LMDZ

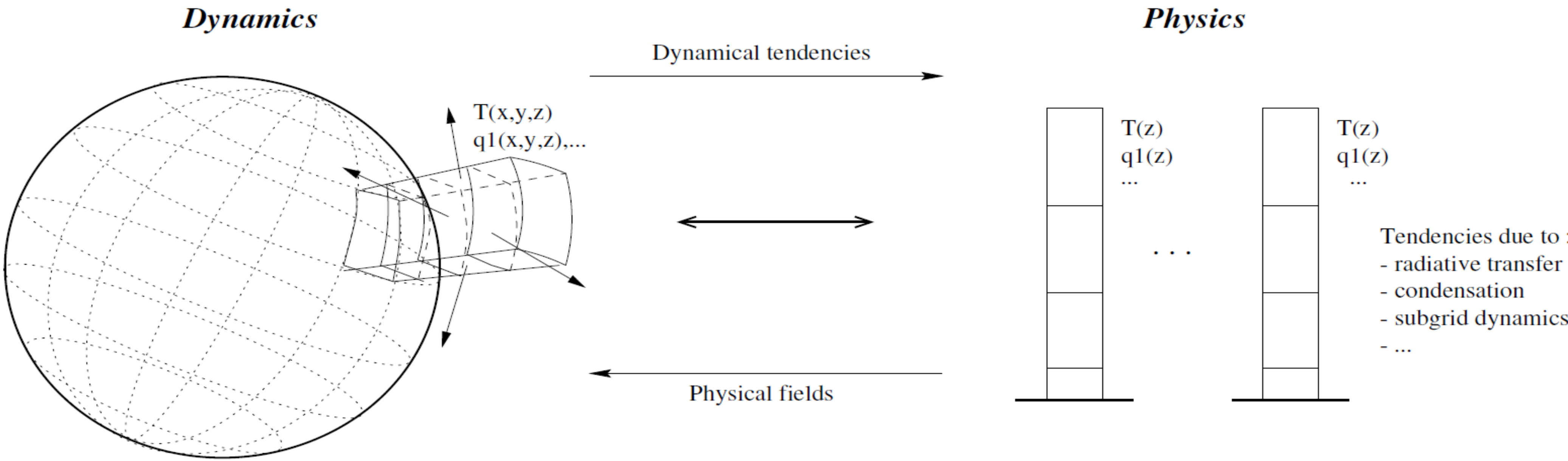
Dynamics/physics organization,
Grids,
Time stepping,
Dissipation...

LMDZ courses, January 14, 2026

Overview of course topics

- Grids:
  - Horizontal grids in the physics & dynamics
  - Vertical discretization
- Time marching:
  - Generalities about time marching schemes
  - What is used in LMDZ
  - Longitudinal polar filter
- Lateral diffusion and sponge layer:
  - Energy cascade
  - Illustrative example of diffusion
  - Sponge layer near model top

Grids in LMDZ



Separation between physics and dynamics:

- "dynamics": solving the GFD equations on the sphere; usually with the assumption of a hydrostatic balance and thin layer approximation. Valid for all terrestrial planets.
- "physics": (planet-specific) local processes, local to individual atmospheric columns.

Horizontal grids in LMDZ

Grid dimensions specified when compiling LMDZ:

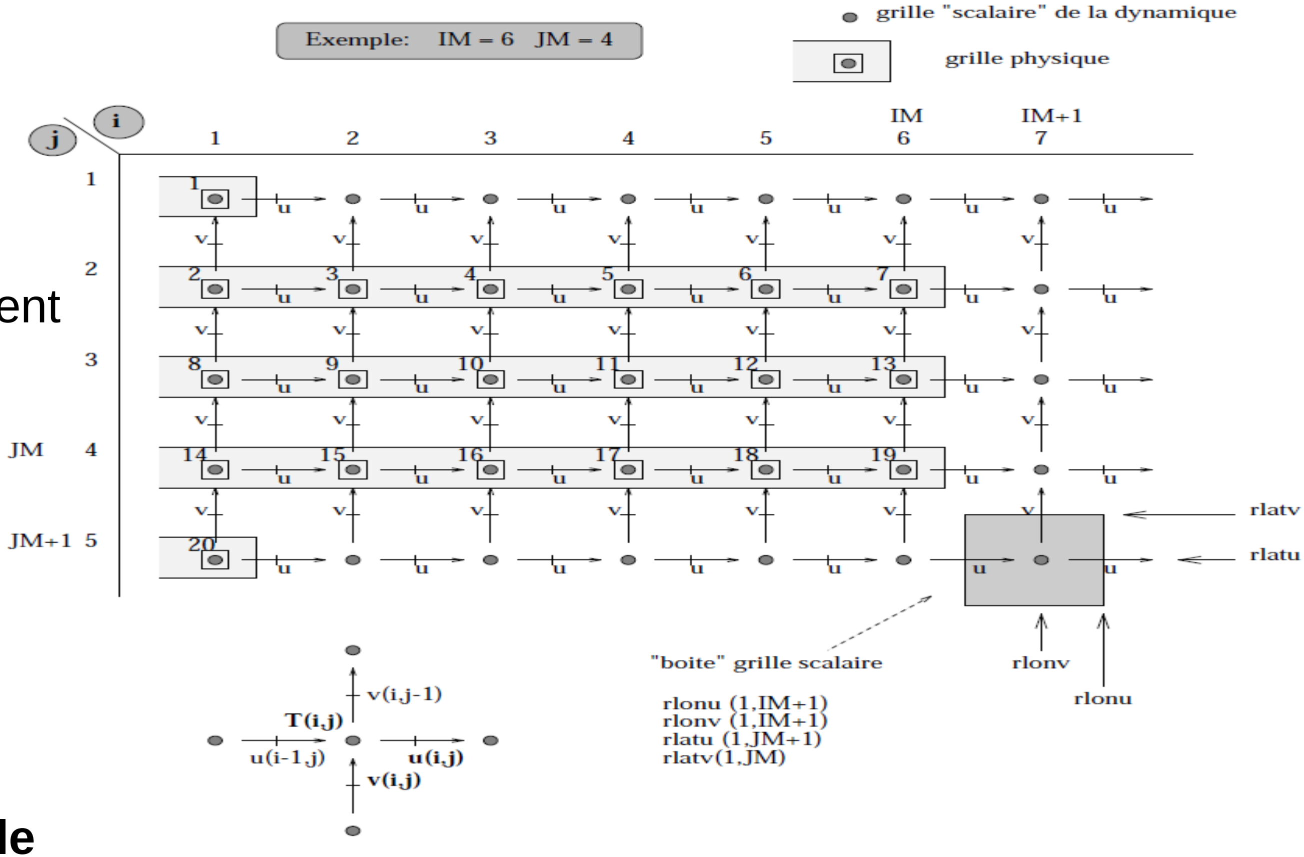
makelmdz_fcm -d iimxjjmxllm ...

In the dynamics:

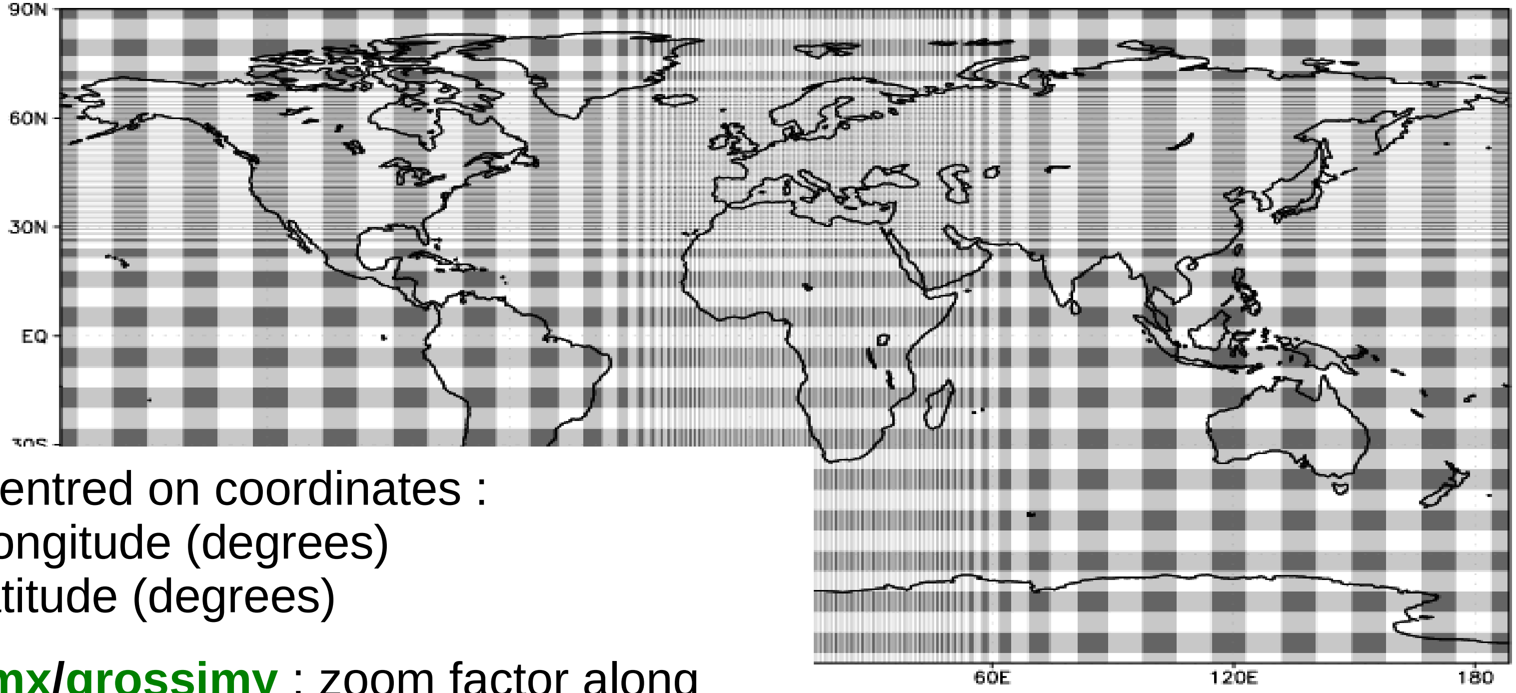
- Staggered grids: u, v and scalars (temperature, tracers) are on different meshes
- Global lonxlat grids with redundant grid points
  - at the poles
  - in longitude

In the physics:

- Collocated variables
- No global lonxlat horizontal grid, columns are labelled **using a single index** (from North Pole to South Pole)



LMDZ, Z for Zoom

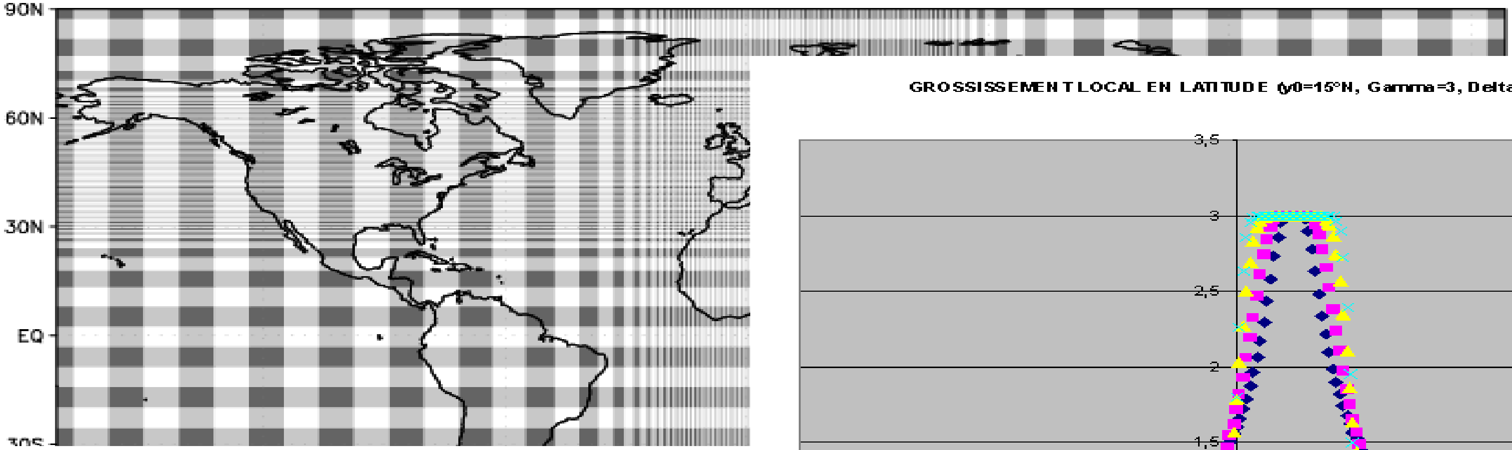


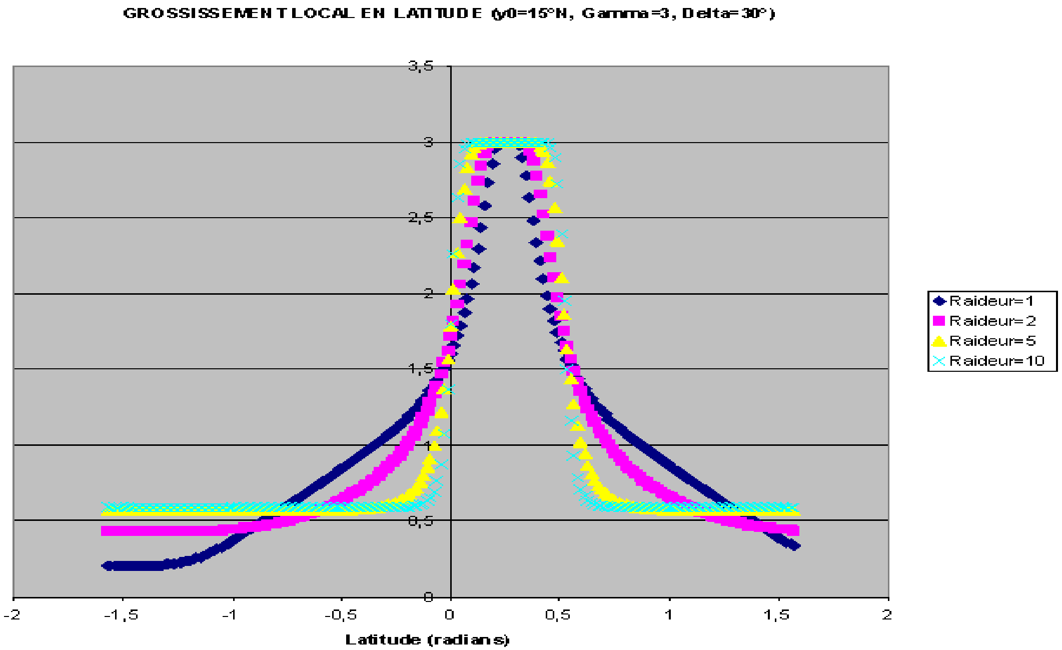
Zoom centred on coordinates :
clon : longitude (degrees)
clat : latitude (degrees)

grossimx/grossimy : zoom factor along x/y directions (i.e. lon/lat)
Computed as the ratio of the smallest mesh (I.e. in the zoom), compared to the mesh size for a global regular grid with the same total number of points.

dzoomx/dzoomy : fraction of the grid containing the zoomed area: dzoom*360° by dzoomy*180°

LMDZ, Z for Zoom





Zoom centred on coordinates :
clon : longitude (degrees)
clat : latitude (degrees)

grossimx/**grossimy** : zoom factor along x/y directions (i.e. lon/lat)
Computed as the ratio of the smallest mesh (I.e. in the zoom), compared to the mesh size for a global regular grid with the same total number of points.

taux/**tauy** : steepness of the transition between inner zoom and outer zoom meshes (typically one tries to avoid sharp transitions; tau ~ 3 is a reasonable value)

Nudging in LMDZ

$$\frac{\partial u}{\partial t} = \left.\frac{\partial u}{\partial t}\right|_{GCM} + \frac{u_{analyse} - u}{\tau}$$

$$\frac{\partial v}{\partial t} = \left.\frac{\partial v}{\partial t}\right|_{GCM} + \frac{v_{analyse} - v}{\tau}$$

Nudging towards analyses or reanalyses with chosen time constants

(nudge = “guide” in French)

Example of nudging parameters:

ok_guide = y

guide_T = n , guide_p = n , guide_q = n

guide_u = y , guide_v = y

tau_min_u = 0.0208333 (days)

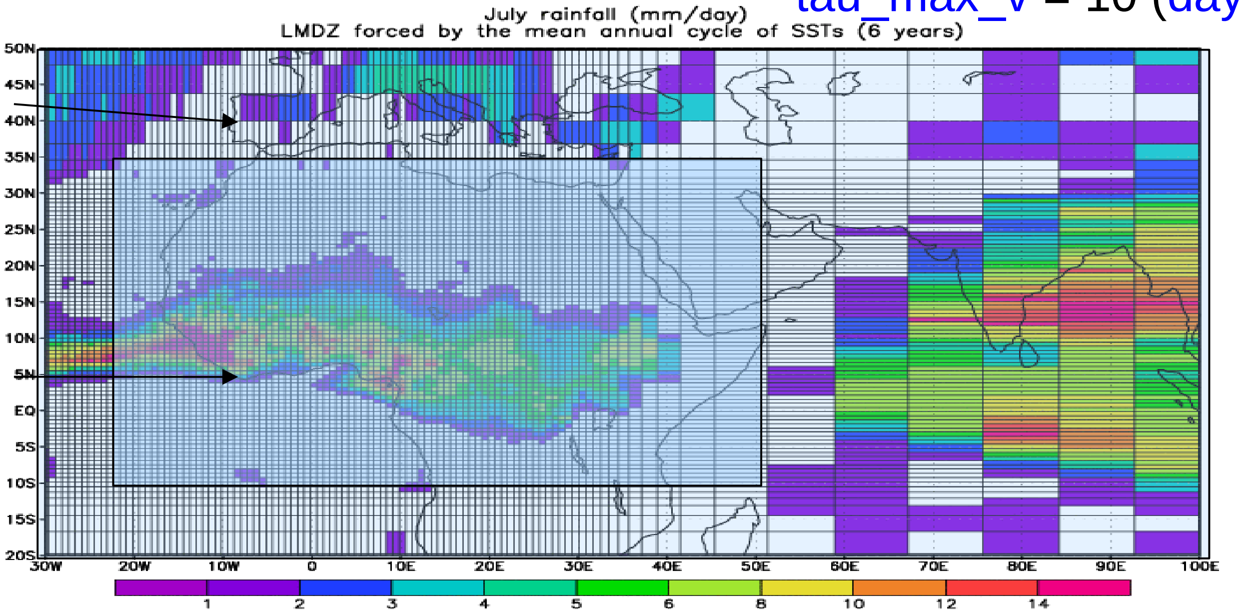
tau_max_u = 10 (days)

tau_min_v = 0.0208333 (days)

tau_max_v = 10 (days)

Strong nudging (τ=30min)

Weak to moderate nudging (τ=10 days)



Vertical discretization in LMDZ

- Model levels are hybrid sigma-pressure levels:

  P(level,time) = **ap**(level) + **bp**(level) . **Ps** (time)

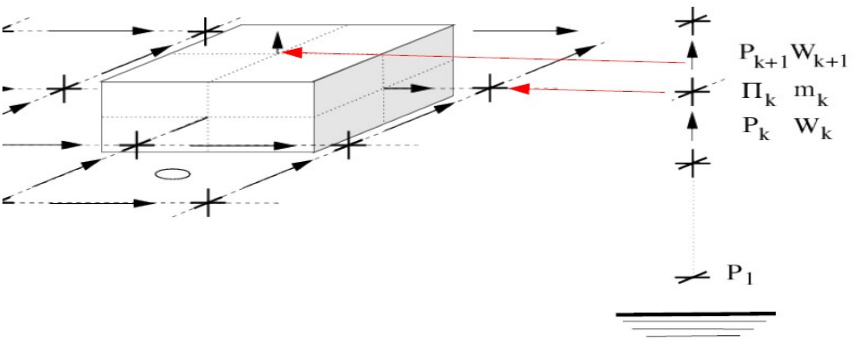
  hybrid coordinates **ap**(k) and **bp**(k) are fixed for a given model run

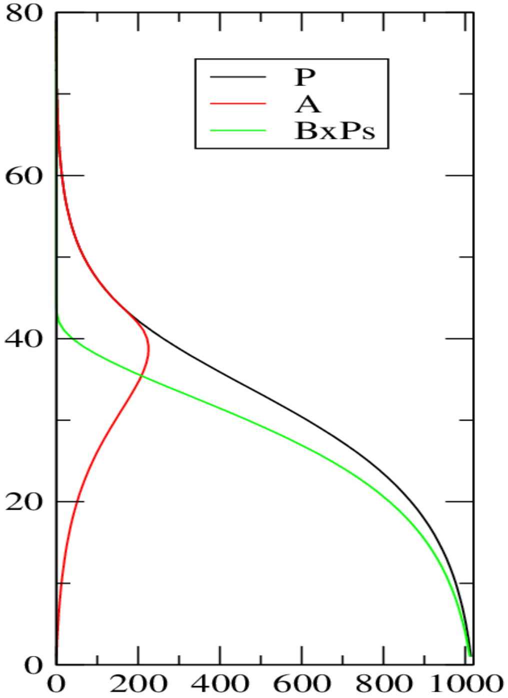
  Surface pressure **Ps**(t) varies during the run

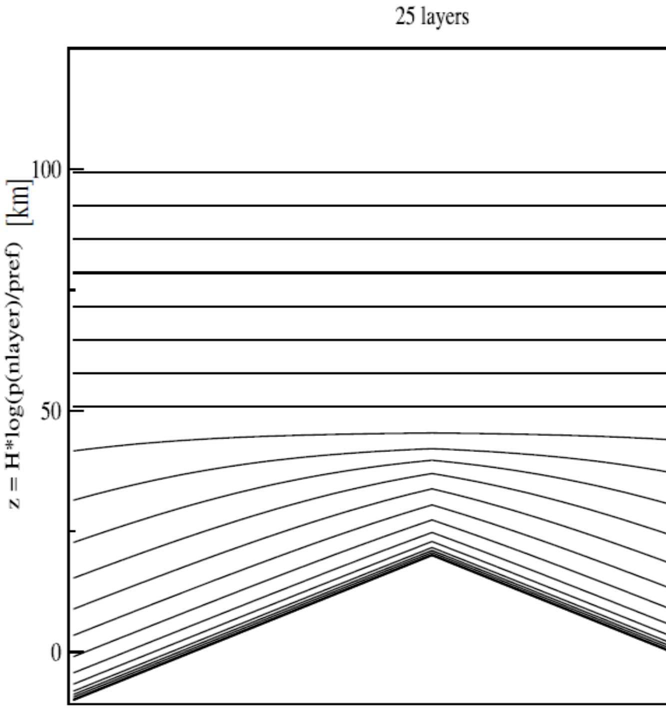
- Near the surface ap ~ 0

  => bp(k) ~ P/Ps

- At high altitudes , bp ~ 0







Vertical discretization in LMDZ

- Setting model levels via the def files (also a function of number of vertical levels) :

  vert_sampling = strato_custom : customable (via other parameters in .def file; see next slide) discretization for stratospheric extensions.

  Multiple other possibilities from this “default”:

  vert_sampling = strato : a default for stratospheric extensions

  vert_sampling = sigma : automated generation of purely sigma levels

  vert_sampling = param : load values from a “sigma.def” file

  vert_sampling = tropo : a default for tropospheric simulations

  vert_sampling = read : read ap() and bp() from file “hybrid.txt”

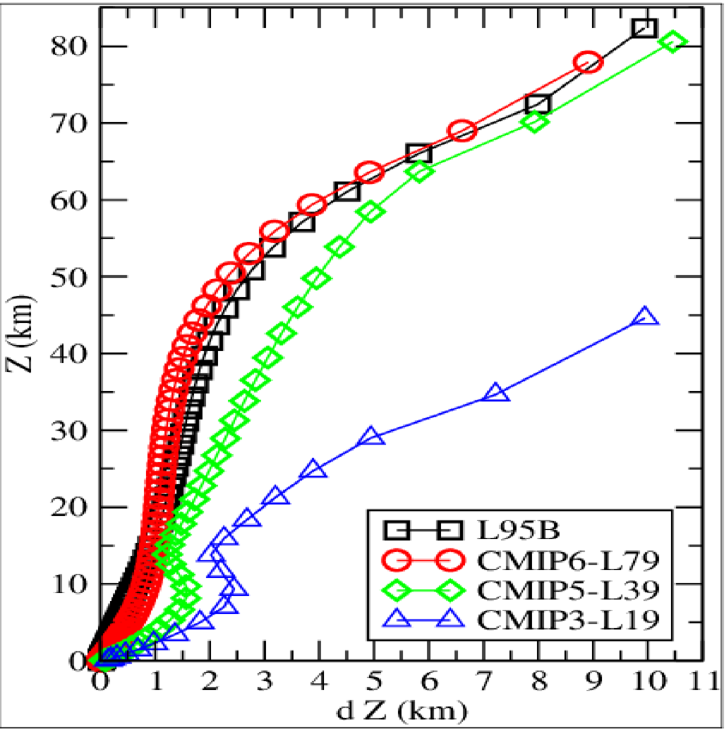
  => Typically you don't need to mess with the vertical discretization,

  the default behaviour most likely matches your needs.

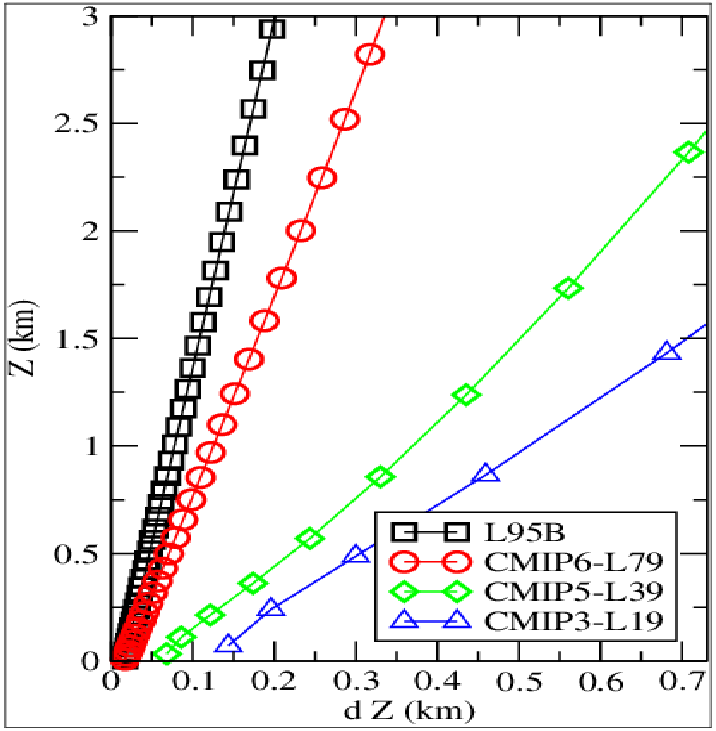
  => Check out routine dyn3d_common/disvert.F90

Vertical discretization in LMDZ

Illustration of typical altitudes and layer thickness of L19/L39/L79/L95 grids



Over the whole atmosphere
0 < z < ~80 km



Near the surface
0 < z < 3 km

“Standard” CMIP6 L79 settings (see DefLists/vert_L79.def)

vert_sampling=strato_custom

vert_scale=7.
vert_dzmin=0.017
vert_dzlow=1.
vert_z0low=8.7
vert_dzmid=2.
vert_z0mid=70.
vert_h_mid=20.
vert_dzhig=11.
vert_z0hig=75.
vert_h_hig=20.

Questions ?

Time marching schemes

- **The big picture**: you want to solve

$$\frac{df(t)}{dt} = R(f,t)$$
$$f(t=0) = f_0$$

from a known initial condition at time t=0 to time t=...

- So it is all about using a time marching scheme, built on Taylor expansion for evaluation of the time derivative, and choosing at which time level t=n.δt the right hand side term R[f(t),t] is to be evaluated :

$$f(t_0+\delta t) = f(t_0) + \frac{\delta t}{1!} f'(t_0) + \frac{(\delta t)^2}{2!} f''(t_0) + \ldots$$

Time marching schemes

- Explicit Euler scheme (1st order in time):

$$\begin{aligned}\frac{df(t)}{dt} &\simeq \frac{f_{n+1}-f_n}{\delta t}\\ R(f,t) &\simeq R(f(t_n),t_n)\end{aligned}$$

- Implicit Euler scheme (1st order in time):

$$\begin{aligned}\frac{df(t)}{dt} &\simeq \frac{f_{n+1}-f_n}{\delta t}\\ R(f,t) &\simeq R(f(t_{n+1}),t_{n+1})\end{aligned}$$

- Crank-Nicholson scheme (2nd order in time):

$$\begin{aligned}\frac{df(t)}{dt} &\simeq \frac{f_{n+1}-f_n}{\delta t}\\ R(f,t) &\simeq \frac{R(f(t_{n+1}),t_{n+1})+R(f(t_n),t_n)}{2}\end{aligned}$$

Time marching schemes

- Matsuno scheme: a predictor-corrector (Euler explicit-Euler Implicit) scheme (1st order):

$$
\begin{aligned}
\frac{df(t)}{dt} &\simeq \frac{f_{n+1} - f_n}{\delta t} \\
f^p(t_{n+1}) &= f(t_n) + \delta t . R(f(t_n), t_n) \\
R(f, t) &\simeq R(f^p(t_{n+1}), t_{n+1})
\end{aligned}
$$

- Leapfrog scheme: use encompassing time steps to evaluate the derivative (2nd order):

$$
\begin{aligned}
\frac{df(t)}{dt} &\simeq \frac{f_{n+1} - f_{n-1}}{2\delta t} \\
R(f, t) &\simeq R(f(t_n), t_n)
\end{aligned}
$$

Time marching schemes

- Illustrative example, on a decay equation (known solution!)

$$\frac{dq(t)}{dt} = -\frac{1}{\tau}q(t) \longrightarrow q(t) = q_0 e^{-\frac{t}{\tau}}$$

- Building Euler explicit (E) & implicit (I) schemes:

$$\begin{aligned} \frac{dq(t)}{dt} &= -\frac{1}{\tau}q(t) \\ \Rightarrow \frac{q^{n+1}-q^n}{\delta t} &\simeq -\frac{1}{\tau}q^n \qquad \text{(E.E.)} \\ \Leftrightarrow q^{n+1}-q^n &= -\frac{\delta t}{\tau}q^n \\ \Leftrightarrow q^{n+1} &= \left[1-\frac{\delta t}{\tau}\right]q^n \end{aligned}$$

$$\begin{aligned} \frac{dq(t)}{dt} &= -\frac{1}{\tau}q(t) \\ \Rightarrow \frac{q^{n+1}-q^n}{\delta t} &\simeq -\frac{1}{\tau}q^{n+1} \qquad \text{(E.I.)} \\ \Leftrightarrow q^{n+1}-q^n &= -\frac{\delta t}{\tau}q^{n+1} \\ \Leftrightarrow \frac{\tau+\delta t}{\tau}q^{n+1} &= q^n \\ \Leftrightarrow q^{n+1} &= \frac{1}{1+\frac{\delta t}{\tau}}q^n \end{aligned}$$

Time marching schemes

- Illustrative example, on a decay equation

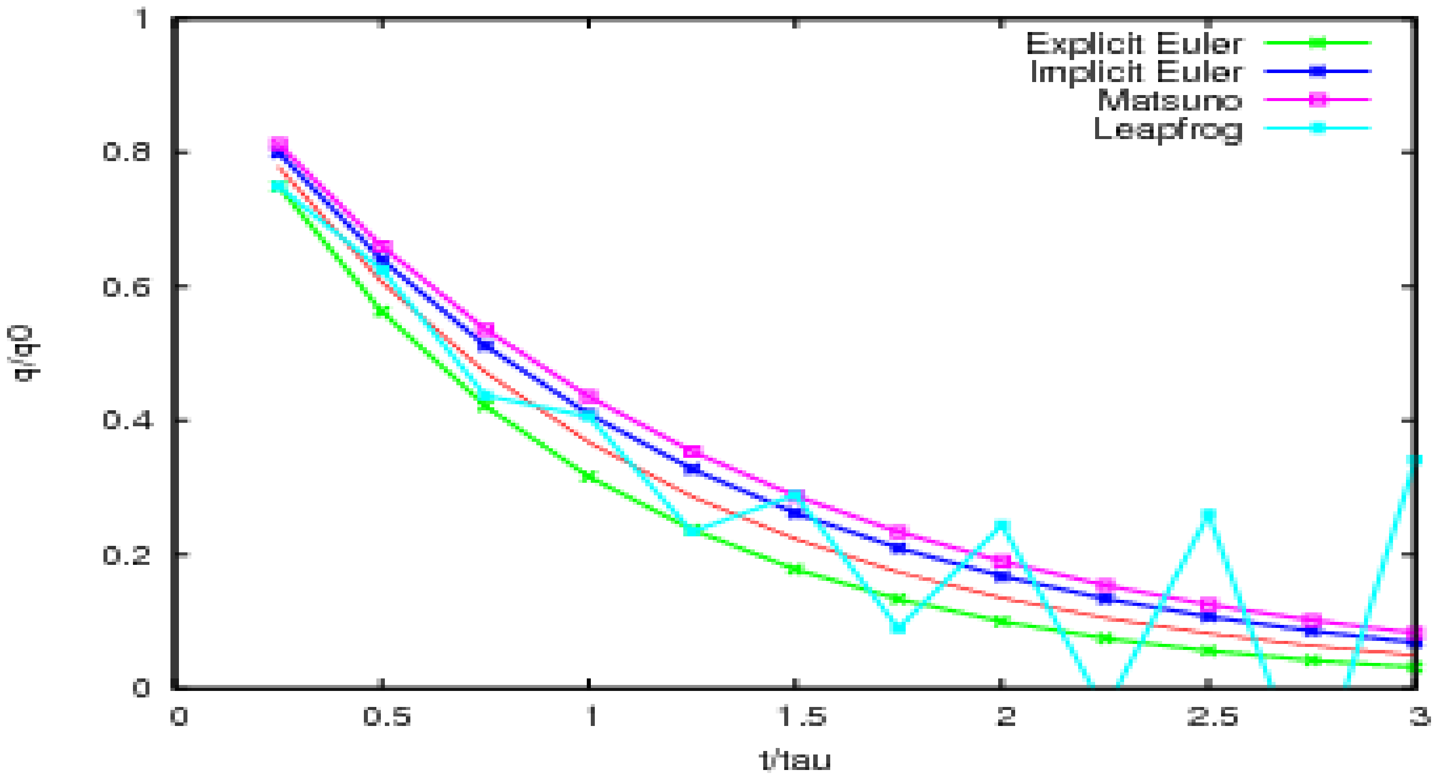
$$\frac{dq(t)}{dt} = -\frac{1}{\tau} q(t) \longrightarrow q(t) = q_0 e^{-\frac{t}{\tau}}$$

- Resulting integration schemes:

$$\begin{aligned} \mathrm{EE}: \quad q^{n+1} &= \left[1 - \frac{\delta t}{\tau}\right] q^n \\ \mathrm{EI}: \quad q^{n+1} &= \left[\frac{1}{1 + \delta t/\tau}\right] q^n \\ \mathrm{CN}: \quad q^{n+1} &= \left[\frac{1 - \delta t/(2\tau)}{1 + \delta t/(2\tau)}\right] q^n \end{aligned}$$

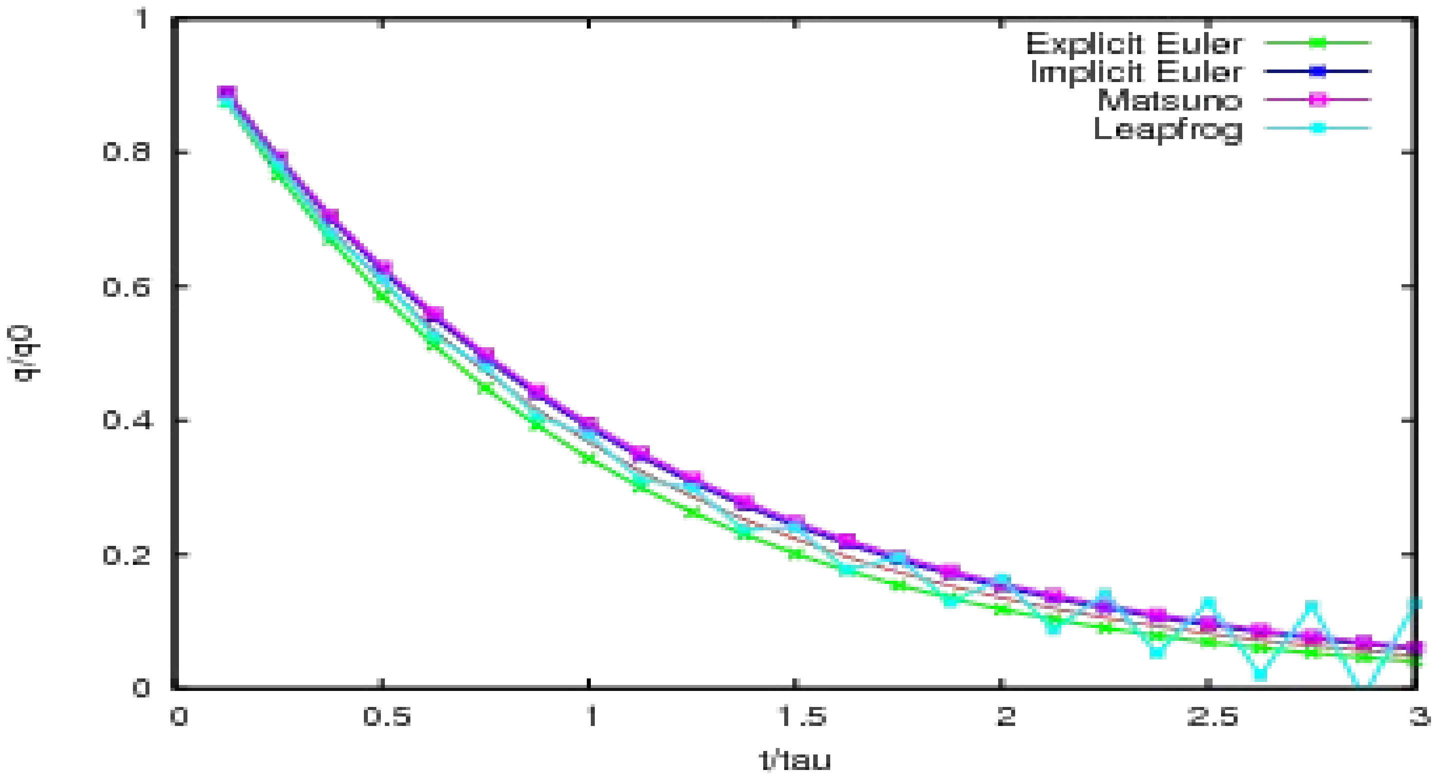
- Stability requirement (CFL) for EE : dt/tau < 2

Time marching schemes



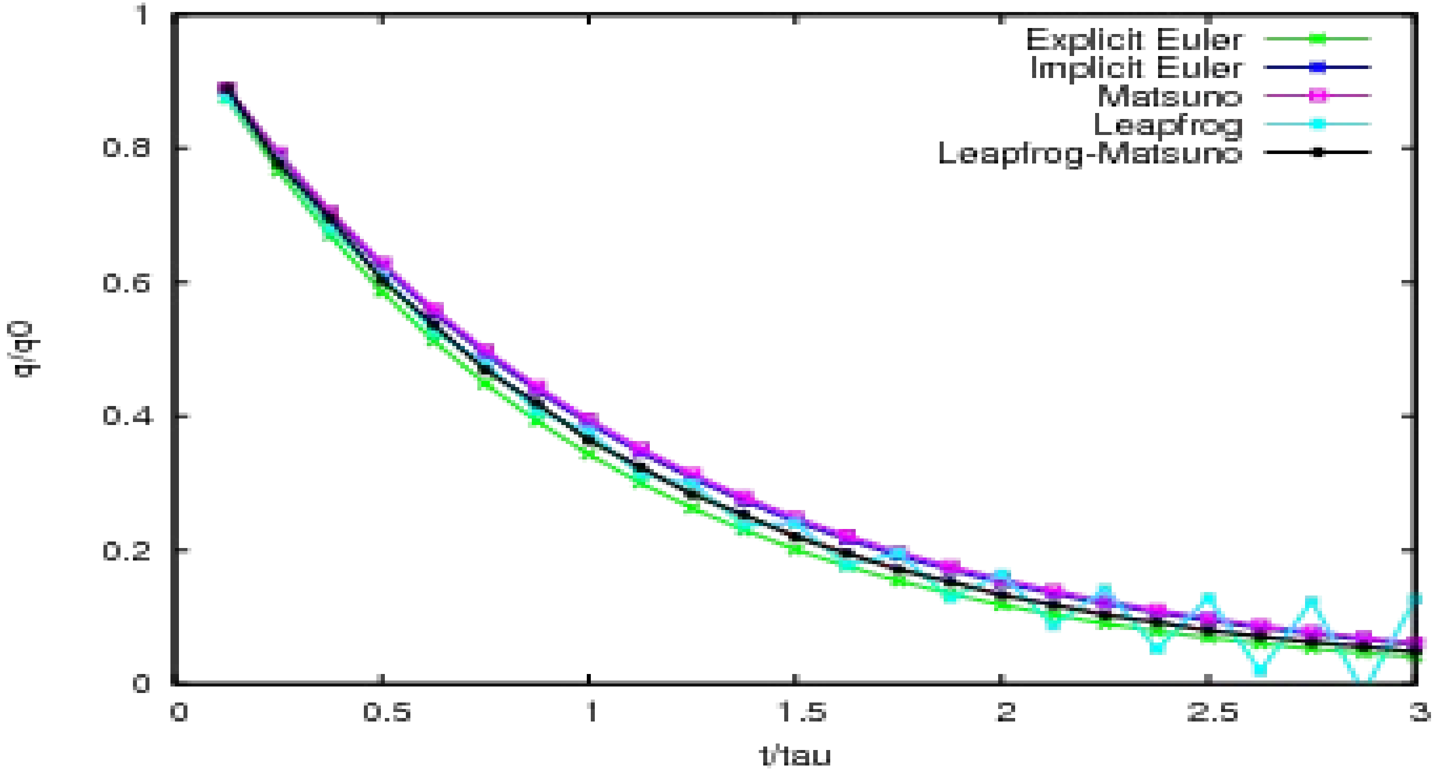
- 4 integration steps per unit of t/tau

Time marching schemes



- 8 integration steps per unit of t/tau

Time marching schemes



- 8 integration steps per unit of t/tau

Time marching in LMDZ

Time splitting between physics/dynamics/dissipation:

$$\frac{\partial \psi}{\partial t} = Dyn(\psi) + Phy(\psi) + Dissip(\psi)$$

- Dynamics : Leapfrog-Matsuno scheme

  Using day_step dynamical steps per day

  Leapfrog steps with a Mastuno step every iperiod step
- Physics : Explicit Euler

  Every iphysiq dynamical steps (multiple of iperiod)
- Dissipation: Explicit Euler

  Every dissip_period dynamical steps (multiple of iperiod)

Side note about explicit or implicit time marching schemes

Even when solving linear spatio-temporal boundary-value problems, e.g.: $$\frac{dA}{dt} = \kappa \frac{\partial^2 A}{\partial x^2}$$

The explicit Euler approach leads to a straightforward expression for grid point values (but with stability constraints) :

$$\frac{A_i^{k+1} - A_i^k}{\delta t} = \frac{\kappa}{h^2}\left[A_{i-1}^k - 2A_i^k + A_{i+1}^k\right]$$

Whereas the implicit Euler approach leads to a (tridiagonal) system of equations to solve:

$$\frac{A_i^{k+1} - A_i^k}{\delta t} = \frac{\kappa}{h^2}\left[A_{i-1}^{k+1} - 2A_i^{k+1} + A_{i+1}^{k+1}\right]$$

=> requires more computations, but may be necessary if time-stepping constrains require using large time steps.

Side note about tridiagonal system solving

When needing to solve a tridiagonal system of the form:

$T.x=y$, T tridiagonal matrix, x & y vectors

Rather than invert T (costly!) to generate T^{-1} (dense matrix) and then compute $x=T^{-1}.y$ (matrix-vector product)

Use the LU decomposition (Gaussian elimination) of T to split the problem into two very simple sub-problems:

1) $L.U=T$, L and U are bidiagonal (lower/upper) matrices

2) Solve $L.z=y$ for vector z (forward substitution step)

3) Solve $U.x=z$ for vector x (backward substitution step)

Side note about incorporating boundary conditions (BC)

$$\frac{\partial f(x,t)}{\partial t} = R(f,x,t) \text{ with BC: } f(x_B,t) = B(x_B,t)$$

- In practice one needs to **treat specifically** the spatial boundary conditions, which could be either imposed on the value (Dirichlet) or its derivative (Neumann) or a mix of the two (Robin) or ...
- The choice of using an implicit or explicit representation also impacts the incorporation of the boundary conditions (along spatial directions)

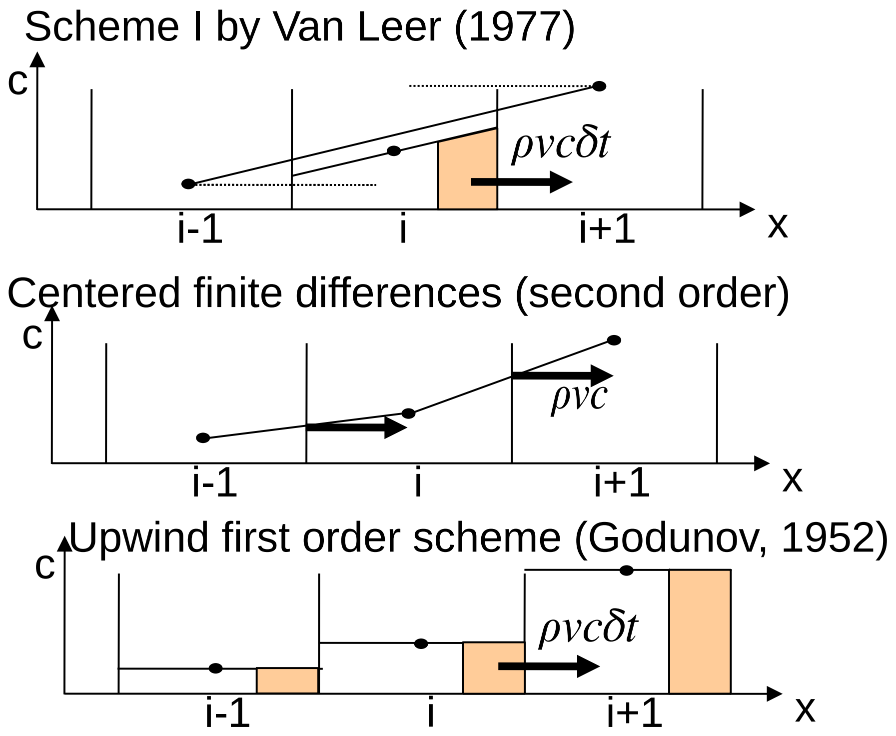
Explicit (B_n known) => $$\frac{f_{n+1} - f_n}{\delta t} \simeq R_n \text{ with BC: } \mathrm{B}_n$$

Implicit (B_{n+1} unknown) => $$\frac{f_{n+1} - f_n}{\delta t} \simeq R_{n+1} \text{ with BC: } \mathrm{B}_{n+1}$$

Tracer advection in LMDZ

Use of the Van Leer I scheme (1977), a second order finite volume scheme with slope limiters (e.g. MUSCL, MINMOD) (Hourdin et Armengaud, 1999).

Guaranties of fundamental physical properties of transport : conservation of the total quantity, positivity, monotony, non amplification of extrema, weak numerical diffusion



- CFL requirement, for an advection velocity Umax : Umax.(dt/dx) = cte , with cte ~ 0(1)

Tracer advection in LMDZ

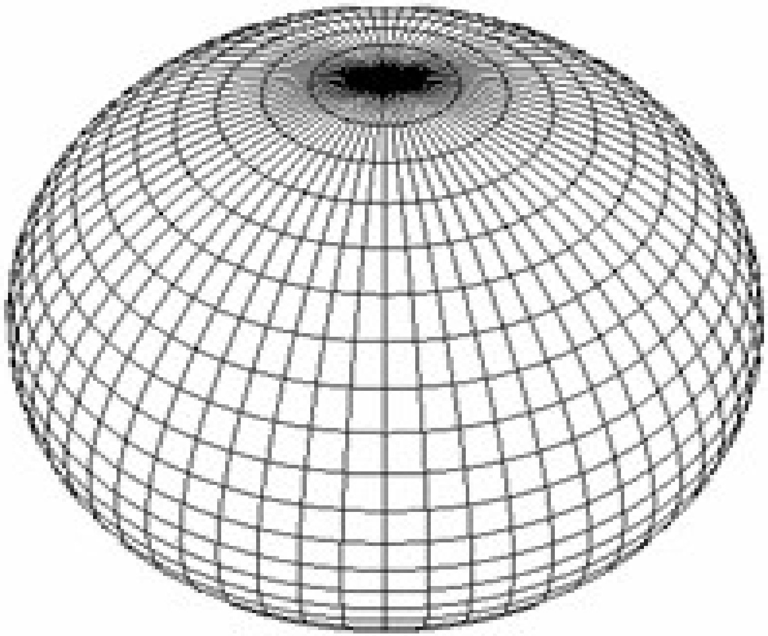
- In practice: Tracer names and various properties are set in the tracer.def file. e.g.:
  &version=1.0
  &lmdz
  default type=tracer phases=g hadv=10 vadv=10 parent=air
  H2O phases=g hadv=14 vadv=14
  H2O phases=l havd=10 vadv=10
  H2O phases=s hadv=10 vadv=10
  Rn
  Pb
- Scheme “10” : Van Leer scheme
- Scheme “14” : Dedicated modified scheme for water vapor
- Other (experimental) schemes are coded; see dyn3d/advtrac.F90

Tracer advection in LMDZ

- In practice: Tracer names and various properties are set in the tracer.def file. e.g.:
  &version=1.0
  &lmdz
  default type=tracer phases=g hadv=10 vadv=10 parent=air
  H2O phases=g hadv=14 vadv=14
  H2O phases=l havd=10 vadv=10
  H2O phases=s hadv=10 vadv=10
  Rn
  Pb
- phases: gas/liquid/solid for a given tracer
- parent: air or another tracer (for isotopes)+ possibility of “tagging”
  see https://lmdz-forge.lmd.jussieu.fr/mediawiki/LMDZPedia

Questions ?

The longitudinal polar filter



- A lon-lat grid implies that the meshes tighten dramatically as the pole is approached.
- CFL conditions there would dictate using an extremely small time step for the time marching scheme.

- Longitudinal (Fourier) filtering, removing high spatial frequencies, is used to enforce that resolved features are at the level of those at ~60° (where longitudinal resolution is half of that at the equator).
- In addition near the poles there is some longitudinal grouping of meshes (applied to the divergence of air transport) by bunches of 2^{ngroup} (typically ngroup=3) which implies that the number of points along longitudes of the GCM must be a multiple of 2^{ngroup}!

Energy spectra and lateral dissipation

- Observations (Nastrom & Gage 1985, Lindborg 1999) collected over length scales from a few to thousands of km display a characteristic energy cascade (from Skamarock, 2004).

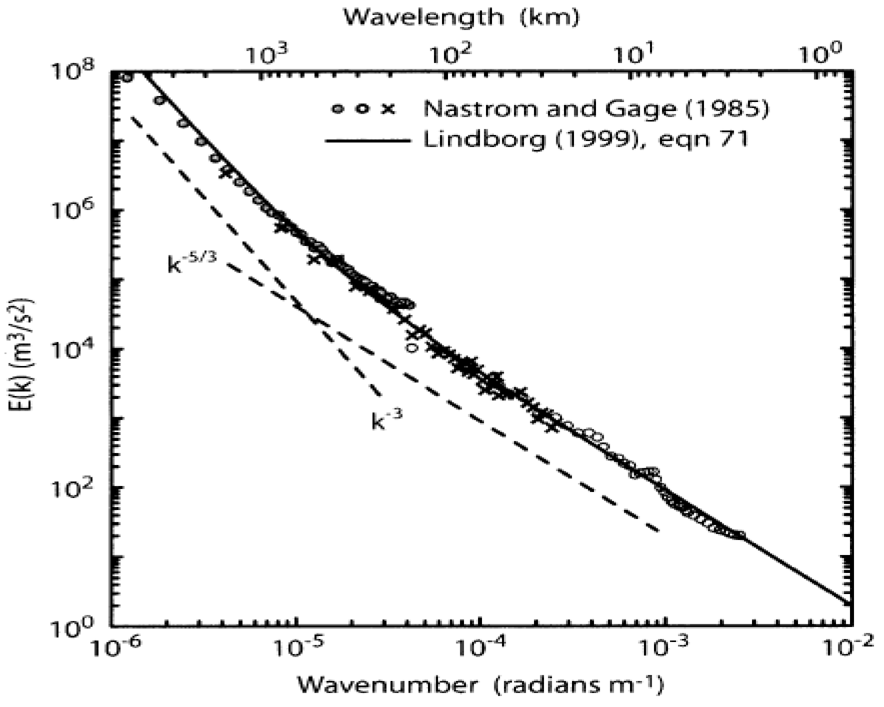


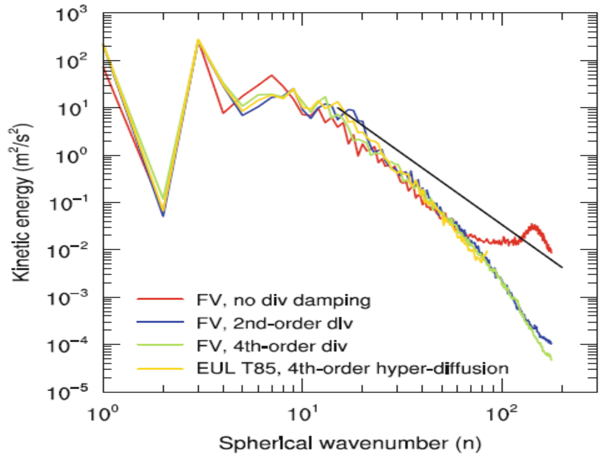
FIG. 1. Nastrom and Gage (1985) spectrum derived from the GASP aircraft observations (symbols) and the Lindborg (1999) functional fit to the MOZAIC aircraft observations.

- In order to fulfil the observed energy cascade from resolved scales to unresolved scales in GCMs, a dissipation term is added:

$$Dissip(\psi) = \frac{(-1)^{q+1}}{\tau} \nabla^{2q} \psi$$

Lateral dissipation in GCMs as a tool to pin the energy cascade

Figures from *Numerical Techniques for Global Atmospheric Models, Lauritzen et al. (eds), 2010*



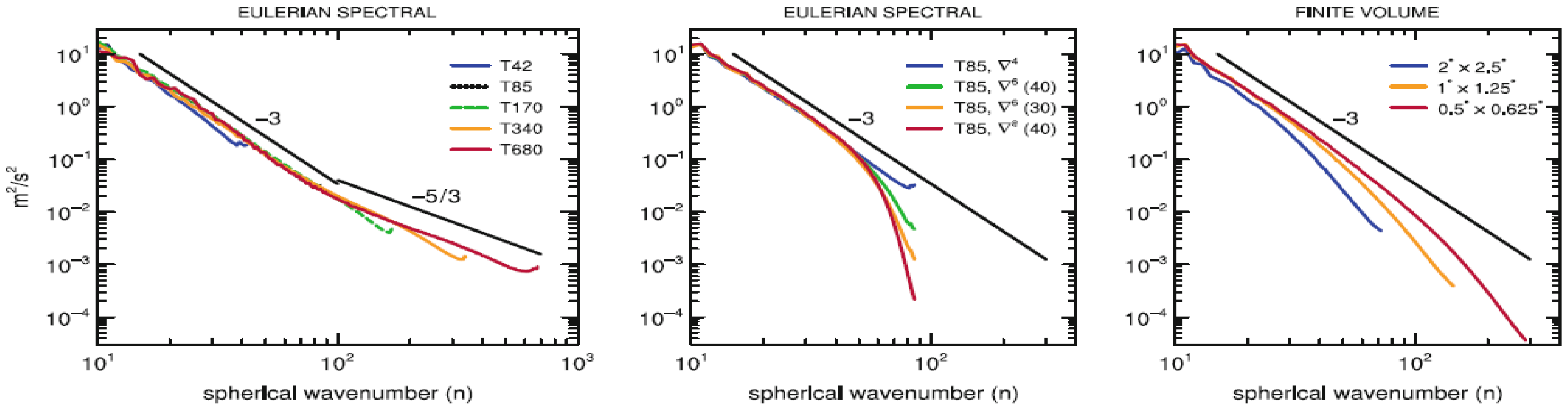


Fig. 13.4 250 hPa kinetic energy spectra as a function of the spherical wavenumber (n) in aqua-planet simulations from (*left*) CAM Eulerian spectral dynamical core with ∇^4 diffusion for different resolutions, (*center*) T85L26 Eulerian spectral dynamical with ∇^4, ∇^6 and ∇^8 diffusion, and (*right*) CAM Finite Volume (FV) dynamical core for different *lat* × *lon* resolutions in degrees and 26 levels

Illustrative example of dissipation

- Simple 1D diffusion equation toy model:

$$\frac{dA}{dt} = \nu \frac{\partial^2 A}{\partial x^2}$$

- Von Neumann (Fourier mode) analysis

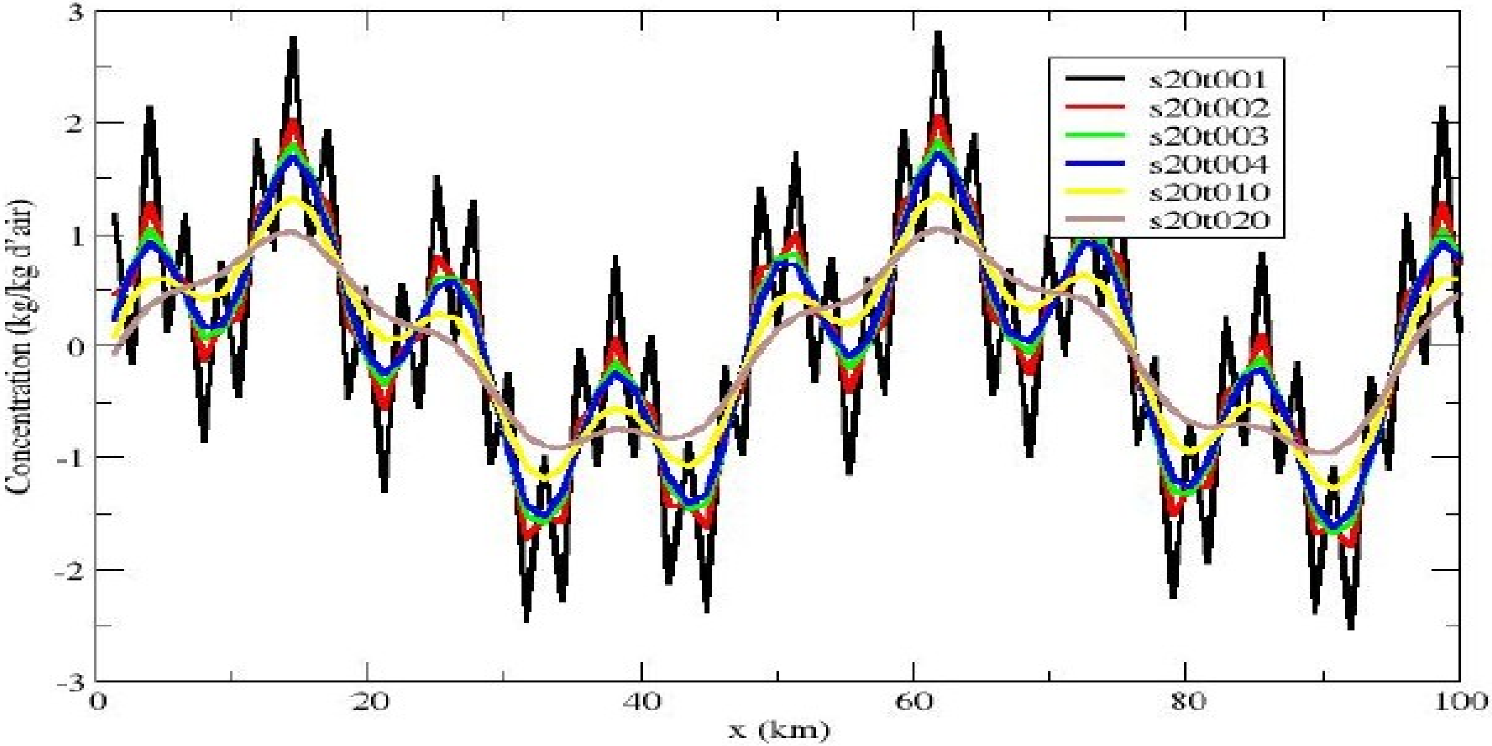
$$A_k(t) = a_k(t).sin(kx)$$

- Explicit Euler time marching (with stability condition!):

$$a_k^{n+1} = \left(1 - \nu k^2 \Delta t\right) a_k^n$$

Note that mode damping is stronger for large k

Illustrative example of dissipation



Temporal evolution, from an initial condition consisting of 2 sine modes and an extreme (2 grid points wavelength) “numerical mode”

Controlling dissipation in LMDZ

- Parameters in file gcm.def:

dissip_period: Apply dissipation every dissip_period dynamical steps (or **specify 0 to let model pick an appropriate value**)

nitergdiv: number of iterations on velocity dissipation operator grad.div

nitergrot: number of iterations on velocity dissipation operator grad.rot

niterh: number of iterations on temperature dissipation operator div.grad

Usual values: nitergdiv=1, nitergrot=2, niterh=2

tetagdiv: dissipation time scale (s) for smallest wavelength for u,v (grad.div component)

tetagrot: dissipation time scale (s) for smallest wavelength for u,v (grad.rot component)

tetatemp: dissipation time scale (s) for smallest wavelength for potential temperature (div.grad)

values depend on horizontal resolution

Controlling dissipation in LMDZ

- Parameters in file gcm.def:

  tetagdiv: dissipation time scale (s) for smallest wavelength for u,v (grad.div component)

  tetagrot: dissipation time scale (s) for smallest wavelength for u,v (grad.rot component)

  tetatemp: dissipation time scale (s) for smallest wavelength for potential temperature (div.grad)

  optimal teta values depend on horizontal resolution

- Moreover there is a multiplicative factor for the dissipation coefficient, which increases with model levels (see dyn3d_common/inidissip.F90), which can be controlled by flag "vert_prof_dissip"

The sponge layer

- In addition to lateral dissipation, it is necessary to damp vertically propagating waves (non-physically reflected downward from model top).
- The sponge layer is limited to topmost layers (usually 4) and added during the dissipation step.
- Sponge modes and parameters (gcm.def):

  iflag_top_bound: 0 for no sponge, 1 for sponge over 4 topmost layers, 2 for sponge from top to 100 times topmost layer pressure

  mode_top_bound: 0 for no relaxation, 1 to relax u,v to zero, 2 to relax u,v to their zonal mean, 3 to relax u,v and potential temperature to their zonal mean.

  tau_top_bound: inverse of characteristic time scale at the topmost layer (halved at each successive descending layer)

Where to get typical values of the run.def/gcm.def parameters

- Check out the various examples of gcm.def_* files located in the **LMDZ/DefLists** subdirectory
- Some examples of specific cases of Zoomed simulation setups are detailed in a collaborative document; get the link by searching “Zoom collection” on LMDZPedia :
 https://lmdz-forge.lmd.jussieu.fr/mediawiki/LMDZPedia

LMDZ_Zoom_collection

Zoom region	Resolution/ Model version/ Physics version	Resolution (° or km) on the zoomed area	Nudging (Y/N)	day_step	iphysic	tetagdiv	tetagrot	tetatemp	clon/clat	grossismx/ grossismy	dzoomx/ dzoomy	taux/ tauy
Europe	128x118(x39?)_NPv3.2		y	2880	15	1200	1200	1200	8.5E / 50N	4.48751 / 3.40784	0.208914 / 0.243706	10/10
Europe	128x88x79_Npv6.0,15T		y	2160	15	900	900	900	22E / 48N	2.62766 / 2.856	0.267585 / 0.169102	6/4
France	9695_79_NPv6_0_15		y	2160	15	900	900	1800	2.208E / 48.713N	5.5278 / 4.23288	0.0603014 / 0.0795774	3/3
Maroc-Afrique du Nord	144x142x79/NPv6.1	50 x 38 km	n	3360	35	720	900	900	8W / 30N	5 / 5	0.1 / 0.1	5/5

Some rules of thumb for run.def parameters

- Time steps in LMDZ:

  dynamical time steps: dtvr = daysec / day_step

  physics time step: dtphys = iphysiq * dtvr

  dissipation time step: dtdiss = dissip_period * dtvr

  tracer advection time step: dtvrtrac = iapp_trac * dtvr

- Constraints to be aware of:

  dtvr limited by CFL for waves: Cmax.dt < min(dx,dy)

  dtrtrac limited by advection CFL: Umax.dt < min(dx,dy)

  iphysiq, dtvrtrac, dissip_period should be multiples of iperiod

Some rules of thumb for run.def parameters

- Constraints to be aware of (continued):

  dissipation time step should be much smaller than dissipation timescales:

  dtdiss << tetatgdiv, tetagrot, tetatemp

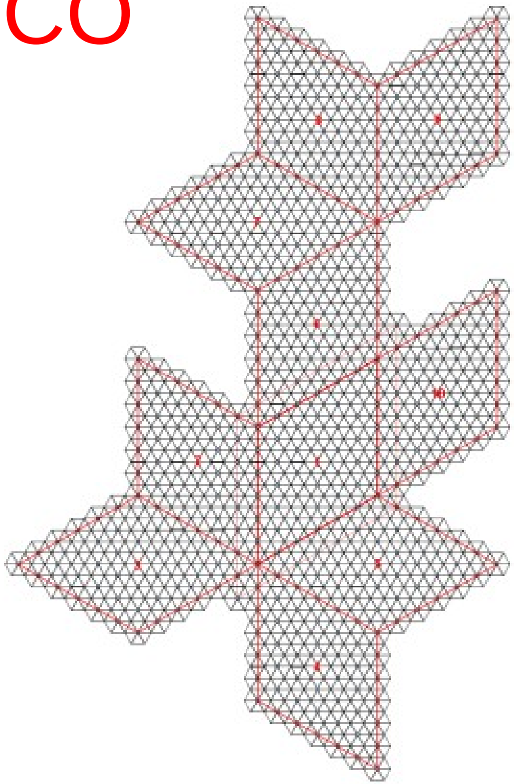
- Changing time step with resolution on a regular grid:

  day_step(max(iim,jjm)=N) ~ day_step(max(iim,jjm)=M) * M/N

- Time step for a zoomed simulation, compared to regular grid:

  day_step(zoom) ~ day_step(regular)*max(grossismx,grossismy)

LMDZ + DYNAMICO

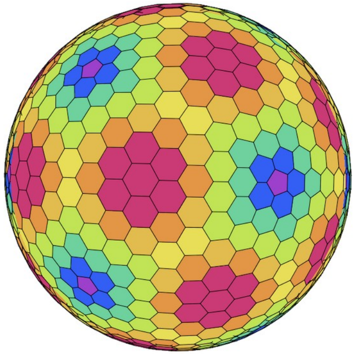


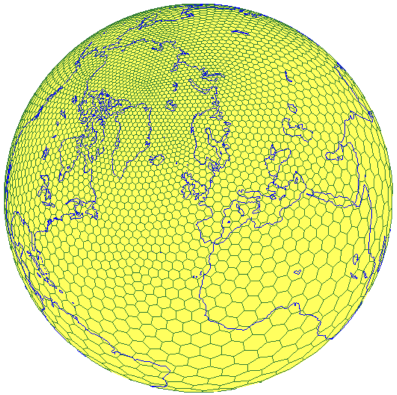
An icosahedral grid…

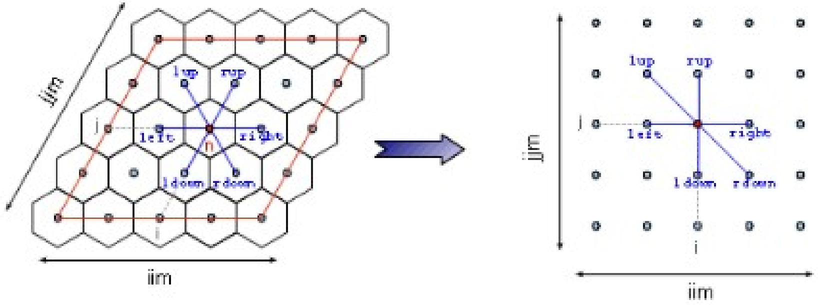
- Cells are mostly (irregular) hexagons, some pentagons
- No zonal filters => more parallelism
- Makes other things trickier !

.. but a similar “user experience”.

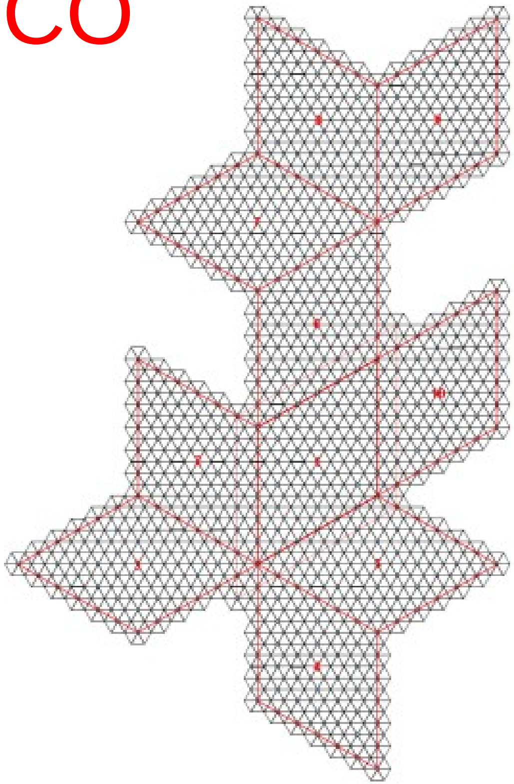
Especially, files are read/written by XIOS which can interpolate between DYNAMIC0 and lon-lat mesh. Except for a good reason, ***input and output fields are longitude-latitude***







LMDZ + DYNAMICO



An icosahedral grid…

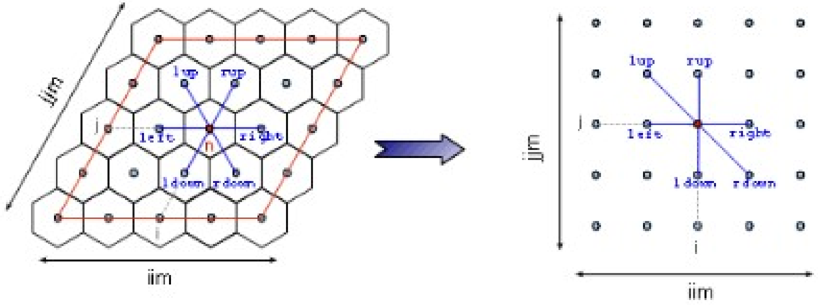
- Cells are mostly (irregular) hexagons, some pentagons
- No zonal filters => more parallelism
- Makes other things trickier !

.. but a similar “user experience”.

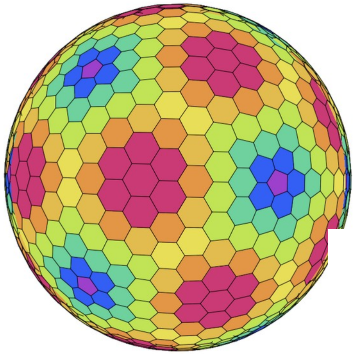
Especially, files are read/written by XIOS which can interpolate between DYNAMIC0 and lon-lat mesh. Except for a good reason, ***input and output fields are longitude-latitude***

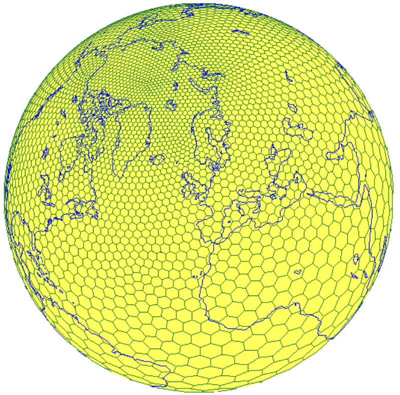
Horizontal resolution

- Controlled by a single parameter “nbp”
- 10 x nbp x nbp grid points (atmospheric columns)
- nbp=40 => about 2 degree resolution (dx ~ 220km)
- dx is divided by 2 when nbp doubles
- maximum dynamics time step: **dt ~ dx/c , c~400 m/s**



LMDZ + DYNAMICO



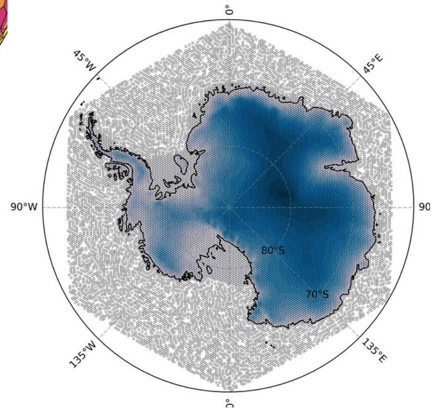


An icosahedral grid…

- Cells are mostly (irregular) hexagons, some pentagons
- No zonal filters => more parallelism
- Makes other things trickier !

.. but a similar “user experience”.

Especially, files are read/written by XIOS which can interpolate between DYNAMIC0 and lon-lat mesh. Except for a good reason, ***input and output fields are longitude-latitude***



Horizontal resolution

- Controlled by a single parameter “nbp”
- 10 x nbp x nbp grid points (atmospheric columns)
- nbp=40 => about 2 degree resolution (dx ~ 220km)
- dx is divided by 2 when nbp doubles
- maximum dynamics time step: **dt ~ dx/c , c~400 m/s**

Zoom

- “polar” zoom via Schmidt transform
- soon (?) via variable-resolution meshing

Limited Area-Model

- metric_type = icosa_area
- set center (lon, lat) and radius
- dx = radius/nbp
- Needs other parameters, and **input files**

LMDZ + DYNAMICO

Transport schemes

- Scheme similar to lon-lat Van Leer
- Transport time step can be a small multiple of dynamics time step (itau_adv)

Horizontal dissipation

- (bi-)laplacian, exactly as lon-lat LMDZ
- Coefficients tuned for a certain resolution

Sponge layer

- Relaxation to zero: simple but breaks total angular momentum conservation
- Relaxation to zonal mean: implemented (via XIOS) by expensive...
- Relaxation to temporal mean: implemented (via XIOS), local, so less expensive

Nudging

- implemented

(incomplete) doc of DYNAMICO parameters/flags:
https://gitlab.in2p3.fr/ipsl/projets/dynamico/dynamico/-/blob/master/PARAMETERS.md?ref_type=heads

Search for DYNAMICO on LMDZPedia!:
https://lmdz-forge.lmd.jussieu.fr/mediawiki/LMDZPedia/index.php/Sp%C3%A9cial:Recherche?search=dynamico&fulltext=Recherche+en+texte+int%C3%A9gral&fulltext=Search