzmax^{pl}$, star), Monin-Obukhov theory ($uzmax^{mo}$, cross) by logarithmic law ($uzmax^{ll}$, sum) 10-m wind value (uas , filled triangle) and upper-level winds (ua , filled circles with line), also for the northward component (green). Temporal evolution of wind speed (top right) with all approximations and upper-level winds at the closest vertical level at 100 m (on log-y scale). Maps of both estimations (bottom left and middle) with the blue cross showing the point of previous figures. Wind rose at the given point (bottom right)

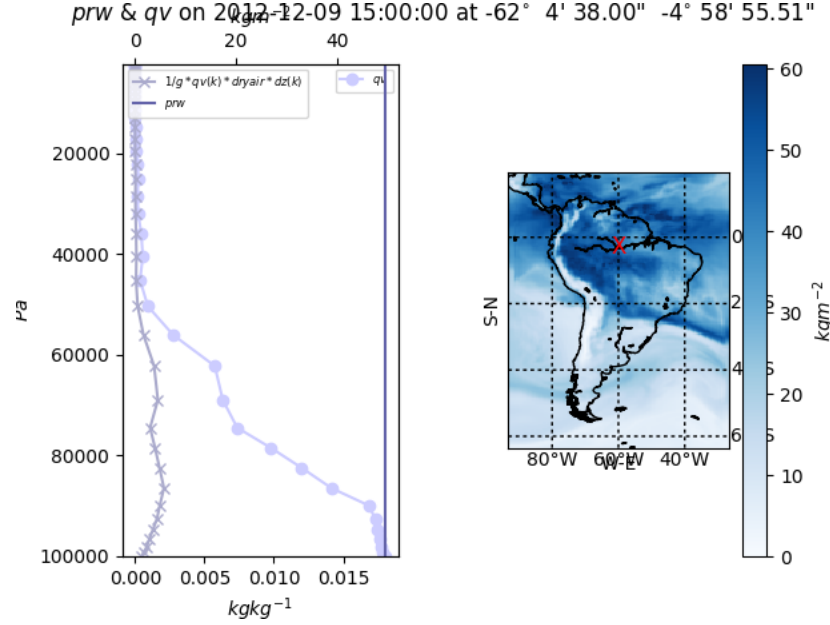


Figure 4: On a given point (left): water path (prw , vertical straight line), vertical profile of water vapour (qv , line with full circles), water pat at each level (line with crosses). Map of water path (right), red cross shows where the vertical is retrieved

3.5 prw: precipitable water or water vapor path

This variable accounts for the column integrated amount of water vapor.

This one is already implemented in a old WRF tool for vertical interpolation called `p_interp.F`. It was modified by L. Fita when he was as post-doc at the ‘Universidad de Cantabria’ related to the `clWRF`. The general equation following WRF standard variables as:

$$prw = \frac{mu + mub}{g} \sum_{iz=1}^{e_vert} QVAPOR[iz] (dnw[iz+1] - dnw[iz]) \quad (5)$$

where mu : perturbation dry air mass in column, mub : base-state dry air mass in column, g : gravity, e_vert : total number of vertical levels, $qvapor$: mixing ratio of water vapour, dnw : full-sigma eta-layer height. See an example on figure 4

3.6 clwvi: condensed water path

This variable provides similar information, but for the liquid condensed water species. It is the same calculation as in 5, but replacing $QVAPOR$ by $QCLOUD + QRAIN$

3.7 clivi: ice water path

This variable provides similar information, but for the liquid condensed water species. It is the same calculation as in 5, but replacing $QVAPOR$ by $QICE + QSNOW + QGRAUPEL + QHAIL$

3.8 clgvi: graupel water path

This variable provides similar information, but for the liquid condensed water species. It is the same calculation as in 5, but replacing $QVAPOR$ by $QGRAUPEL$

3.9 clhvi: hail water path

This variable provides similar information, but for the liquid condensed water species. It is the same calculation as in 5, but replacing *QVAPOR* by *QHAIL*

3.10 psl: sea level pressure

This accounts for the pressure at the sea level (extrapolation of the pressure at the level of the sea). That means the pressure that might be without the presence of orography.

Three different methodologies have been implemented

- One using hydrostatic-Shuelli method already implemented in the module `phys/module_diag_afwa.F` (assuming a constant lapse-rate of $6.5\text{ }^{\circ}\text{km}^{-1}$, see appendix D.1) [`psl_diag` = 1]
- Using smoothed surface pressure and a target upper-level pressure, already implemented in `p_interp.F90` (see appendix D.3) [`psl_diag` = 2]
- ECMWF method taken from LMDZ from the module `pppmer.F90`, following the methodology by Mats Hamrud and Philippe Courtier from ECMWF (see appendix D.2) [`psl_diag` = 3]

These three methodologies have been implemented and can be switched by a new `namelist.input` parameter labeled `psl_diag` (in `cordex` section). Its default value is 3. Even, on using the ‘ptarget’ method (`psl_diag` = 2) one can select the degree of smoothing of the surface place by the selecting the number of times that the smoothing (as the mean of the point and its surrounding 8 neighbors) has to be applied (`psmooth`, default 5) and the upper pressure to be used as target (`ptarget`, default 70000 Pa).

On figure 5 is shown the different outcomes applying each approximation. There are some problems with the ptarget methodology in both psl estimate and borders for each parallel process on applying the smoothing

3.11 cape: convective available potential energy

This variable accounts for all the energy that convectively might be released.

From AMS glossary is described as: (http://glossary.ametsoc.org/wiki/Convective_available_potential_energy)

On a thermodynamic diagram this is called positive area and can be seen as the region between the lifted parcel process curve and the environmental sounding, from the parcel’s level of free convection to its level of neutral buoyancy. CAPE may be expressed as follows:

$$CAPE = \int_{p_f}^{p_n} R_d(T_{vp} - T_{ve})d \ln p \quad (6)$$

where T_{vp} is the virtual temperature of a lifted parcel moving upward moist adiabatically from the level of free convection to the level of neutral buoyancy, T_{ve} is the virtual temperature of the environment, R_d is the specific gas constant for dry air, p_f is the pressure at the level of free convection, and p_n is the pressure at the level of neutral buoyancy. The value depends on whether the moist-adiabatic process is considered to be reversible or irreversible (*conventionally irreversible*, or a *pseudoadiabatic* process in which condensed water immediately falls out of the parcel) and whether the latent heat of freezing is considered (conventionally not). It is assumed that the environment is in hydrostatic balance and that the pressure of the parcel is the same as that of the environment. Virtual temperature is used for the parcel and environment to account for the effect of moisture on air density.

It has been at this stage only the calculation already implemented in WRF inside the module `module_diag_afwa.F` via the function `Buoyancy`, which at the same time it provides: Convective inhibition (CIN), Height at the Level of free convection (ZLFC), Pressure at the Level of free convection (PLFC) and Lifted index (LI)

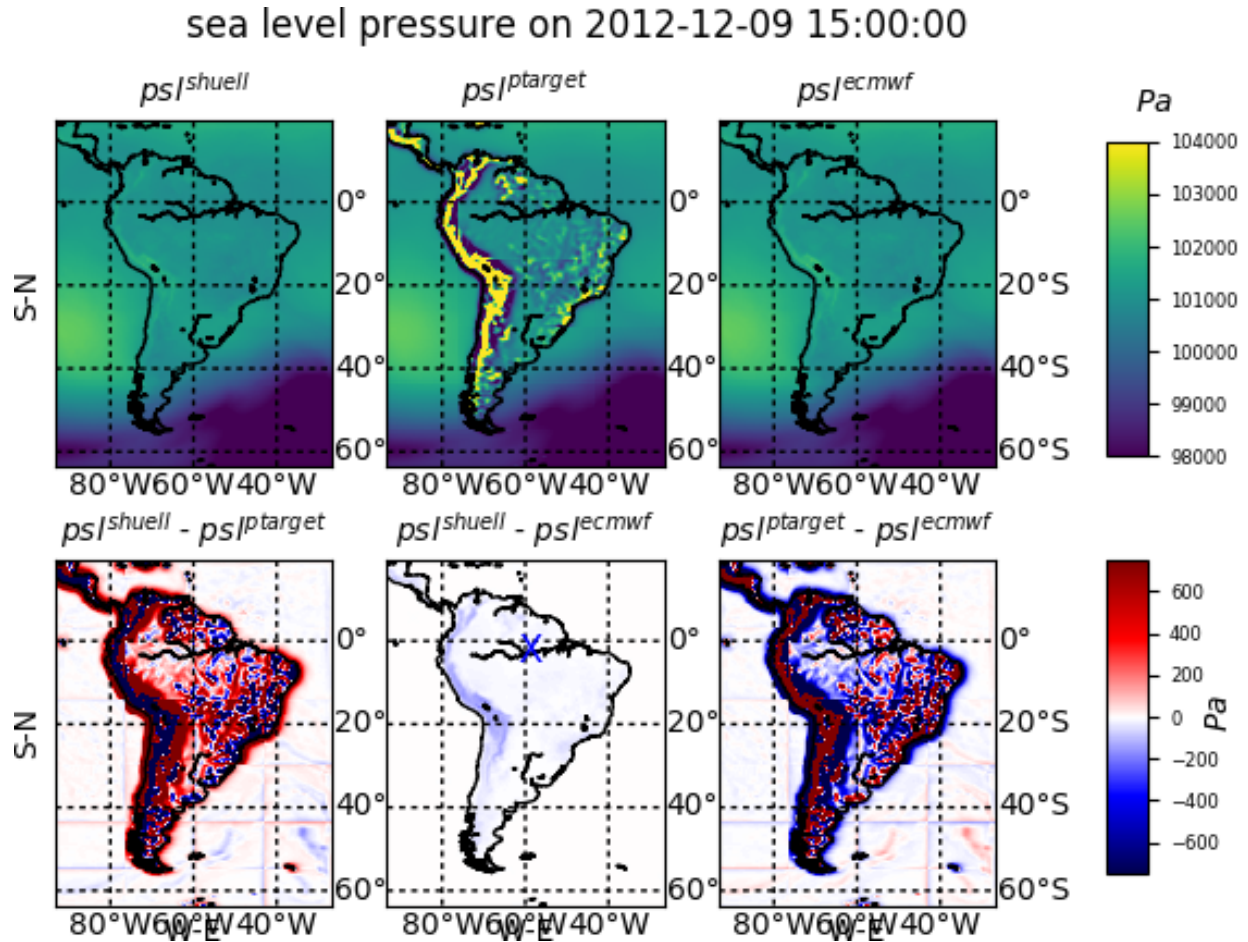


Figure 5: sea level pressure estimates. Following hydrostatic-Shuell method at a given time-step (psl^{shuell} , upper left), p-target ($psl^{ptarget}$, upper middle) and ECMWF (psl^{ecmwf} , upper right). Differences between methods $psl^{shuell} - psl^{ptarget}$ (bottom left), $psl^{shuell} - psl^{ecmwf}$ (bottom middle) and $psl^{ptarget} - psl^{ecmwf}$ (bottom right)

3.12 cin: convective inhibition

This variable accounts for the process which inhibits the convection. Already provided by the implementation of the AFWA's CAPE calculation

From AMS glossary is described as: http://glossary.ametsoc.org/wiki/Convective_inhibition)

The energy needed to lift an air parcel upward adiabatically to the lifting condensation level (*LCL*) and then as a pseudoadiabatic process from the *LCL* to its level of free convection (*LFC*).

For an air parcel possessing positive *CAPE*, the *CIN* represents the negative area on a thermodynamic diagram. The negative area typically arises from the presence of a lid, or the amount of kinetic energy that must be added to a parcel to enable that parcel to reach the *LFC*. Even though other factors may be favorable for development of convection, if convective inhibition is sufficiently large, deep convection will not form. The convective inhibition is expressed (analogously to *CAPE*) as follows:

$$CIN = - \int_{p_i}^{p_f} R_d (T_{vp} - T_{ve}) d \ln p \quad (7)$$

where p_i is the pressure at the level at which the parcel originates, p_f is the pressure at the *LFC*, R_d is the specific gas constant for dry air, T_{vp} is the virtual temperature of the lifted parcel, and T_{ve} is the virtual temperature of the environment. It is assumed that the environment is in hydrostatic balance and that the pressure of the parcel is the same as that of the environment. Virtual temperature is used for the parcel and environment to account for the effect of moisture on air density.

3.13 sund: duration of sunshine

This variable accounts for the time where short-wave radiation is above 120 Wm^{-2} .

It is already implemented in a advance version of the cWRF <http://www.meteo.unican.es/wiki/cordexwrf/SoftwareTools/ClWrf>. But here will directly computed using XIOS (<http://forge.ipsl.jussieu.fr/ioserver/>)

See results of the variable in figure 6

3.14 hur: relative humidity

Relative humidity can be obtained following the Clausius-Clapeyron formula and its approximation from the *August-Roche-Magnus* formula of saturated water vapor pressure e_s

$$e_s = 6.1094 * e^{\frac{17.625 * tempC}{tempC + 243.04}} \quad (8)$$

$$w_s = \frac{0.622 * e_s}{presshPa - e_s} \quad (9)$$

$$hur = \frac{q}{ws * 1000}. \quad (10)$$

being $tempC$: temperature in Celsius degree ($^{\circ}C$), $presshPa$: pressure in hPa , e_s : saturated water vapor pressure, w_s : saturated mixing ratio $kgkg^{-1}$, q : mixing ratio $kgkg^{-1}$

3.15 hus: specific humidity

From the AMS glossary http://glossary.ametsoc.org/wiki/Specific_humidity

$$q = \frac{r_v}{r_v + 1} \quad (11)$$

where r_v : mixing ratio

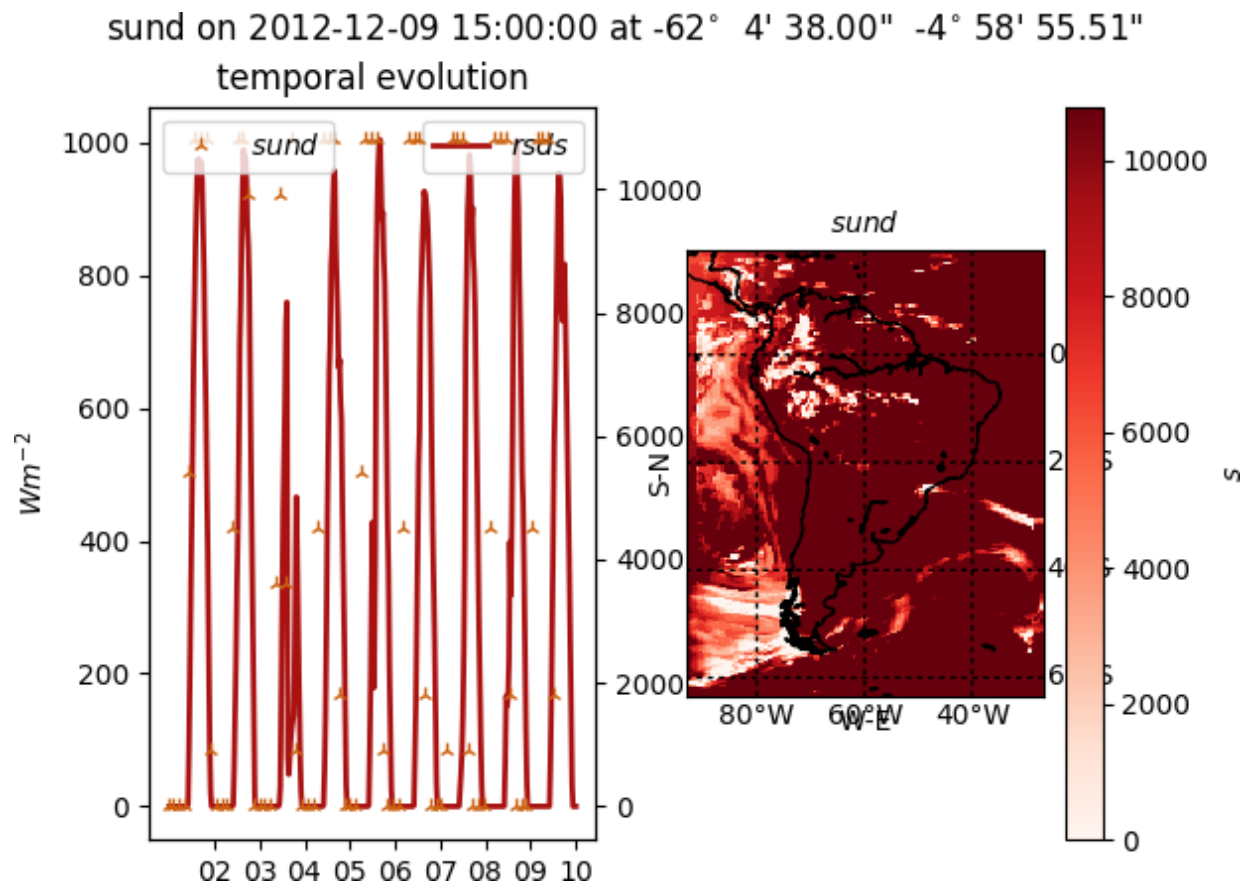


Figure 6: Temporal evolution (left) of shortwave downward radiation (*sdown*, red line, left y-axis) and sunshine duration (*sund*, stars, right y-axis. *sund* map at a given time (right))

3.16 zg: geopotential height

$$zg = PH + PHB \quad (12)$$

where PHB , WRF base geopotential height, P , WRF perturbation geopotential height

3.17 press: air-pressure

$$press = P + PH \quad (13)$$

where PB , WRF base pressure, P , WRF perturbation pressure

3.18 ta: air-temperature

$$ta = (T + 300) \left(\frac{P + PB}{p0} \right)^{R/C_p} \quad (14)$$

where T , WRF temperature (which is as potential temperature), PB , WRF base pressure, P , WRF perturbation pressure, $p0$: pressure reference 100000 Pa

3.19 ua/va: air-wind Earth oriented

$$\begin{cases} ua = U_{unstg} * COSALPHA - V_{unstg} * SINALPHA \\ va = U_{unstg} * SINALPHA + V_{unstg} * COSALPHA \end{cases} \quad (15)$$

where U_{unstg} , unstaggered WRF eastward wind, V_{unstg} , unstaggered WRF northward wind, $COSALPHA$, local cosine of map rotation, $SINALPHA$, local sine of map rotation

3.20 cdgen

Drag coefficient at surface. Computation of drag coefficient depends on selected surface scheme. In order to avoid this scheme dependency, a general calculation of the coefficient has been introduced as it is shown in equation 16, after Garratt (1992).

$$C_d = \left(\frac{u^*}{wss} \right)^2 \quad (16)$$

Being, u^* : from similarity theory, wss 10 m wind speed

3.21 tauuv

Surface Downward wind stress at 10m.

It is implemented following the equation 17, begin C_D drag coefficient. Winds are Earth-rotated. The generic drag coefficient **cdgen** is used to compute these variables.

$$tauw = (C_D uas^2, C_D vas^2) \quad (17)$$

3.22 evspsblpot

Potential evapotranspiration is computed following its computation from ORCHIDEE model (Organising Carbon and Hydrology In Dynamic Ecosystems, <http://orchidee.ipsl.fr/>). The implementation is retrieved from the module `src_sechiba/enerbil.f90` and basically consists no an implementation of the Penman-Monteith formulation (Monteith, 1965). It is a simple formulation (see equation 18)

$$potevap = \rho(1) * qc * (q2_{sat} - qv(1)) \quad (18)$$

$$qc = C_d \sqrt{uas^2 + vas^2} \quad (19)$$

where qc : surface drag coefficient, $q2_{sat}$: Saturated air at 2m (can be assumed to be $q2 == qsf?$), uas, vas : 10 m wind components.

Up to now there is only one implementation and it is selected via namelist parameter `potevap_diag`, up to now only with value 1 for the ORCHIDEE implementation

3.23 rsus

Surface Upwelling Shortwave Radiation, is understood as the shortwave radiation from land. It is provided accumulated by radiation schemes CAM and RRTMG (`sw_ra_scheme = 3,4`) in variable `swupb`. However, it might be re-calculated (if necessary) in a generic way as the reflected shortwave radiation due to albedo as it is shown in equation 20

$$rsus = -albedo * swdown \quad (20)$$

Being, *albedo*: albedo, *swdown*: downward at surface shortwave radiation

3.24 rlus

Surface Upwelling Longwave Radiation, is understood as the longwave radiation from land. It is provided accumulated by radiation schemes CAM and RRTMG (`sw_ra_scheme = 3,4`) in variable `slupb`. However, it might be re-calculated (if necessary) in a generic way as the longwave radiation due to surface temperature following black body formulation as it is shown in equation 21

$$rlus = CtBoltzman * skt^4 \quad (21)$$

Being, *CtBoltzman*: albedo, *skt*: skin temperature

4 Additional variables

Some other variables not required by CORDEX, but might be interesting for other purposes will be also added

4.1 Water vapor balance terms

These covers the different column integrated terms of the water balance equation. The equation of the water vapour budget:

$$\begin{aligned} TEN_q &= HOR_q + VER_q + MP_q \\ \frac{\partial q_q}{\partial t} &= -V_h \vec{\nabla} q_q - w \frac{\partial q_q}{\partial z} + SO_q - SI_q \end{aligned} \quad (22)$$

Where q stands for either of the five water species concentration (vapor, snow, ice, rain and liquid), V_h stands for horizontal wind speed, w stands for the vertical wind speed and MP for the loss or gain of water due to cloud microphysical processes. The term in the left-hand side of the equation represents the water species tendency (TEN or 'PW'), referring to the difference between q at the model previous time step and at the end of the actual time step, divided by the time step. TEN equals to the horizontal advection (HOR or 'F', first term in right-hand side of the equation), the vertical advection (VER or 'Z', second term in right-hand side) and the sources (SO) or sink (SI) of atmospheric water due to microphysical processes (MP). All terms are expressed in $kgkg^{-1}s^{-1}$. However, SO , and SI ca not be provided because they are micro-physics dependent an make difficult to provide a general formula for them.

In order to obtain the total column mass of water due to each term (in units of mm), it is applied to each term of eq. 22 (similarly as in 5):

$$-\frac{1}{g} \int_{p_{sfc}}^{p_{top}} dp \quad (23)$$

Following the methodology of Huang et al. (2014) and Yang et al. (2011), Fita and Flaounas (2017) implemented the water budget terms in a new module in WRF in order to allow the computation of the terms during model integration. For the CORDEX module, only the vertically integrated variables will be implemented. Microphysics processes depends on the micro-physics scheme used during model run. It is know the the budget is closed, thus, residual of the terms must be the micor-physics term. Due to the complexity of each micro-physics scheme and the impossibility to generalize the calculation, the accumulation of diabatic heating from the micro-physics scheme is provided as a proxy.

All water species decomposition is shown in figures 7 and 8

It has also been grouped by vertical levels as it is done with the clouds: $p \geq 68000 Pa$, $40000 \leq p < 68000 Pa$, $p < 40000 Pa$. Decomposition of each term is shown for water vapour qv and snow in figures from 9 to 12.

W.B. AC PW normalized by σ on 2012-12-09 15:00:00

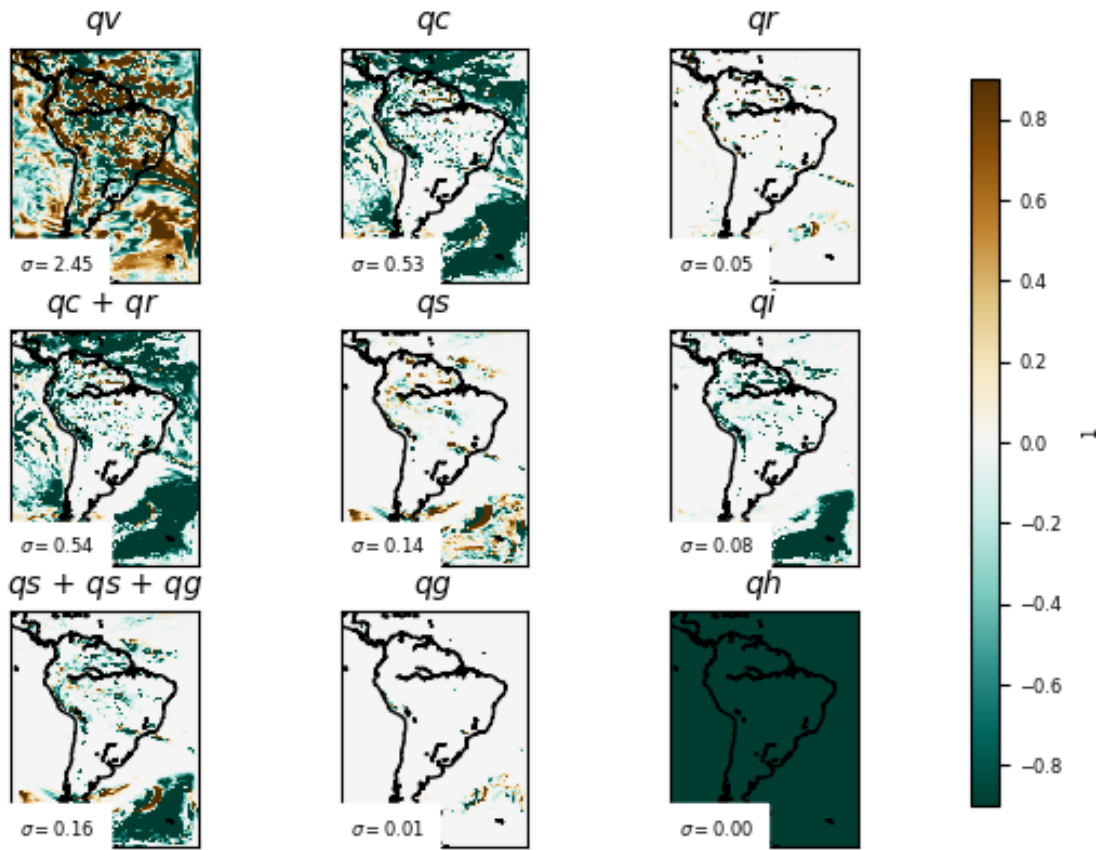


Figure 7: Water budget 3h-accumulated vertically integrated total tendency 'PW' at a given time, for water vapour (qv , top left), cloud (qc , top middle), rain (qr , top right), water condensed species ($qc + qr$, middle left), snow (qs , middle middle), ice (qi , middle right), water solid species ($qs + qi + qg$, bottom left), graupel (qg , bottom middle), hail (qh , bottom right). Number on low left corner of the figure correspond to the standard deviation (σ in mm) value used for the normalization

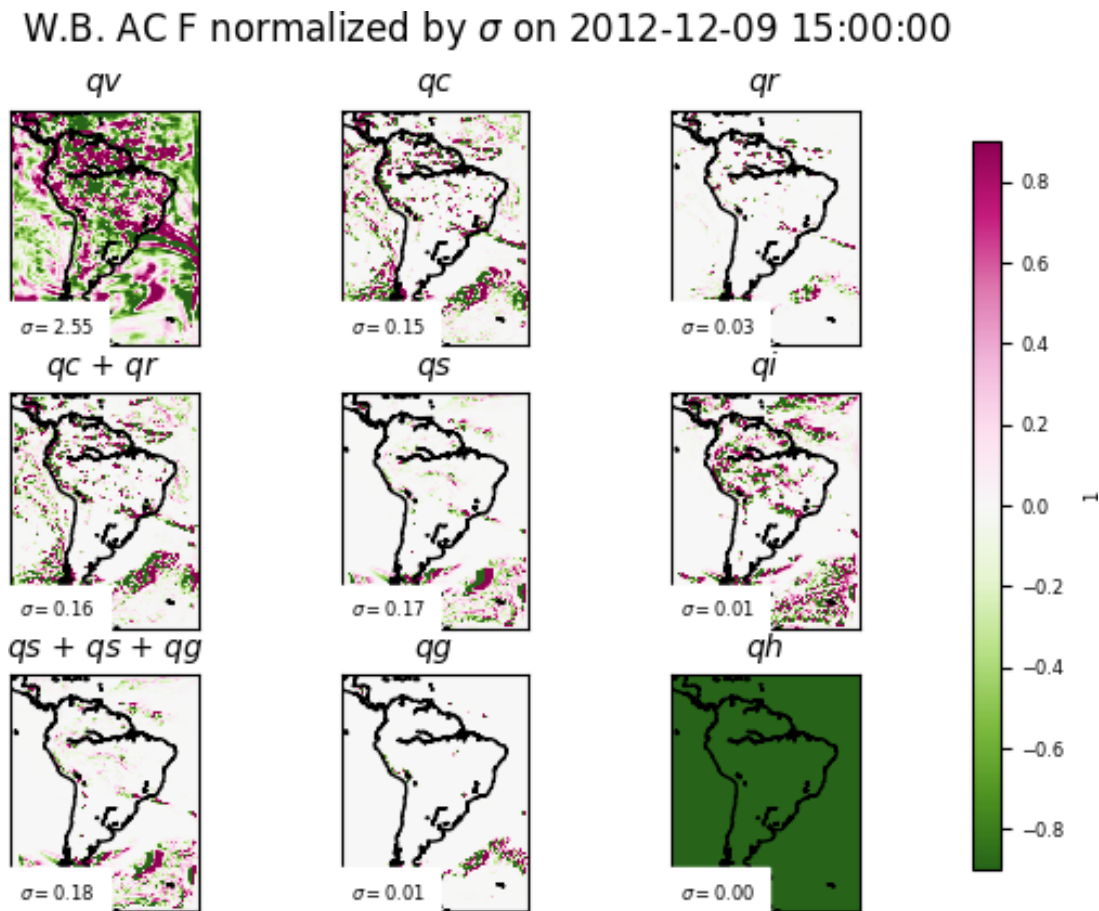


Figure 8: As in 7, but for Water budget 3h-accumulated vertically integrated horizontal advection 'F' at a given time

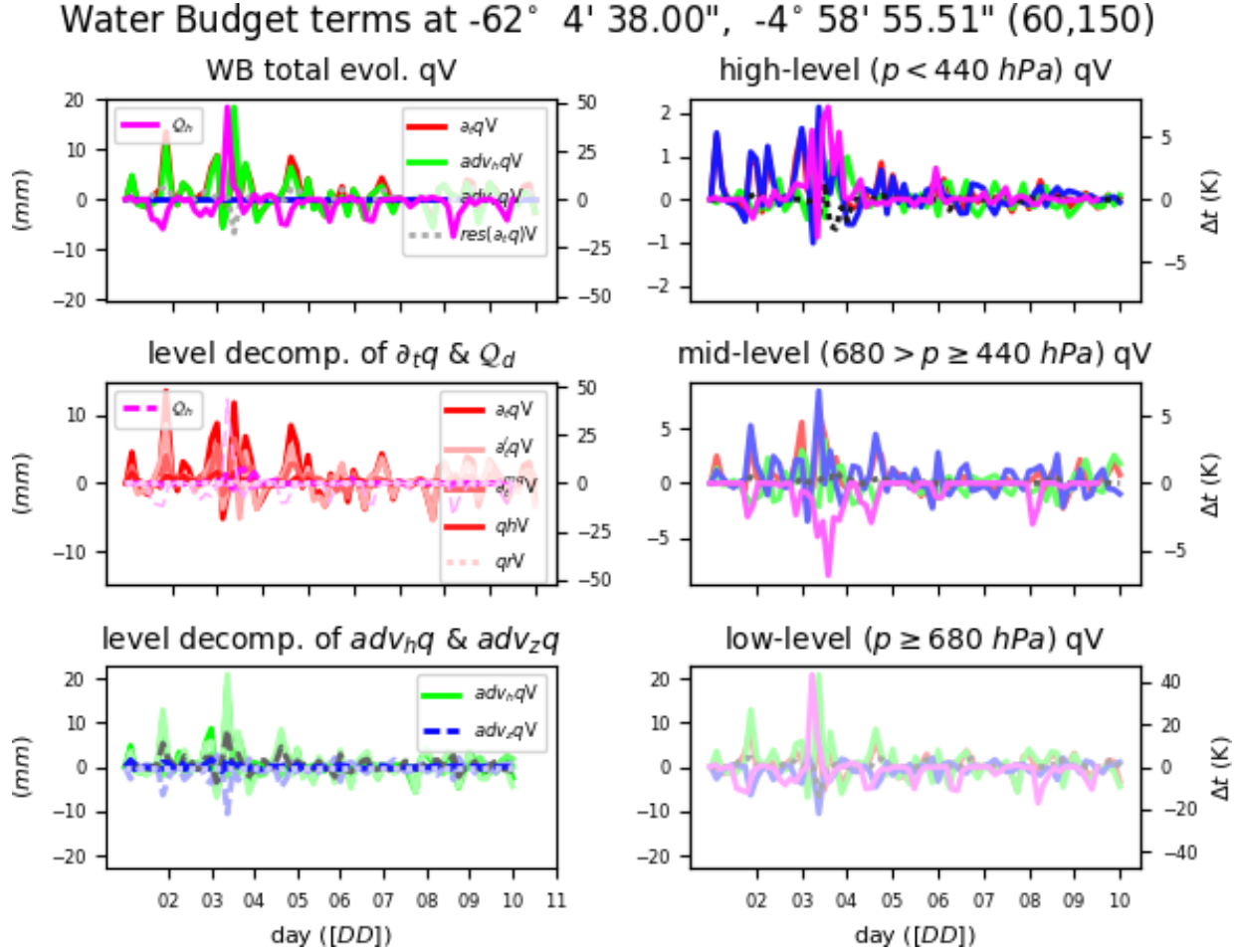


Figure 9: Water budget evolution at a given point for water vapour of vertically integrated water-budget terms: total tendency 'PW' ($\partial_t qv$, red), horizontal advection 'F' ($adv_h qv$, green), vertical advection 'Z' ($adv_z qv$, green), residual PW - F - Z ($res(\partial_t qv)$, gray dashed) and diabatic heating from micro-physics (Q_d , pink) (top left), only high-level vertically integrated values ($p < 440$ hPa, top right), high/mid/low-level (degree of color intensity) decomposition of $partial_t qv$ (red) and Q_d (pink) and their respective residuals as dashed lines (middle left), only mid-level vertically integrated values ($680 > p \leq 440$ hPa, middle right), high/mid/low-level (degree of color intensity) decomposition of $adv_h qv$ (green) and $adv_z qv$ (blue) and their respective residuals as dashed lines (bottom left) and only low-level vertically integrated values ($p \geq 680$ hPa, bottom right)

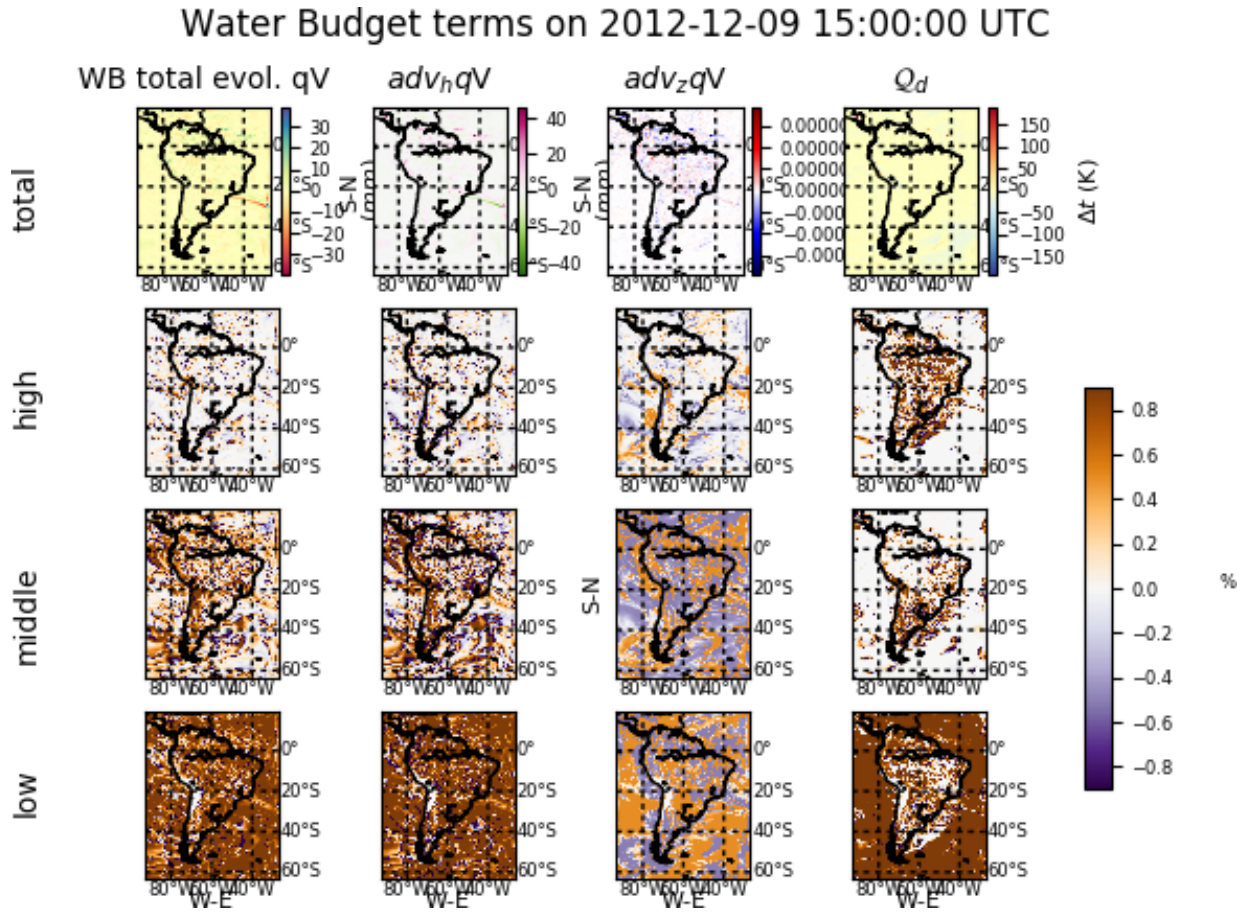


Figure 10: water vapour water budget maps of each component and diabatic heating from micro-physics at a given time and the percentual contribution at each different vertically integrated layer respective the total. total tendency 'PW' ($\partial_t qv$, first column), horizontal advection 'F' ($adv_h qv$, second col), vertical advection 'Z' ($adv_z qv$, third col.) and diabatic heating from micro-physics (Q_d , 4th col). Percentage contribution of high level ($p < 440 \text{ hPa}$) integration to the total (second row), percentage for mid level ($680 > p \geq 440 \text{ hPa}$) integration to the total (third row) and percentage of low-level ($p \geq 680 \text{ hPa}$) integration (bottom row)

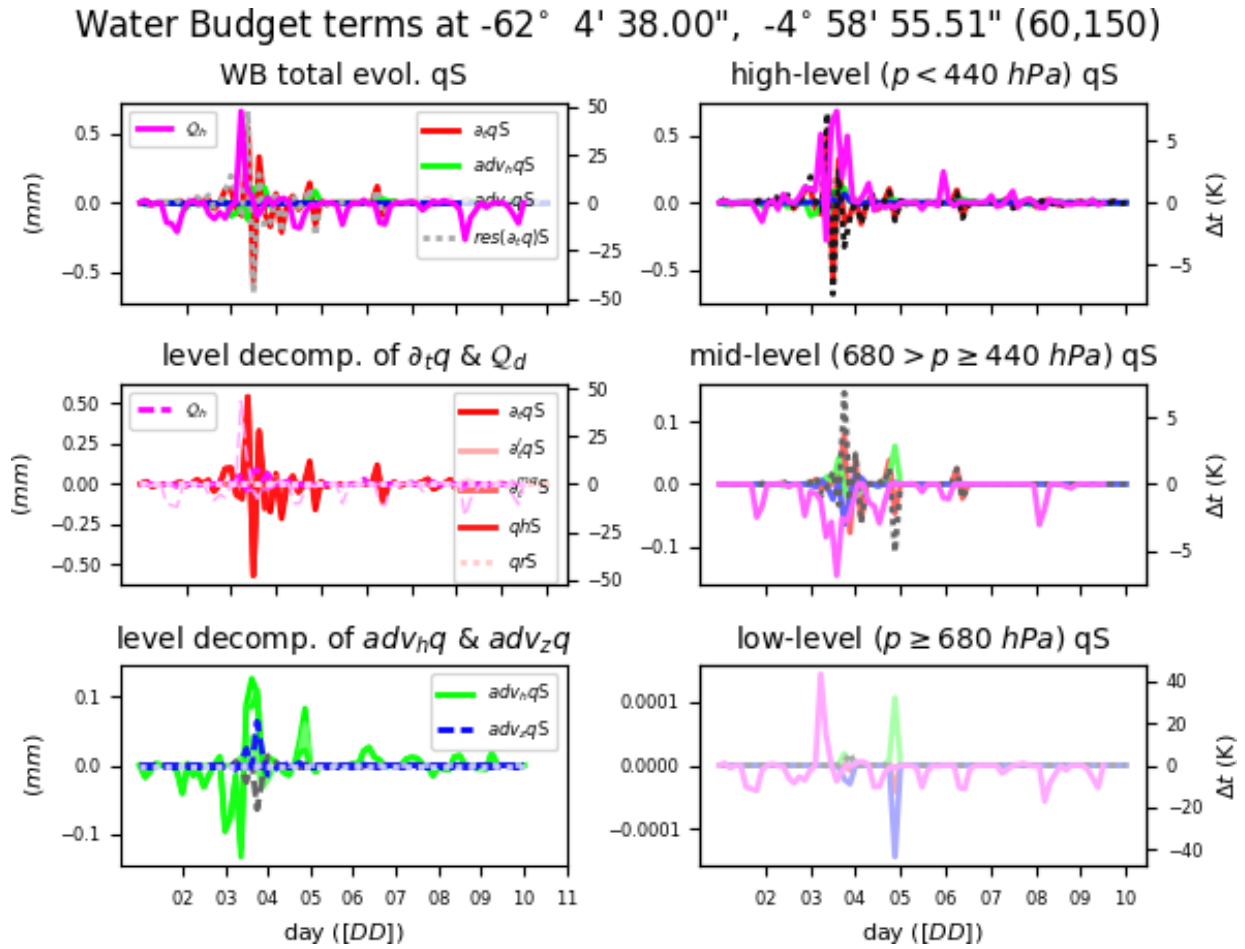
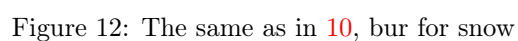


Figure 11: The same as in figure 9, but for snow



4.2 zmlagen: generic boundary layer height

Boundary layer height is a clear example of model dependence and even scheme dependence of how a diagnostic is computed. Each pbl scheme has its own assumptions and has to be compiled in a specific way.

However, one could try to find a general definition as it was done in (García-Díez et al., 2013) after (Nielsen-Gammon et al., 2008). The method consists in searching for the first level where potential temperature exceeds the minimum potential temperature reached in the mixed layer (ML) by more than 1.5 K. It has been implemented as it is shown below

1. Mixed layer depth (k_{MLD}) first layer at which the variation of mixing ratio upwards from first layer value achieves a given percentage: $\frac{|qv(k_{MLD}) - qv(1)|}{qv(1)} > \delta qv$ (here applied a $\delta qv = 0.1$)
2. Minimum potential temperature within the MLD: $\theta_{min_{MLD}} = \min(\theta(1), \dots, \theta(k_{MLD}))$
3. Boundary layer level (k_{zmla}) first level where: $\theta(k_{zmla}) + \delta\theta > \theta_{min_{MLD}}$ (here $\delta\theta = 1.5 K$)
4. Boundary layer height ($zmla$) height above ground ($zagl$): $zmla = zagl(k_{zmla})$

Comparison of this implementation with the $zmla$ directly provided by WRF's pbl scheme is shown in figure 13. No general rule has been applied to determine the correct value of δqv used to determine depth of mixed layer. They can be determined by the `namelist.input` parameters `zmlagen_dqv` for δqv (default value 0.1) and `zmlagen_dtheta` for $\delta\theta$ (default value 1.5 K)

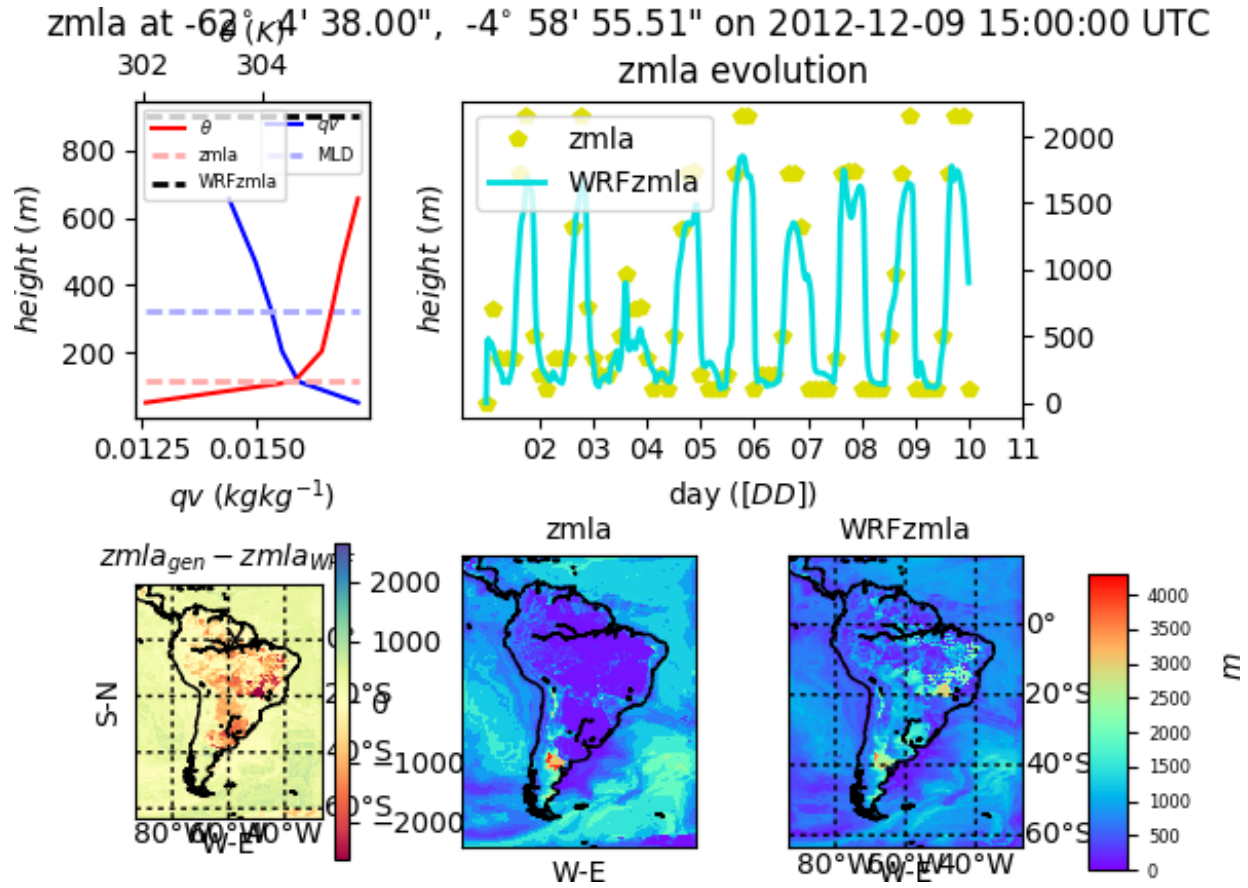


Figure 13: Vertical characteristics of the atmosphere at a given point (top left): potential temperature vertical profile (θ K, red line), vertical profile of mixing ratio (qv $kg\,kg^{-1}$, blue line), mixed layer depth (MLD, dashed horizontal line at 323.522 m), derived boundary layer height (zmla, horizontal dashed line at 107.122 m and WRF derived pbl scheme value (WRF_{zmla} at 903.017 m). Comparison of temporal evolutions (top right) between derived zmla (yellow stars) and WRF's pbl scheme (blue line). Map of differences between derived and WRF simulated ($zmla - zmla_{WRF}$, bottom left), zmla map (bottom middle) and $zmla_{WRF}$ (bottom right)

5 Work done

Following similar experiences like clWRF, the implementation would be done as follows:

1. Introduce the new variables in the `registry.cordex`
2. Reproduce the structure of any of the diagnostics module (e.g. `phys/module_diag_cl.F`) and introduce each computation from the different sources.
3. This accounts for a lot of additional variables, thus all the module related will be activated on computation by a pre-compilation flag called `DIAGCORDEX`
4. Introduction of a new `namelis.input` section for that variables with more than one option
5. Output these variables in a new output file `wrfcdx_d<domain>_<date>`
6. As an additional work, all the instantaneous variables used for the different accumulations and extremes, can also be retrieved. It is only necessary to:
 - Give an output unit on the `registry.cordex` (see instructions at the end of the file)
 - Uncomment in the code (`phys/module_diagnostics_dirver.F` and `module_diag_cordex.F`), the commented lines with the key word: `INSTVALS`
 - re-compile WRF

6 Missing variables

There are certain variables which could not be introduced (yet?)

6.1 wsgsmax100: Daily Maximum Near-Surface Wind Speed of Gust at 100 m

The wind gust at 100 m is understood that should follow a similar implementation than for the `wsgsmax`, but at 100 m, since is understood than an extrapolation of such turbulent phenomena it would require a complete new set of equations. This one is let to open discussion.

6.2 ic_lightning, cg_lightning, tot_lightning: intra-cloud, ground and total lightning flashes

There is lightning scheme implementation in WRF. (`lightning_option` among other from `namelist.input`). It might require some adjustment prior it's use.

It does not sees to provide cloud/ground discrimination

6.3 praccmov

Moving accumulated precipitation values for different temporal thresholds (τ): 30 minutes, 1 hour, 3 hour, 6 hours and 24 hours.

This variable might be useful for the impact studies on infrastructures like bridges. The idea would be to provide its maximum between output times (t_{out}) as it is suggested in the equation [24](#)

$$praccmov\tau(t) = \sum_{it=t-\tau}^t pr(it) \quad (24)$$

$$maxpraccmov\tau = max[praccmov\tau(t_{out} + \delta t), ..., praccmov\tau(t_{out})] \quad (25)$$

7 Others

It will be some other hard work to do related to it.

7.1 New variables

Pretty sure that as we get closer to stake-holders, decision makers, impact and mitigation communities more variables will arise... keep in touch !?

7.2 CF-compilant file

WRF does not provide a real CF-compilant file format. It would be necessary to add at least at the output (at least on the `wrfcdx_d<domain>_<date>` file):

- `time` variable: CF-version of variable with times in the file
- `atrrIBUTES`: WRF does not provide variables with standard attributes like: `standar_name`, `long_name`, ...

7.3 Optimization

Avoid the use of namelist options and got the variables/method directly without the introduction of ifs which might make WRF run slowly. This could be done directly via pre-compilation flags, using for example, the namelist options as pre-compilation options?

Acknowledgements All the coders of WRF, LMDZ, ORCHIDEE are acknowledged for their work on the developing and maintaining of the models. M. A. Jiménez from Universitat de les Illes Balears is acknowledged by her explanations on certain PBL calculations.

References

- Brasseur, O. (2001). Development and application of a physical approach to estimating wind gusts. *Monthly Weather Review*, 129(1):5–25.
- Fita, L., Fernández, J., and García-Díez, M. (2010). Clwrf: Wrf modifications for regional climate simulation under future scenarios. *Proceedings of 11th WRF Users' Workshop*.
- Fita, L. and Flaounas, E. (2017). Medicanes as subtropical cyclones: the december 2005 case from the perspective of surface pressure tendency diagnostics and atmospheric water budget. *Q. J. Royal Met. Soc.*, under revision.
- García-Díez, M., Fernández, J., Fita, L., and Yagüe, C. (2013). Seasonal dependence of wrf model biases and sensitivity to pbl schemes over europe. *Q. J. of Roy. Met. Soc.*, 139:501–514.
- Garratt, J. (1992). *The Atmospheric Boundary Layer*. Cambridge Univ. Press, Cambridge, U.K.
- Hourdin, F., Musat, I., Bony, S., Braconnot, P., Codron, F., Dufresne, J.-L., Fairhead, L., Filiberti, M.-A., Friedlingstein, P., Grandpeix, J.-Y., Krinner, G., LeVan, P., Li, Z.-X., and Lott, F. (2006). The lmdz4 general circulation model: climate performance and sensitivity to parametrized physics with emphasis on tropical convection. *Clim. Dyn.*, 27(7-8):787–813.
- Huang, H.-L., Yang, M.-J., and Sui, C.-H. (2014). Water budget and precipitation efficiency of typhoon morakot (2009). *J. Atmos. Sci.*, 71:112–129.
- Jourdir, B. (2015). Ressource éolienne en france métropolitaine : méthodes d'évaluation du potentiel, variabilité et tendances. *Climatologie: École Doctorale Polytechnique, 2015. Français. ph:+33 01238226*, pages 1–229.
- Monteith, J. L. (1965). Evaporation and environment. the state and movement of water in living organisms. *19th Symp. Soc. Exp. Biol*, pages 205–234.

- Nielsen-Gammon, J. W., Powell, C. L., Mahoney, M. J., Angevine, W. M., Senff, C., White, A., Berkowitz, C., Doran, C., and Knupp, K. (2008). Multisensor estimation of mixing heights over a coastal city. *Journal of Applied Meteorology and Climatology*, 47(1):27–43.
- Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Duda, D. M. B. M. G., Huang, X.-Y., Wang, W., and Powers, J. G. (2008). A description of the advanced research wrf version 3. *NCAR TECHNICAL NOTE*, 475:NCAR/TN-475+STR.
- Yang, M. J., Braun, S. A., and Chen, D.-S. (2011). Water budget of typhoon nari (2001). *Mon. Weather Rev.*, 139:3809–3828.

A clt

A.1 code from LMDZ

This is the main core of the calculation of clt from LMDZ extracted from subroutine `newmicro.f90` and implemented in PyNCplot.

```

FUNCTION var_clt(clfra, dz)
(...)
  REAL(r_k), PARAMETER                :: ZEPSEC=1.0D-12
  REAL(r_k), PARAMETER                :: zero=0.d0
  REAL(r_k), PARAMETER                :: one=1.d0

(...)

  zclear = one
  zcloud = zero

  DO iz=1,dz
    zclear = zclear*(one-MAX(clfra(iz),zcloud))/(one-MIN(zcloud,1.-ZEPSEC))
    var_clt = one - zclear
    zcloud = clfra(iz)
  END DO
FUNCTION var_clt

```

B cllmh

B.1 cllmh: code from LMDZ

This is the main core of the calculation of cllmh from LMDZ extracted from subroutine `newmicro.f90` and implemented in PyNCplot.

```

FUNCTION var_cllmh(clfra, p, dz)
(...)
! Low limit pressure for medium clouds [Pa]
  REAL(r_k), PARAMETER                :: prmhc = 44000.d0
! Low limit pressure for High clouds [Pa]
  REAL(r_k), PARAMETER                :: prmlc = 68000.d0

(...)
  REAL(r_k), DIMENSION(3)             :: var_cllmh

(...)
  zclearl = one
  zcloudl = zero
  zclearm = one
  zcloudm = zero
  zclearh = one
  zcloudh = zero

  var_cllmh = one

  DO iz=1, dz
    IF (p(iz) < prmhc) THEN
      var_cllmh(3) = var_cllmh(3)*(one-MAX(clfra(iz),zcloudh))/(one-MIN(zcloudh,one-ZEPSEC))

```

```

      zcloudh = clfra(iz)
    ELSE IF ( (p(iz) >= prmh) .AND. (p(iz) < prmlc) ) THEN
      var_cllmh(2) = var_cllmh(2)*(one-MAX(clfra(iz),zcloudm))/(one-MIN(zcloudm,one-ZEPSEC))
      zcloudm = clfra(iz)
    ELSE IF (p(iz) >= prmlc) THEN
      var_cllmh(1) = var_cllmh(1)*(one-MAX(clfra(iz),zcloudl))/(one-MIN(zcloudl,one-ZEPSEC))
      zcloudl = clfra(iz)
    ELSE
      PRINT *, ' ' // TRIM(fname) // ': This is weird, pressure:', p(iz), ' Pa fails out!!'
      PRINT *, '   from high, low cloud pressures:', prmh, ' ', prmlc, ' Pa at z-level:', iz
      PRINT *, '   p_high > p:', prmh, '> ', p(iz), ' Pa'
      PRINT *, '   p_low > p >= p_high:', prmlc, '> ', p(iz), ' >= ', prmh, ' Pa'
      PRINT *, '   p_low >= p:', prmlc, '>= ', p(iz), ' Pa'
      STOP
    END IF
  END DO

  var_cllmh = one - var_cllmh

```

C wsgsmax

C.1 afwa

```

! Calculate the max 10 m wind speed between output times
! -----
! UPDATE 20150112 - GAC
! Diagnose from model 10 m winds, and blend with 1 km AGL
! winds when precipitation rate is > 50 mm/hr to account
! for increased surface wind gust potential when precip
! is heavy and when winds aloft are strong. Will use the
! higher of the surface and the blended winds. Blending
! is linear weighted between 50-150 mm/hr precip rates.
! -----
DO j = jms, jme
  DO i = ims, ime
    wind_vel = uv_wind ( grid % u10(i,j) , grid % v10(i,j) )
    prate_mm_per_hr = ( grid % afwa_precip(i,j) / grid % dt ) * 3600.

    ! Is this an area of heavy precip? Calculate 1km winds to blend down
    ! -----
    IF ( prate_mm_per_hr .GT. 50. ) THEN
      is_target_level=.false.
      DO k=kms,kme
        IF ( ( zagl(i,k,j) >= 1000. ) .and. &
              ( .NOT. is_target_level ) .and. &
              ( k .ne. kms ) ) THEN
          is_target_level = .true.
          u1km = u_phy(i,k-1,j) + (1000. - (zagl(i,k-1,j))) &
                * ((u_phy(i,k,j) - u_phy(i,k-1,j))/(zagl(i,k,j)))
          v1km = v_phy(i,k-1,j) + (1000. - (zagl(i,k-1,j))) &
                * ((v_phy(i,k,j) - v_phy(i,k-1,j))/(zagl(i,k,j)))
          EXIT ! We've found our level, break the loop
        ENDIF
      END DO
    END IF
  END DO

```

```

ENDDO

! Compute blended wind
! -----
factor = MAX ( ( ( 150. - prate_mm_per_hr ) / 100. ), 0. )
ublend = grid % u10(i,j) * factor + u1km * (1. - factor)
vblend = grid % v10(i,j) * factor + v1km * (1. - factor)
wind_blend = uv_wind ( ublend, vblend )

! Set the surface wind to the blended wind if higher
! -----
IF ( wind_blend .GT. wind_vel ) THEN
    wind_vel = wind_blend
ENDIF
ENDIF

IF ( wind_vel .GT. grid % wspd10max(i,j) ) THEN
    grid % wspd10max(i,j) = wind_vel
ENDIF
ENDDO
ENDDO

```

C.2 lmdz

```

!-----gustiness calculation-----!
IF (iflag_gusts==0) THEN
    gustiness(1:klon)=0
ELSE IF (iflag_gusts==1) THEN
    gustiness(1:klon)=f_gust_bl*ale_bl(1:klon)+f_gust_wk*ale_wake(1:klon)
ELSE IF (iflag_gusts==2) THEN
    gustiness(1:klon)=f_gust_bl*ale_bl_stat(1:klon)+f_gust_wk*ale_wake(1:klon)
    ! ELSE IF (iflag_gusts==2) THEN
    !     do i = 1, klon
    !         gustiness(i)=f_gust_bl*ale_bl(i)+sigma_wk(i)*f_gust_wk&
    !             *ale_wake(i) !! need to make sigma_wk accessible here
    !     enddo
    ! ELSE IF (iflag_gusts==3) THEN
    !     do i = 1, klon
    !         gustiness(i)=f_gust_bl*alp_bl(i)+f_gust_wk*alp_wake(i)
    !     enddo
ENDIF

```

D psl

D.1 afwa

```

MSLP = PSFC
!
! COMPUTE LAYER TAU (VIRTUAL TEMP*RD/G).
TVRT = TLEV1*(1.0+0.608*QLEV1)
!TAU = TVRT*RD*GI
!
! COMPUTE TAU AT THE GROUND (Z=ZSFC) AND SEA LEVEL (Z=0)
! ASSUMING A CONSTANT LAPSE RATE OF GAMMA=6.5DEG/KM.

```

```

      TVRSFC = TVRT + (ZLEV1 - ZSFC)*GAMMA
      TAUSFC = TVRSFC*RD*GI
      TVRSL  = TVRT + (ZLEV1 - ZSL)*GAMMA
      TAUSL  = TVRSL*RD*GI
!
!      IF NEED BE APPLY SHEULL CORRECTION.
      IF ((TAUSL.GT.TAUCR).AND.(TAUSFC.LE.TAUCR)) THEN
        TAUSL=TAUCR
      ELSEIF ((TAUSL.GT.TAUCR).AND.(TAUSFC.GT.TAUCR)) THEN
        TAUSL = TAUCR-CONST*(TAUSFC-TAUCR)**2
      ENDIF
!
!      COMPUTE MEAN TAU.
      TAUAVG = 0.5*(TAUSL+TAUSFC)
!
!      COMPUTE SEA LEVEL PRESSURE.
      MSLP = PSFC*EXP(ZSFC/TAUAVG)

```

D.2 lmdz

```

!>=====
!!
!! Auteur(s) I.Musat (LMD/CNRS) date: 20151106
!!
!! Objet: Calcul pression au niveau de la mer cf. Arpege-IFS
!! ctstar: calcule la temperature de l'air a la surface (tasfc) et
!!          la temperature de l'air standard a la surface (tastd)
!! pppmer: calcule la slp a partir de tasfc, tastd, de la pression a la surface (pab1)
!!          et du geopotentiel de la surface
!!=====
!! nlon--input-R-temperature au milieu de chaque couche (en K)
!! t--input-R-temperature au milieu de chaque couche (en K)
!! pab--input-R-pression pour chaque inter-couche (en Pa)
!! pal---input-R-pression pour le mileu de chaque couche (en Pa)
!! pphis---input-R-geopotentiel du sol (en m2/s2)
!! tasfc---output-R-temperature air au sol (en K)
!! tastd---output-R-temperature air 'standard' au sol (en K)
!! pmer---output-R-pression au niveau de la mer (en Pa)
!!=====

```

D.3 p_interp

```

N = 1.0

expon=287.04*.0065/9.81

! Fill in missing values
IF ( extrapolate == 0 ) RETURN      !! no extrapolation - we are out of here

! First find where about 400 hPa is located
kk = 0
find_kk : do k = 1, num_metgrid_levels
  kk = k
  if ( interp_levels(k) <= 40000. ) exit find_kk
end do find_kk

```

```

data_out=0.
do itt = 1, ito
  do j = 1, iy
    do i = 1, ix

!           We are below both the ground and the lowest data level.

!           First, find the model level that is closest to a "target" pressure
!           level, where the "target" pressure is delta-p less than the local
!           value of a horizontally smoothed surface pressure field. We use
!           delta-p = 150 hPa here. A standard lapse rate temperature profile
!           passing through the temperature at this model level will be used
!           to define the temperature profile below ground. This is similar
!           to the Benjamin and Miller (1990) method, using
!           700 hPa everywhere for the "target" pressure.

!           ptarget = (psfc(i,j,itt)*.01) - 150.
ptarget = 700.
dpmin=1.e4
kupper = 0
loop_kIN : do kin=iz,1,-1
  kupper = kin
  dp=abs( (pres_field(i,j,kin,itt)*.01) - ptarget )
  if (dp.gt.dpmin) exit loop_kIN
  dpmin=min(dpmin,dp)
enddo loop_kIN
ptarget=ptarget*100.
!       pbot=max(pres_field(i,j,1,itt),psfc(i,j,itt))
!       zbot=0.

!       tbotextrap=tk(i,j,kupper,itt)*(pbot/pres_field(i,j,kupper,itt))**expon
!       tvbotextrap=virtual(tbotextrap,qv(i,j,1,itt))

!       data_out(i,j,itt,1) = (zbot+tvbotextrap/.0065*(1.-(interp_levels(1)/pbot)**expon))
tbotextrap=tk(i,j,kupper,itt)*(psfc(i,j,itt)/ptarget)**expon
tvbotextrap=virtual(tbotextrap,qv(i,j,kupper,itt))
data_out(i,j,itt,1) = psfc(i,j,itt)*((tvbotextrap+0.0065*ter(i,j))/tvbotextrap)**(1/expon)
!!       IF (i==ix/2 .AND. j==iy/2 ) THEN
!       IF (ter(i,j) > 2500.) THEN
!!       PRINT *,itt,' ptarget',ptarget,'kupper:',kupper
!!       PRINT *,'tk:',tk(i,j,kupper,itt),'psfc:',psfc(i,j,itt)
!!       PRINT *,'tbot:',tbotextrap,'tvbot:',tvbotextrap,'ter:',ter(i,j)
!!       PRINT *,'qv:',qv(i,j,kupper,itt),'mslp:',data_out(i,j,itt,1)
!!       ENDIF

      enddo
    enddo
  enddo

```

E cape

E.1 afwa

```

!~ Thermo / dynamical constants
! -----
REAL                :: Rd                !~ Dry gas constant
PARAMETER ( Rd = 287.058 )              !~ J deg-1 kg-1
REAL                :: Cp                !~ Specific heat constant pressure
PARAMETER ( Cp = 1004.67 )              !~ J deg-1 kg-1
REAL                :: g                 !~ Acceleration due to gravity
PARAMETER ( g = 9.80665 )               !~ m s-2
REAL                :: RUNDEF
PARAMETER ( RUNDEF = -9.999E30 )

!~ Initialize variables
! -----
ostat = 0
CAPE = REAL ( 0 )
CIN = REAL ( 0 )
ZLFC = RUNDEF
PLFC = RUNDEF

!~ Look for submitted parcel definition
!~ 1 = Most unstable
!~ 2 = Mean layer
!~ 3 = Surface based
! -----
IF ( parcel > 3 .or. parcel < 1 ) THEN
    prcl = 1
ELSE
    prcl = parcel
END IF

!~ Inititalize our parcel to be (sort of) surface based. Because of
!~ issues we've been observing in the WRF model, specifically with
!~ excessive surface moisture values at the surface, using a true
!~ surface based parcel is resulting a more unstable environment
!~ than is actually occuring. To address this, our surface parcel
!~ is now going to be defined as the parcel between 25-50 hPa
!~ above the surface. UPDATE - now that this routine is in WRF,
!~ going to trust surface info. GAC 20140415
! -----

!~ Compute mixing ratio values for the layer
! -----
DO k = sfc, nz
    ws ( k ) = SaturationMixingRatio ( tK(k), p(k) )
    w ( k ) = ( rh(k)/100.0 ) * ws ( k )
END DO

srclev = sfc
srctK = tK ( sfc )
srcrh = rh ( sfc )
srcp = p ( sfc )

```



```

srcws      = ws      ( sfc )
srcw       = w       ( sfc )
srctheta   = Theta ( tK(sfc), p(sfc)/100.0 )

!~ Compute the profile mixing ratio.  If the parcel is the MU parcel,
!~ define our parcel to be the most unstable parcel within the lowest
!~ 180 mb.
!  -----
mlev = sfc + 1
DO k = sfc + 1, nz

    !~ Identify the last layer within 100 hPa of the surface
    !  -----
    pdiff = ( p (sfc) - p (k) ) / REAL ( 100 )
    IF ( pdiff <= REAL (100) ) mlev = k

    !~ If we've made it past the lowest 180 hPa, exit the loop
    !  -----
    IF ( pdiff >= REAL (180) ) EXIT

    IF ( prcl == 1 ) THEN
        !IF ( (p(k) > 70000.0) .and. (w(k) > srcw) ) THEN
        IF ( (w(k) > srcw) ) THEN
            srctheta = Theta ( tK(k), p(k)/100.0 )
            srcw = w ( k )
            srclev = k
            srctK = tK ( k )
            srcrh = rh ( k )
            srcp = p ( k )
        END IF
    END IF

END DO

!~ If we want the mean layer parcel, compute the mean values in the
!~ lowest 100 hPa.
!  -----
lyrcnt = mlev - sfc + 1
IF ( prcl == 2 ) THEN

    srclev = sfc
    srctK = SUM ( tK (sfc:mlev) ) / REAL ( lyrcnt )
    srcw = SUM ( w (sfc:mlev) ) / REAL ( lyrcnt )
    srcrh = SUM ( rh (sfc:mlev) ) / REAL ( lyrcnt )
    srcp = SUM ( p (sfc:mlev) ) / REAL ( lyrcnt )
    srctheta = Theta ( srctK, srcp/100. )

END IF

srcthetaeK = Thetae ( srctK, srcp/100.0, srcrh, srcw )

!~ Calculate temperature and pressure of the LCL
!  -----
tlclK = TLCL ( tK(srclev), rh(srclev) )

```

```

plcl = p(srclev) * ( (tlclK/tK(srclev))**(Cp/Rd) )

!~ Now lift the parcel
!  -----

buoy = REAL ( 0 )
pw = srcw
wflag = .false.
DO k = srclev, nz
  IF ( p (k) <= plcl ) THEN

    !~ The first level after we pass the LCL, we're still going to
    !~ lift the parcel dry adiabatically, as we haven't added the
    !~ the required code to switch between the dry adiabatic and moist
    !~ adiabatic cooling. Since the dry version results in a greater
    !~ temperature loss, doing that for the first step so we don't over
    !~ guesstimate the instability.
    !  -----

    IF ( wflag ) THEN
      flag = .false.

      !~ Above the LCL, our parcel is now undergoing moist adiabatic
      !~ cooling. Because of the latent heating being undergone as
      !~ the parcel rises above the LFC, must iterative solve for the
      !~ parcel temperature using equivalent potential temperature,
      !~ which is conserved during both dry adiabatic and
      !~ pseudoadiabatic displacements.
      !  -----
      ptK = The2T ( srcthetak, p(k), flag )

      !~ Calculate the parcel mixing ratio, which is now changing
      !~ as we condense moisture out of the parcel, and is equivalent
      !~ to the saturation mixing ratio, since we are, in theory, at
      !~ saturation.
      !  -----
      pw = SaturationMixingRatio ( ptK, p(k) )

      !~ Now we can calculate the virtual temperature of the parcel
      !~ and the surrounding environment to assess the buoyancy.
      !  -----
      ptvK = VirtualTemperature ( ptK, pw )
      tvK = VirtualTemperature ( tK (k), w (k) )

      !~ Modification to account for water loading
      !  -----
      freeze = 0.033 * ( 263.15 - pTvK )
      IF ( freeze > 1.0 ) freeze = 1.0
      IF ( freeze < 0.0 ) freeze = 0.0

      !~ Approximate how much of the water vapor has condensed out
      !~ of the parcel at this level
      !  -----
      freeze = freeze * 333700.0 * ( srcw - pw ) / 1005.7

```

```

    pTvK = pTvK - pTvK * ( srcw - pw ) + freeze
    dTvK ( k ) = ptvK - tvK
    buoy ( k ) = g * ( dTvK ( k ) / tvK )

ELSE

    !~ Since the theta remains constant whilst undergoing dry
    !~ adiabatic processes, can back out the parcel temperature
    !~ from potential temperature below the LCL
    ! -----
    ptK   = srctheta / ( 100000.0/p(k) )**(Rd/Cp)

    !~ Grab the parcel virtual temperture, can use the source
    !~ mixing ratio since we are undergoing dry adiabatic cooling
    ! -----
    ptvK = VirtualTemperature ( ptK, srcw )

    !~ Virtual temperature of the environment
    ! -----
    tvK   = VirtualTemperature ( tK (k), w (k) )

    !~ Buoyancy at this level
    ! -----
    dTvK ( k ) = ptvK - tvK
    buoy ( k ) = g * ( dtvK ( k ) / tvK )

    wflag = .true.

END IF

ELSE

    !~ Since the theta remains constant whilst undergoing dry
    !~ adiabatic processes, can back out the parcel temperature
    !~ from potential temperature below the LCL
    ! -----
    ptK   = srctheta / ( 100000.0/p(k) )**(Rd/Cp)

    !~ Grab the parcel virtual temperture, can use the source
    !~ mixing ratio since we are undergoing dry adiabatic cooling
    ! -----
    ptvK = VirtualTemperature ( ptK, srcw )

    !~ Virtual temperature of the environment
    ! -----
    tvK   = VirtualTemperature ( tK (k), w (k) )

    !~ Buoyancy at this level
    ! -----
    dTvK ( k ) = ptvK - tvK
    buoy ( k ) = g * ( dtvK ( k ) / tvK )

END IF

```

```

      !~ Chirp
      ! -----
!      WRITE ( *, '(I15,6F15.3)' ) k,p(k)/100.,ptK,pw*1000.,ptvK,tvK,buoy(k)

END DO

!~ Add up the buoyancies, find the LFC
! -----
flag = .false.
lfclev = -1
nbuoy = REAL ( 0 )
pbuoy = REAL ( 0 )
DO k = sfc + 1, nz
  IF ( tK (k) < 253.15 ) EXIT
  CAPE = CAPE + MAX ( buoy (k), 0.0 ) * ( hgt (k) - hgt (k-1) )
  CIN = CIN + MIN ( buoy (k), 0.0 ) * ( hgt (k) - hgt (k-1) )

  !~ If we've already passed the LFC
  ! -----
  IF ( flag .and. buoy (k) > REAL (0) ) THEN
    pbuoy = pbuoy + buoy (k)
  END IF

  !~ We are buoyant now - passed the LFC
  ! -----
  IF ( .not. flag .and. buoy (k) > REAL (0) .and. p (k) < plcl ) THEN
    flag = .true.
    pbuoy = pbuoy + buoy (k)
    lfclev = k
  END IF

  !~ If we think we've passed the LFC, but encounter a negative layer
  !~ start adding it up.
  ! -----
  IF ( flag .and. buoy (k) < REAL (0) ) THEN
    nbuoy = nbuoy + buoy (k)

    !~ If the accumulated negative buoyancy is greater than the
    !~ positive buoyancy, then we are capped off. Got to go higher
    !~ to find the LFC. Reset positive and negative buoyancy summations
    ! -----
    IF ( ABS (nbuoy) > pbuoy ) THEN
      flag = .false.
      nbuoy = REAL ( 0 )
      pbuoy = REAL ( 0 )
      lfclev = -1
    END IF
  END IF

END DO

!~ Calculate lifted index by interpolating difference between
!~ parcel and ambient Tv to 500mb.

```

```

! -----
DO k = sfc + 1, nz

    pm = 50000.
    pu = p ( k )
    pd = p ( k - 1 )

    !~ If we're already above 500mb just set lifted index to 0.
    !~ -----
    IF ( pd .le. pm ) THEN
        lidx = 0.
        EXIT

    ELSEIF ( pu .le. pm .and. pd .gt. pm ) THEN

        !~ Found trapping pressure: up, middle, down.
        !~ We are doing first order interpolation.
        ! -----
        lidxu = -dTvk ( k ) * ( pu / 100000. ) ** (Rd/Cp)
        lidxd = -dTvk ( k-1 ) * ( pd / 100000. ) ** (Rd/Cp)
        lidx = ( lidxu * (pm-pd) + lidxd * (pu-pm) ) / (pu-pd)
        EXIT

    ENDIF

END DO

!~ Assuming the the LFC is at a pressure level for now
! -----
IF ( lfclev > 0 ) THEN
    PLFC = p ( lfclev )
    ZLFC = hgt ( lfclev )
END IF

IF ( PLFC /= PLFC .OR. PLFC < REAL ( 0 ) ) THEN
    PLFC = REAL ( -1 )
    ZLFC = REAL ( -1 )
END IF

IF ( CAPE /= CAPE ) cape = REAL ( 0 )

IF ( CIN /= CIN ) cin = REAL ( 0 )

```

E.2 lmdz

Not simple to get the right code, computation is split in different subroutines being the main driving one `phylmd/cva_driver.F90`

F cin

F.1 afwa

Similar as in with CAPE ([E.1](#))

F.2 lmdz

G WRF CORDEX variables definition

Here is provided the table (1) of equivalencies among CORDEX CF-names and their definitions in WRF (directly from Registry/registry.cordex)

Here is provided the table (3) with the definitions of the water-budget additional variables in WRF (directly from Registry/registry.cordex)

Table 1: Equivalencies among CORDEX CF variables' name and how are they defined in WRF module (from `registry.cordex`)

CF name	WRF name	description	units
2D			
lon	LON	LONGITUDE	degrees_east
lat	LAT	LATITUDE	degrees_north
capev	CDXCAPE	CONVECTIVE AVAILABLE POTENTIAL ENERGY	Jkg-1
cin	CIN	CONVECTIVE INHIBITION	Jkg-1
cltmean	CLTMEAN	MEAN TOTAL CLOUDINESS IN CORDEX OUTPUT	1
cllmean	CLLMEAN	MEAN LOW-LEVEL CLOUDINESS (p >= 68000 Pa) IN CORDEX OUTPUT	1
clmmean	CLMMEAN	MEAN MID-LEVEL CLOUDINESS (44000 <= p < 68000 Pa) IN CORDEX OUTPUT	1
clhmean	CLHMEAN	MEAN HIGH-LEVEL CLOUDINESS (p < 44000 Pa) IN CORDEX OUTPUT	1
mrso	MRSO	TOTAL SOIL CONTENT	kgm-2
prw	PRW	WATER VAPOR PATH	kgm-2
psl	PSL	SEA LEVEL PRESSURE	Pa
clwvi	CLWVI	LIQUID WATER PATH	kgm-2
clivi	CLIVI	ICE WATER PATH	kgm-2
clgvi	CLGVI	GRAUPEL WATER PATH	kgm-2
clhvi	CLHVI	HAIL WATER PATH	kgm-2
hurs	HURS	2M RELATIVE HUMIDITY	1
huss	HUSS	2M SPECIFIC HUMIDITY	1
lfcv	LFCV	PRESSURE LEVEL FREE CONVECTION	Pa
lfcz	LFCZ	HEIGHT LEVEL FREE CONVECTION	m
li	LI	LIFTED INDEX	1
slw	SLW	TOTAL SOIL LIQUID WATER CONTENT	kgm-2
uas	UAS	10M EASTWARD WIND SPEED	ms-1
vas	VAS	10M NORTHWARD WIND SPEED	ms-1
wsgsmax	WSGSMAX	Maximum near-surface wind speed of gust	ms-1
usgsmax	USGSMAX	Eastward maximum near-surface wind speed of gust	ms-1
vsgsmax	VSGSMAX	Northward maximum near-surface wind speed of gust	ms-1
wsgspercen	WSGSPERCEN	Percentage of time steps where grid point got wind gust	%
totwsgsmax	TOTWSGSMAX	Total (TKE + h. pr) Maximum near-surface wind speed of gust	ms-1
totugsmax	TOTUGSMAX	Total Eastward maximum near-surface wind speed of gust	ms-1
totvgmax	TOTVGSMAX	Total Northward maximum near-surface wind speed of gust	ms-1
totwsgspercen	TOTWSGSPERCEN	Percentage of time steps where grid point got total wind gust	%
zmlagen	ZMLAGEN	Generic boundary layer height $\theta(zmlagen) > \min(\theta[mix. layer]) + 1.5K$	m
wsz100max	WSZ100MAX	Maximum 100m nwind speed	ms-1

Table 2: Contionuation of table 1

uz100max	UZ100MAX	Eastward maximum 100 m wind speed	ms-1
vz100max	VZ100MAX	Northward maximum 100 m wind speed	ms-1
sund	SUND	SUNSHINE LENGTH (ac. time SWDOWN > 120. Wm-2)	second
tauu	TAUU	northward downward wind stress at 10 m	m2s-2
tauv	TAUV	eastward downward wind stress at 10 m	m2s-2
3D			
HUR	HUR	AIR RELATIVE HUMIDITY	1
HUS	HUS	AIR SPECIFIC HUMIDITY	1
ZG	ZG	AIR GEOPOTENTIAL HEIGHT	m
PRESS	PRESS	AIR PRSSURE	Pa
TA	TA	AIR TEMPERATURE	K
UA	UA	AIR EASTWARD WIND SPEED	ms-1
VA	VA	AIR NORTHWARD WIND SPEED	ms-1

Table 3: Water-budget equivalencies and their name and how are they defined in WRF module (from `registry.cordex`)

name	WRF name	description	units
Q_{hac}	WBACDIABH	Water Budget column integrated and time accumulation of diabatic heating from Micro-Physics	K
$\partial_t qvac$	WBACPW	Water Budget column integrated and time accumulated for water vapor content	mm
$\partial_t qcac$	WBACPWC	Water Budget col. int. & time accumulated for cloud content	mm
$\partial_t qrac$	WBACPWR	Water Budget col. int. & time accumulated for rain content	mm
$\partial_t qsac$	WBACPWS	Water Budget col. int. & time accumulated for snow content	mm
$\partial_t qiacc$	WBACPWI	Water Budget col. int. & time accumulated for ice content	mm
$\partial_t qhac$	WBACPWH	Water Budget col. int. & time accumulated for hail content	mm
$\partial_t qgac$	WBACPWG	Water Budget col. int. & time accumulated for graupel content	mm
$adv_h qvac$	WBACF	W.B. c-int. acc. hor. convergence of water vapour (+, conv.; -, div.)	mm
$adv_h qcac$	WBACFC	W.B. c-int. acc. hor. convergence of cloud (+, conv.; -, div.)	mm
$adv_h qrac$	WBACFR	W.B. c-int. acc. hor. convergence of rain (+, conv.; -, div.)	mm
$adv_h qsac$	WBACFS	W.B. c-int. acc. hor. convergence of snow (+, conv.; -, div.)	mm
$adv_h qiacc$	WBACFI	W.B. c-int. acc. hor. convergence of ice (+, conv.; -, div.)	mm
$adv_h qhac$	WBACFH	W.B. c-int. acc. hor. convergence of hail (+, conv.; -, div.)	mm
$adv_h qgac$	WBACFG	W.B. c-int. acc. hor. convergence of graupel (+, conv.; -, div.)	mm
$adv_z qvac$	WBACZ	W.B. c-int. acc. ver. convergence of water vapour (+, conv.; -, div.), always 0	mm
$adv_z qcac$	WBACZC	W.B. c-int. acc. ver. convergence of cloud (+, conv.; -, div.)	mm
$adv_z qrac$	WBACZR	W.B. c-int. acc. ver. convergence of rain (+, conv.; -, div.)	mm
$adv_z qsac$	WBACZS	W.B. c-int. acc. ver. convergence of snow (+, conv.; -, div.)	mm
$adv_z qiacc$	WBACZI	W.B. c-int. acc. ver. convergence of ice (+, conv.; -, div.)	mm
$adv_z qhac$	WBACZH	W.B. c-int. acc. ver. convergence of hail (+, conv.; -, div.)	mm
$adv_z qgac$	WBACZG	W.B. c-int. acc. ver. convergence of graupel (+, conv.; -, div.)	mm
Low-mid-level			
Q_{hac}^l	WBACDIABHL	W.B. low level acc. of diabatic heating from MP	K
Q_{hac}^m	WBACDIABHM	W.B. mid-level acc. of diabatic heating from MP	K
Q_{hac}^h	WBACDIABHH	W.B. high-level acc. of diabatic heating from MP	K
$\partial_t^l qvac$	WBACPWL	W.B. low level ($p \geq 68000$ Pa) acc. for QV	mm
$\partial_t^m qvac$	WBACPWM	W.B. mid level ($44000 \text{ Pa} \leq p < 68000 \text{ Pa}$) acc. for QV	mm
$\partial_t^h qvac$	WBACPWHV	W.B. high level ($p < 44000$ Pa) acc. for QV	mm
$adv_h^l qvac$	WBACFLV	W.B. low-lev. acc. hor. convergence of QV	mm
$adv_h^m qvac$	WBACFMV	W.B. mid-lev. acc. hor. convergence of QV	mm
$adv_h^h qvac$	WBACFHV	W.B. high-lev. acc. hor. convergence of QV	mm
$adv_z^l qvac$	WBACZLV	W.B. low level acc. ver. convergence of QV	mm

Table 4: Continuation of table 3

$adv_z^m qvac$	WBACZMV	W.B. mid level acc. ver. convergence of QV	mm
$adv_z^h qvac$	WBACZHV	W.B. high level acc. ver. convergence of QV	mm
$\partial_t^l qcac$	WBACPWLC	W.B. low level ($p \geq 68000$ Pa) acc. for QC	mm
$\partial_t^m qcac$	WBACPWMC	W.B. mid level ($44000 \text{ Pa} \leq p < 68000 \text{ Pa}$) acc. for QC	mm
$\partial_t^h qcac$	WBACPWHC	W.B. high level ($p < 44000$ Pa) acc. for QC	mm
$adv_h^l qcac$	WBACFLC	W.B. low-lev. acc. hor. convergence of QC	mm
$adv_h^m qcac$	WBACFMC	W.B. mid-lev. acc. hor. convergence of QC	mm
$adv_h^h qcac$	WBACFHC	W.B. high-lev. acc. hor. convergence of QC	mm
$adv_z^l qcac$	WBACZLC	W.B. low level acc. ver. convergence of QC	mm
$adv_z^m qcac$	WBACZMC	W.B. mid level acc. ver. convergence of QC	mm
$adv_z^h qcac$	WBACZHC	W.B. high level acc. ver. convergence of QC	mm
$\partial_t^l qrac$	WBACPWLR	W.B. low level ($p \geq 68000$ Pa) acc. for QR	mm
$\partial_t^m qrac$	WBACPWMR	W.B. mid level ($44000 \text{ Pa} \leq p < 68000 \text{ Pa}$) acc. for QR	mm
$\partial_t^h qrac$	WBACPWHR	W.B. high level ($p < 44000$ Pa) acc. for QR	mm
$adv_h^l qrac$	WBACFLR	W.B. low-lev. acc. hor. convergence of QR	mm
$adv_h^m qrac$	WBACFMR	W.B. mid-lev. acc. hor. convergence of QR	mm
$adv_h^h qrac$	WBACFHR	W.B. high-lev. acc. hor. convergence of QR	mm
$adv_z^l qrac$	WBACZLR	W.B. low level acc. ver. convergence of QR	mm
$adv_z^m qrac$	WBACZMR	W.B. mid level acc. ver. convergence of QR	mm
$adv_z^h qrac$	WBACZHR	W.B. high level acc. ver. convergence of QR	mm
$\partial_t^l qsac$	WBACPWLS	W.B. low level ($p \geq 68000$ Pa) acc. for QS	mm
$\partial_t^m qsac$	WBACPWMS	W.B. mid level ($44000 \text{ Pa} \leq p < 68000 \text{ Pa}$) acc. for QS	mm
$\partial_t^h qsac$	WBACPWHS	W.B. high level ($p < 44000$ Pa) acc. for QS	mm
$adv_h^l qsac$	WBACFLS	W.B. low-lev. acc. hor. convergence of QS	mm
$adv_h^m qsac$	WBACFMS	W.B. mid-lev. acc. hor. convergence of QS	mm
$adv_h^h qsac$	WBACFHS	W.B. high-lev. acc. hor. convergence of QS	mm
$adv_z^l qsac$	WBACZLS	W.B. low level acc. ver. convergence of QS	mm
$adv_z^m qsac$	WBACZMS	W.B. mid level acc. ver. convergence of QS	mm
$adv_z^h qsac$	WBACZHS	W.B. high level acc. ver. convergence of QS	mm
$\partial_t^l qiac$	WBACPWLI	W.B. low level ($p \geq 68000$ Pa) acc. for QI	mm
$\partial_t^m qiac$	WBACPWMI	W.B. mid level ($44000 \text{ Pa} \leq p < 68000 \text{ Pa}$) acc. for QI	mm
$\partial_t^h qiac$	WBACPWHI	W.B. high level ($p < 44000$ Pa) acc. for QI	mm
$adv_h^l qiac$	WBACFLI	W.B. low-lev. acc. hor. convergence of QI	mm
$adv_h^m qiac$	WBACFMI	W.B. mid-lev. acc. hor. convergence of QI	mm
$adv_h^h qiac$	WBACFHI	W.B. high-lev. acc. hor. convergence of QI	mm
$adv_z^l qiac$	WBACZLI	W.B. low level acc. ver. convergence of QI	mm
$adv_z^m qiac$	WBACZMI	W.B. mid level acc. ver. convergence of QI	mm
$adv_z^h qiac$	WBACZHI	W.B. high level acc. ver. convergence of QI	mm
$\partial_t^l qgac$	WBACPWLG	W.B. low level ($p \geq 68000$ Pa) acc. for QG	mm
$\partial_t^m qgac$	WBACPWMG	W.B. mid level ($44000 \text{ Pa} \leq p < 68000 \text{ Pa}$) acc. for QG	mm
$\partial_t^h qgac$	WBACPWHG	W.B. high level ($p < 44000$ Pa) acc. for QG	mm
$adv_h^l qgac$	WBACFLG	W.B. low-lev. acc. hor. convergence of QG	mm
$adv_h^m qgac$	WBACFMG	W.B. mid-lev. acc. hor. convergence of QG	mm
$adv_h^h qgac$	WBACFHG	W.B. high-lev. acc. hor. convergence of QG	mm
$adv_z^l qgac$	WBACZLG	W.B. low level acc. ver. convergence of QG	mm
$adv_z^m qgac$	WBACZMG	W.B. mid level acc. ver. convergence of QG	mm
$adv_z^h qgac$	WBACZHG	W.B. high level acc. ver. convergence of QG	mm
$\partial_t^l qhac$	WBACPWLH	W.B. low level ($p \geq 68000$ Pa) acc. for QH	mm
$\partial_t^m qhac$	WBACPWMH	W.B. mid level ($44000 \text{ Pa} \leq p < 68000 \text{ Pa}$) acc. for QH	mm
$\partial_t^h qhac$	WBACPWHH	W.B. high level ($p < 44000$ Pa) acc. for QH	mm
$adv_h^l qhac$	WBACFLH	W.B. low-lev. acc. hor. convergence of QH	mm
$adv_h^m qhac$	WBACFMH	W.B. mid-lev. acc. hor. convergence of QH	mm
$adv_h^h qhac$	WBACFHH	W.B. high-lev. acc. hor. convergence of QH	mm
$adv_z^l qhac$	WBACZLH	W.B. low level acc. ver. convergence of QH	mm
$adv_z^m qhac$	WBACZMH	W.B. mid level acc. ver. convergence of QH	mm
$adv_z^h qhac$	WBACZHH	W.B. high level acc. ver. convergence of QH	mm