

# LMDZ

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

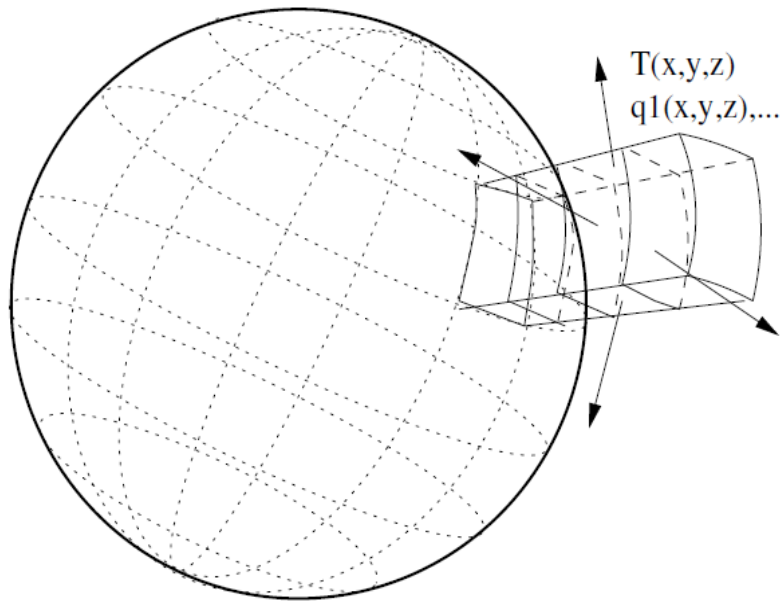
LMDZ courses, December 19 2018

# 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

*Dynamics*



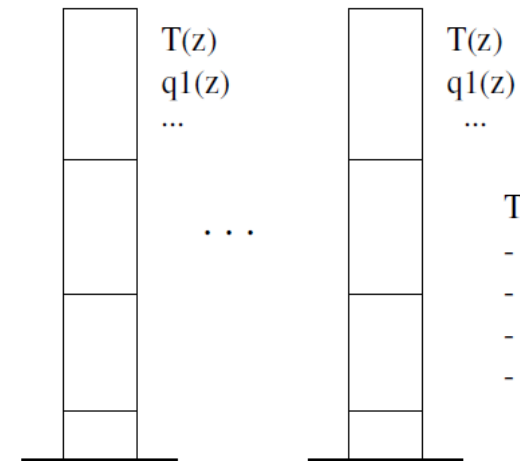
Dynamical tendencies



Physical fields



*Physics*



Tendencies due to  
- radiative transfer  
- condensation  
- subgrid dynamics  
- ...

## 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:

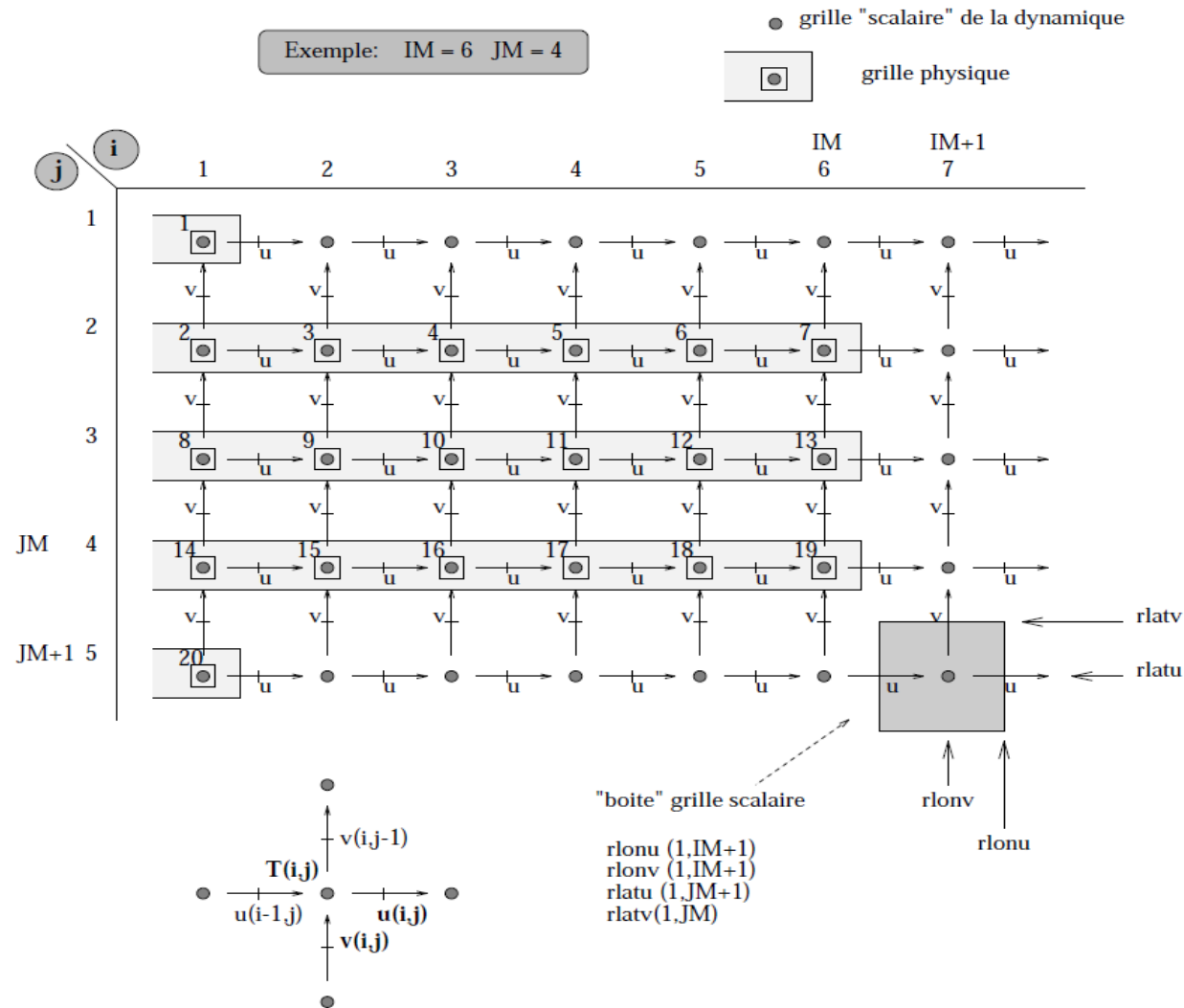
`makeImdz[_fcm] -d iimxjjmxllm ...`

In the dynamics:

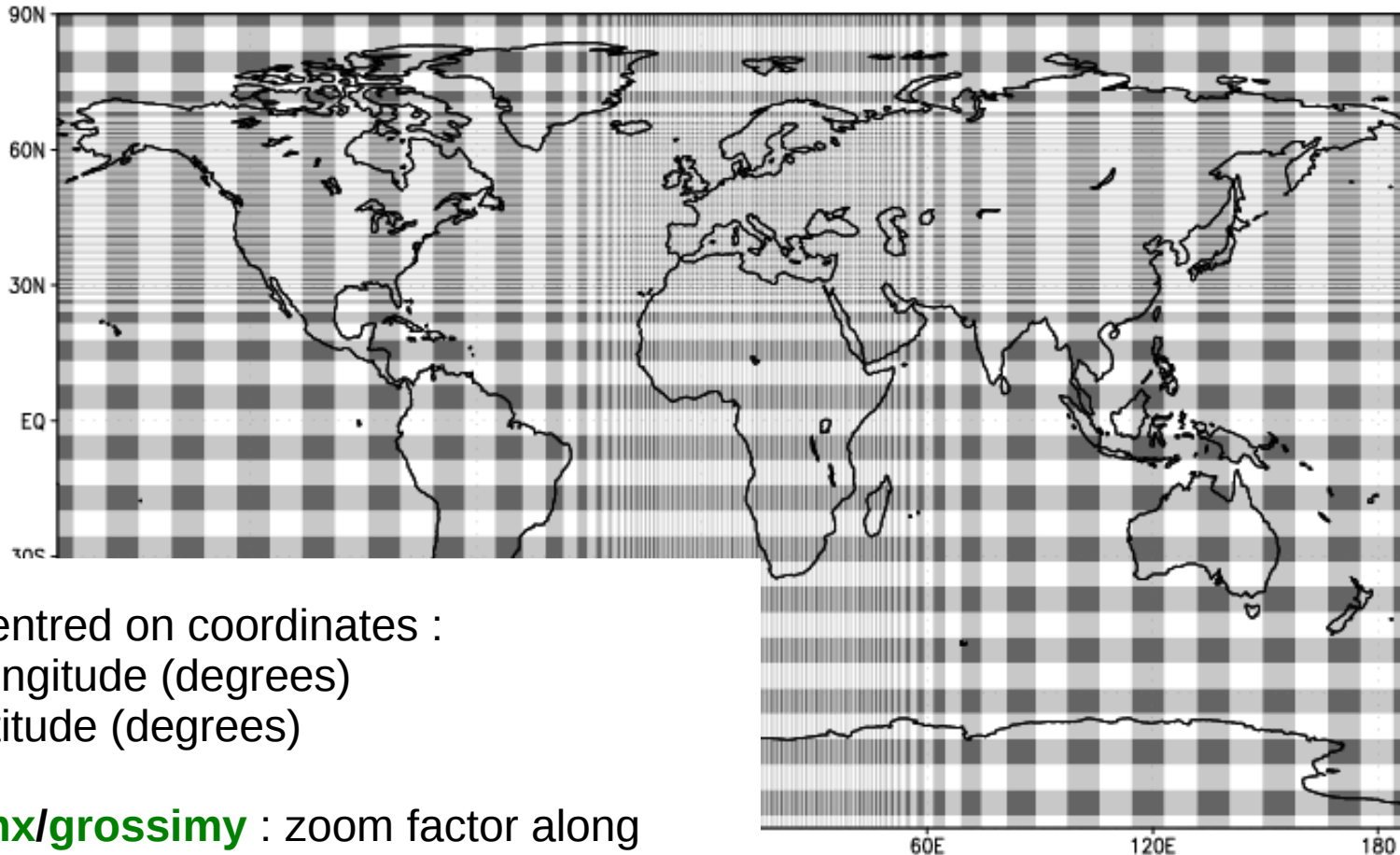
- Staggered grids, u, v and scalars (temperature, tracers) are on different meshes
- Global lonxlat grids with redundant grid points
  - at the poles
  - in longitude

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)

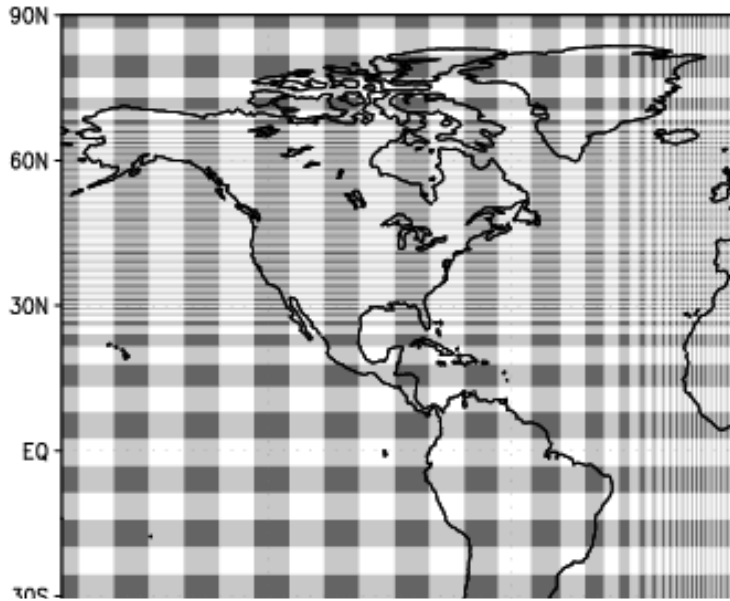
**clat** : latitude (degrees)

**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:  $dzoom \times 360^\circ$  by  $dzoomy \times 180^\circ$

# LMDZ, Z for Zoom



Zoom centred on coordinates :

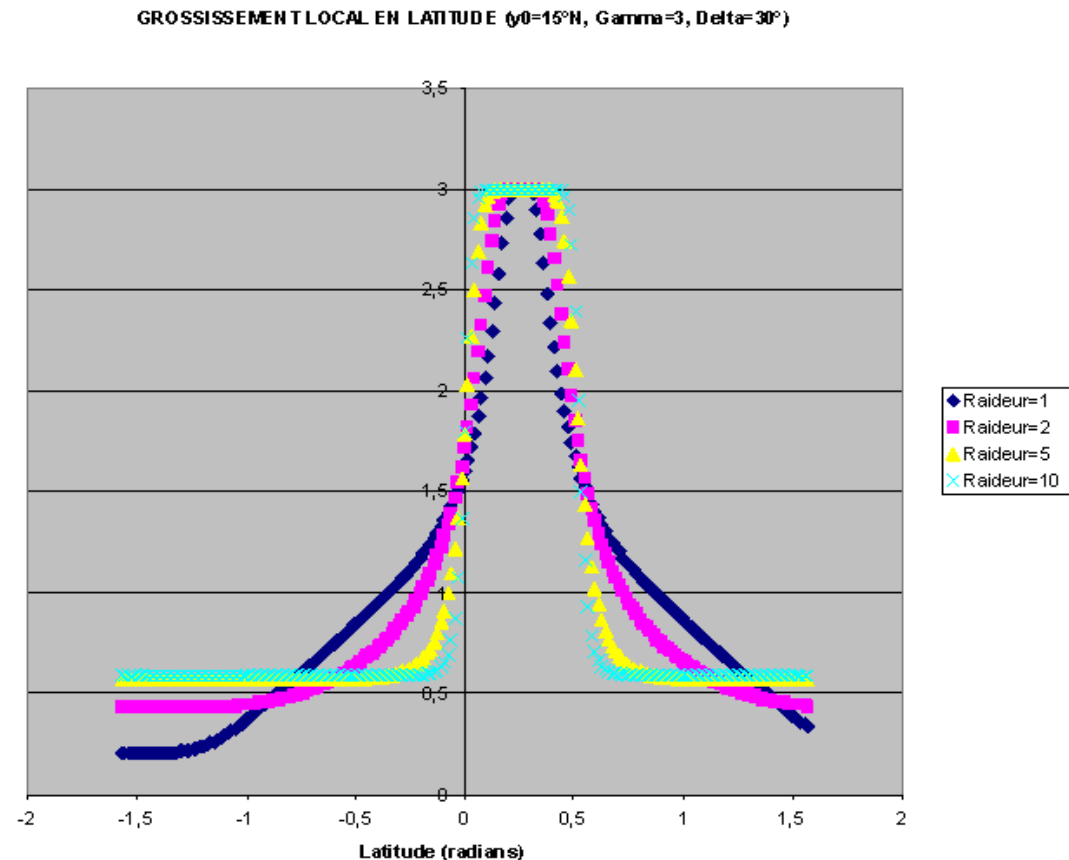
**clon** : longitude (degrees)

**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;  $\tau \sim 3$  is a reasonable value)



# Nudging in LMDZ

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial t}_{GCM} + \frac{u_{analyse} - u}{\tau}$$
$$\frac{\partial v}{\partial t} = \frac{\partial v}{\partial t}_{GCM} + \frac{v_{analyse} - v}{\tau}$$

Nudging towards  
analyses or reanalyses with  
given time constants

$u_{analyse}$   $v_{analyse}$

Example of nudging parameters:

$ok\_guide = y$

$guide\_T = n$  ,  $guide\_p = n$  ,  $guide\_q = n$

$guide\_u = y$  ,  $guide\_v = y$

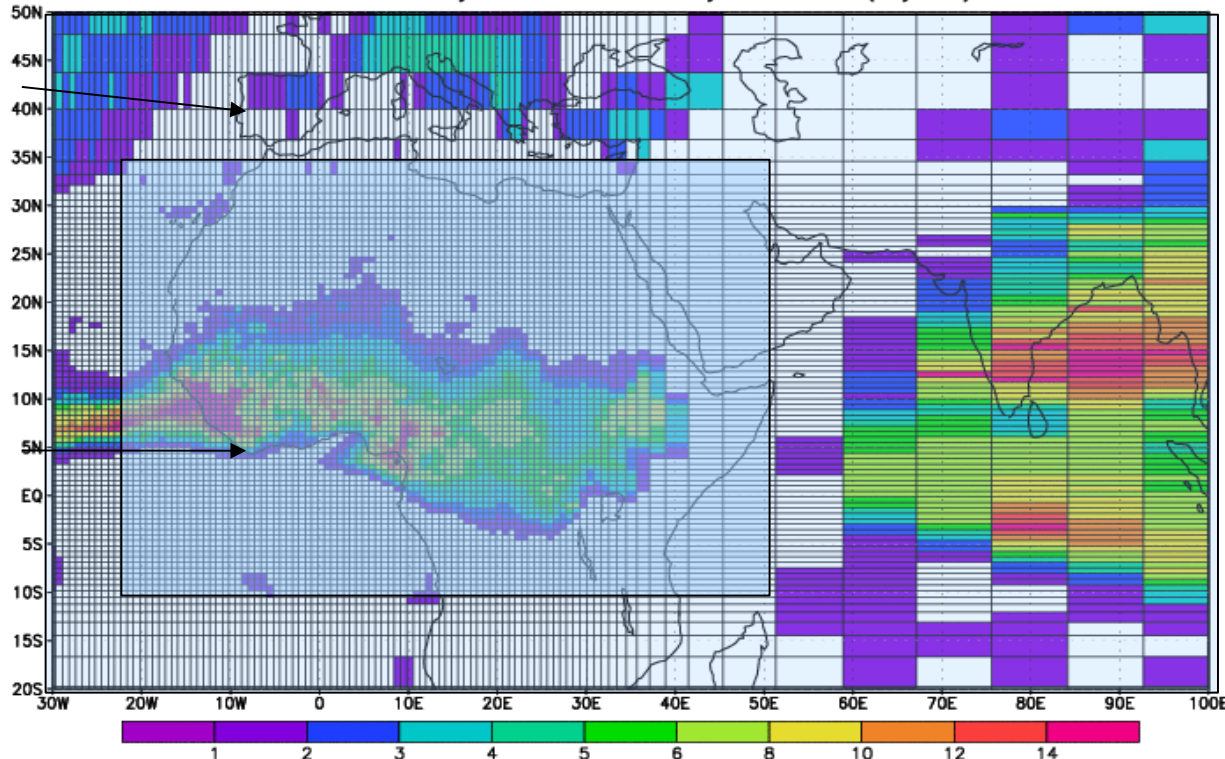
$tau\_min\_u = 0.0208333$  (days)

$tau\_max\_u = 10$  (days)

$tau\_min\_v = 0.0208333$  (days)

$tau\_max\_v = 10$  (days)

July rainfall (mm/day)  
LMDZ forced by the mean annual cycle of SSTs (6 years)



Strong nudging  
( $\tau=30\text{min}$ )

Weak to  
moderate  
nudging  
( $\tau=10$  days)

# Vertical discretization in LMDZ

- Model levels are hybrid **sigma-pressure** levels:

$$P(\text{level}, \text{time}) = \mathbf{ap}(\text{level}) + \mathbf{bp}(\text{level}) \cdot \mathbf{Ps}(\text{time})$$

hybrid coordinates  $\mathbf{ap}(k)$  and  $\mathbf{bp}(k)$  are fixed for a given model run

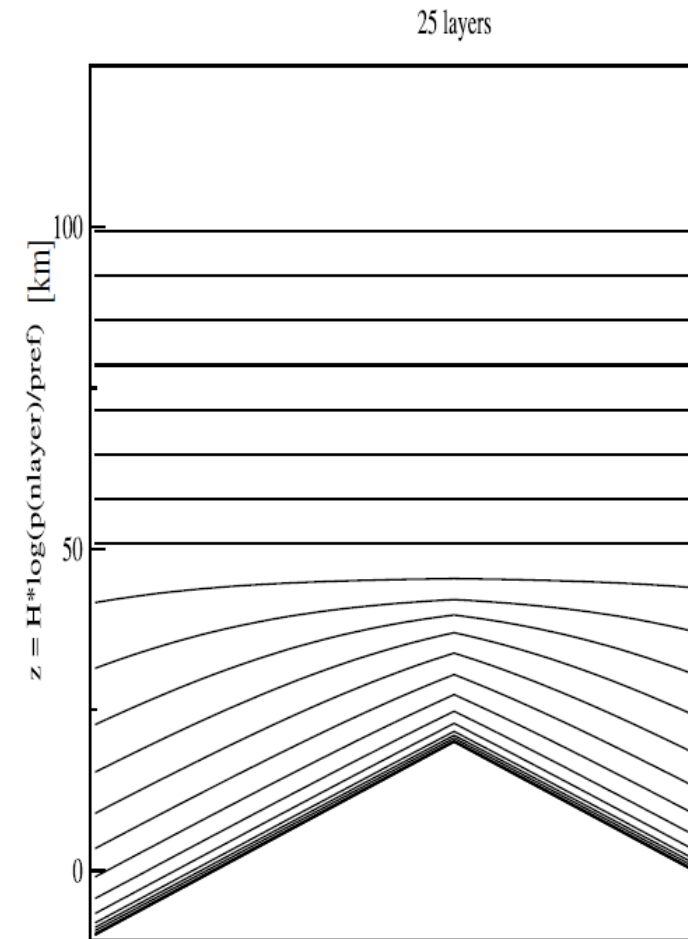
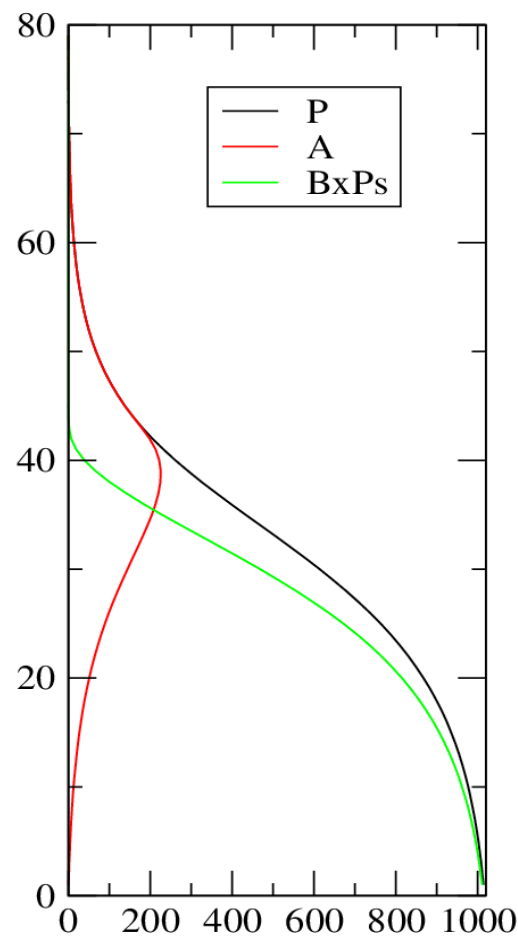
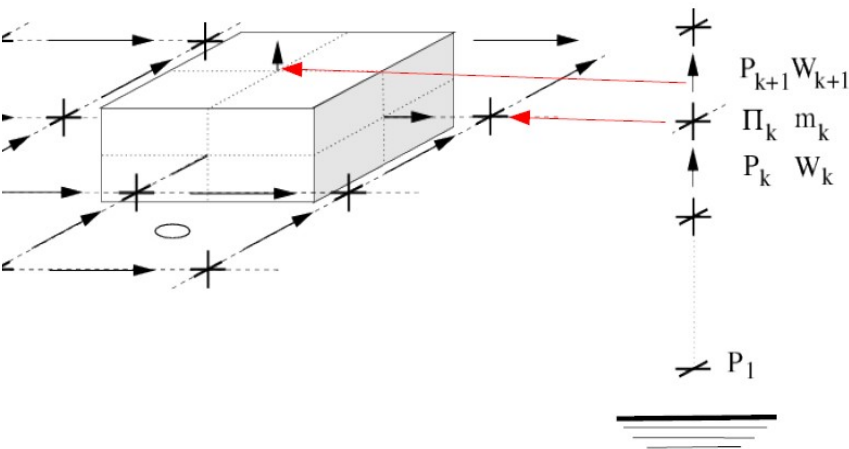
Surface pressure  $\mathbf{Ps}(t)$  varies during the run

- Near the surface  $\mathbf{ap} \sim 0$

$$\Rightarrow \mathbf{bp}(k) \sim P/P_s$$

- At high altitudes,  $\mathbf{bp} \sim 0$

$$\Rightarrow \mathbf{ap}(k) \sim P$$





# Vertical discretization in LMDZ

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

**vert\_sampling = strato\_custom** : customizable (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”

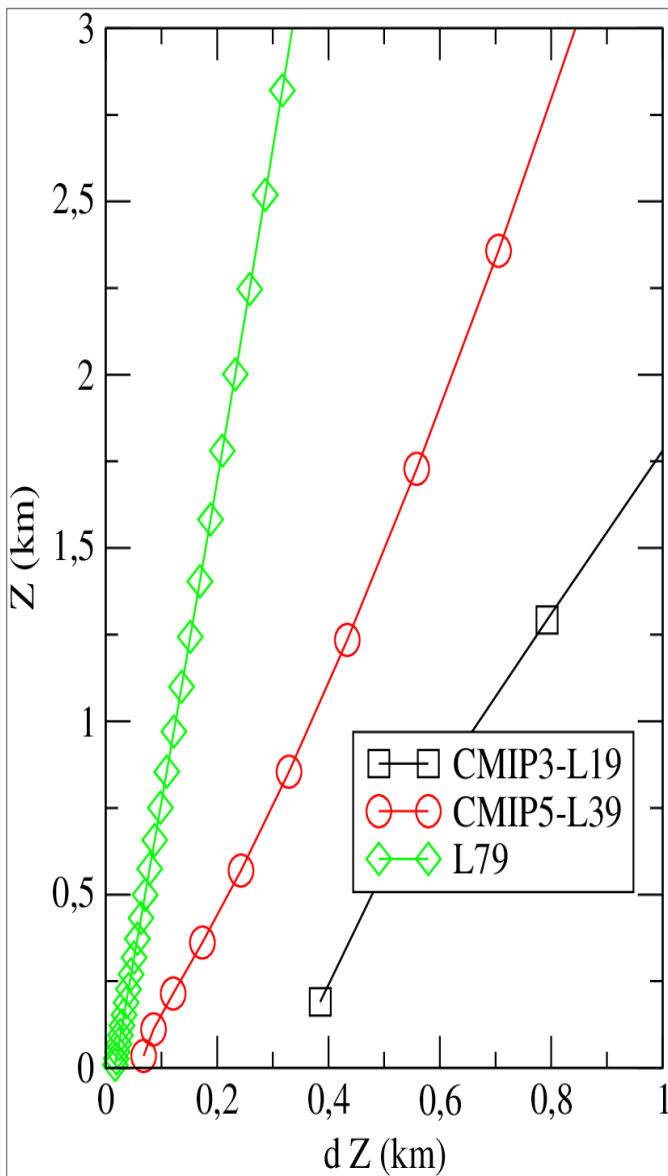
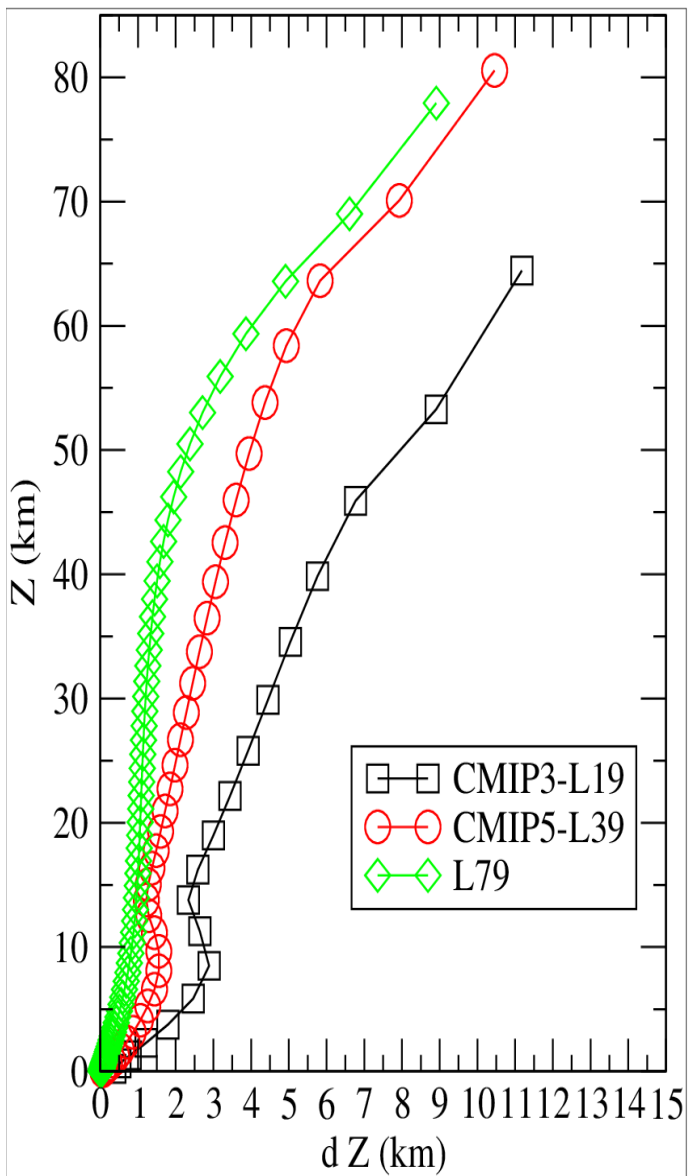
=> Typically you don't need to mess with the vertical discretization,

**the default behaviour most likely matches your needs.**

=> Check out routine `dyn3d_common/disvert.F90`

# Vertical discretization in LMDZ

- Example of typical altitudes of L19/L39/L79 vertical layer position and thickness



## “Standard” L79 settings

`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.`

# Time marching schemes

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

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

- And 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**  $t=n \cdot dt$  the right hand side term  $R[f(t), t]$  is to be evaluated

# Time marching schemes

- **Explicit Euler** scheme (1<sup>st</sup> order in time):

$$\frac{df(t)}{dt} \simeq \frac{f_{n+1} - f_n}{\delta t}$$

$$R(f, t) \simeq R(f(t_n), t_n)$$

- **Implicit Euler** scheme (1<sup>st</sup> order in time):

$$\frac{df(t)}{dt} \simeq \frac{f_{n+1} - f_n}{\delta t}$$

$$R(f, t) \simeq R(f(t_{n+1}), t_{n+1})$$

- **Crank-Nicholson** scheme (2<sup>nd</sup> order in time):

$$\frac{df(t)}{dt} \simeq \frac{f_{n+1} - f_n}{\delta t}$$

$$R(f, t) \simeq \frac{R(f(t_{n+1}), t_{n+1}) + R(f(t_n), t_n)}{2}$$

# Time marching schemes

- **Matsuno** scheme: a predictor-corrector (Euler explicit-Euler Implicit) scheme (1<sup>st</sup> order):

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

- **Leapfrog** scheme: use encompassing time steps to evaluate the derivative (2<sup>nd</sup> order):

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

# 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:

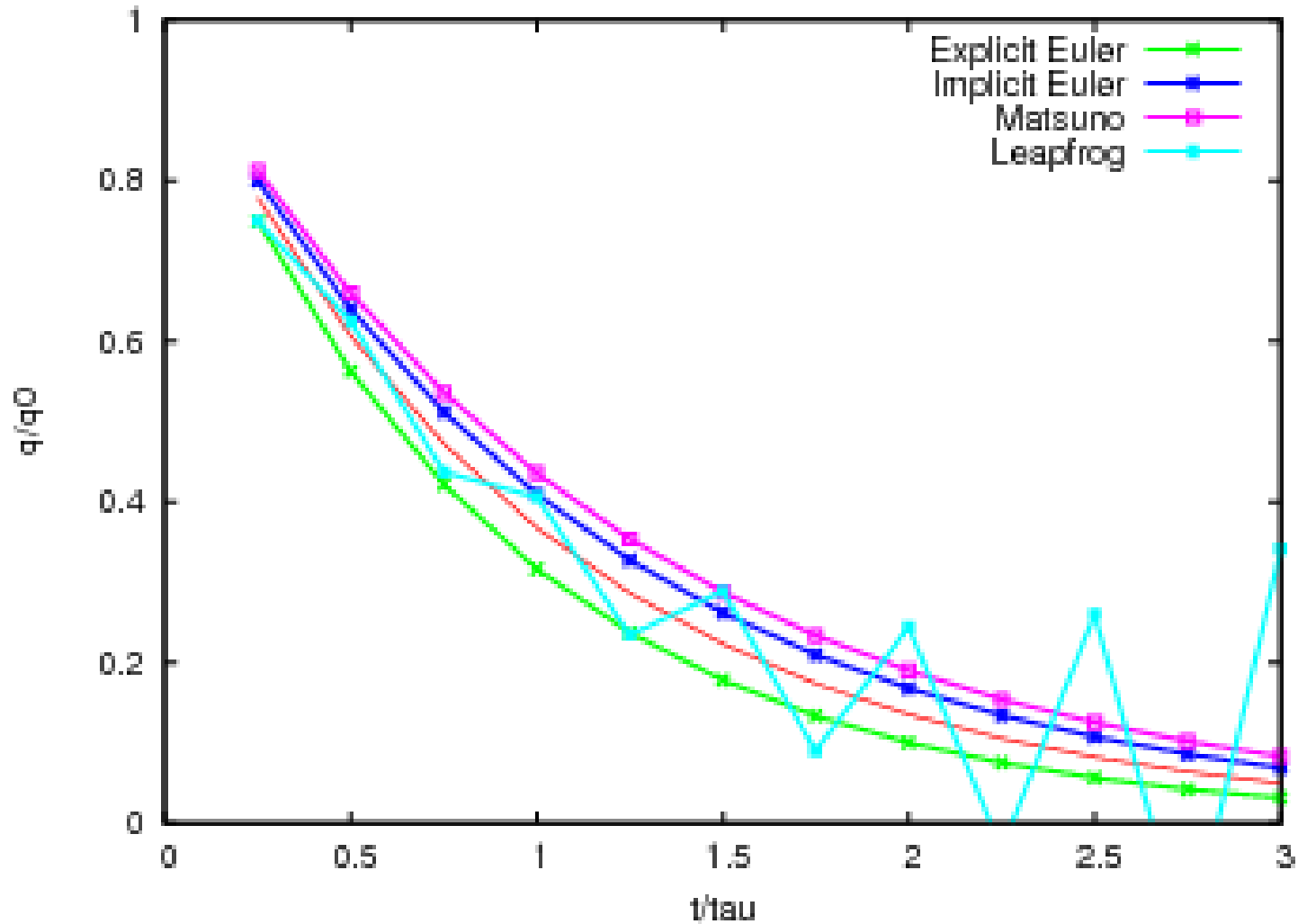
$$\text{EE : } q^{n+1} = \left[ 1 - \frac{\delta t}{\tau} \right] q^n$$

$$\text{EI : } q^{n+1} = \left[ \frac{1}{1 + \delta t/\tau} \right] q^n$$

$$\text{CN : } q^{n+1} = \left[ \frac{1 - \delta t/(2\tau)}{1 + \delta t/(2\tau)} \right] q^n$$

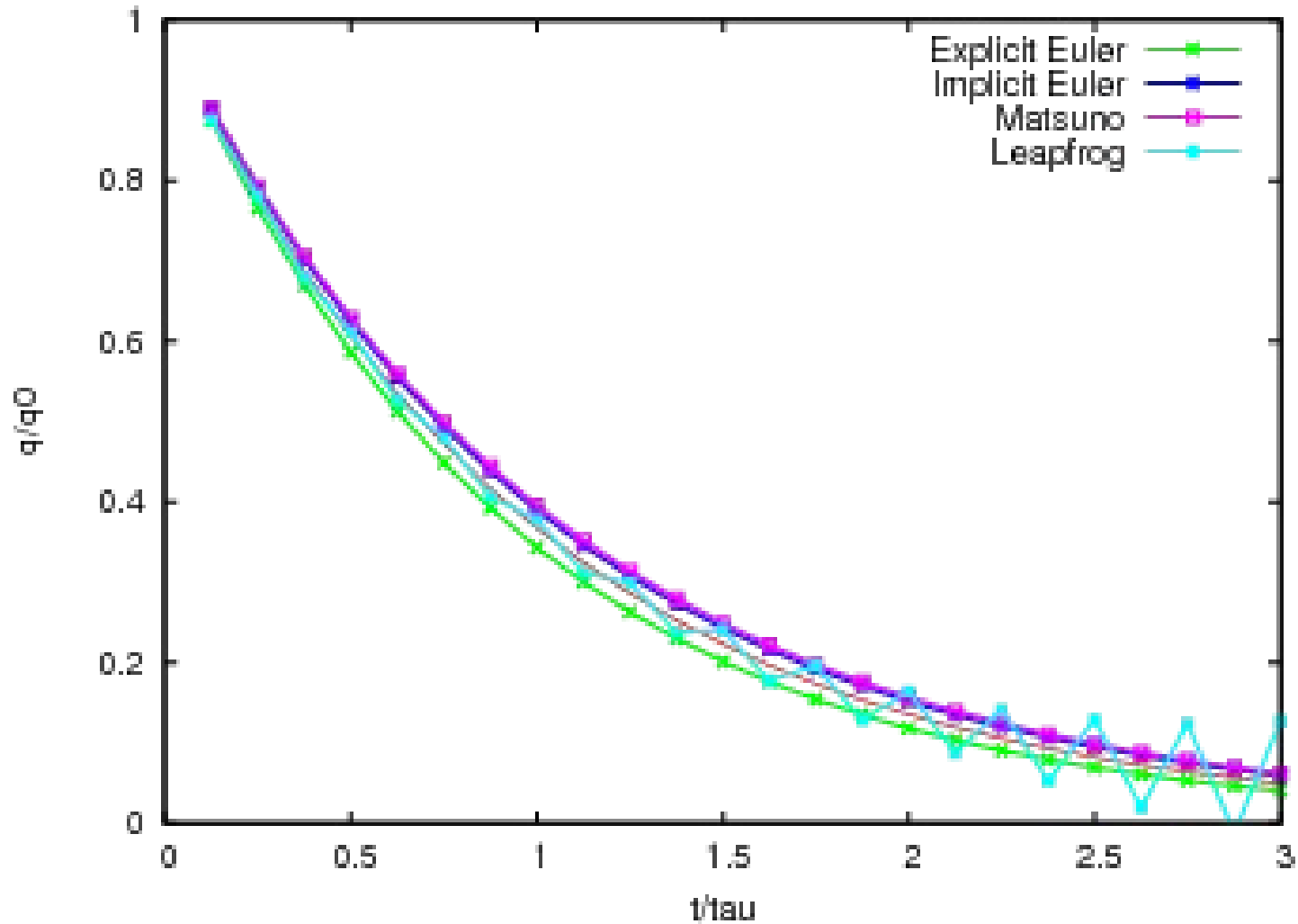
- Stability requirement (CFL) for EE :  $dt/\tau < 2$

# Time marching schemes



- 4 integration steps per unit of  $t/\tau$

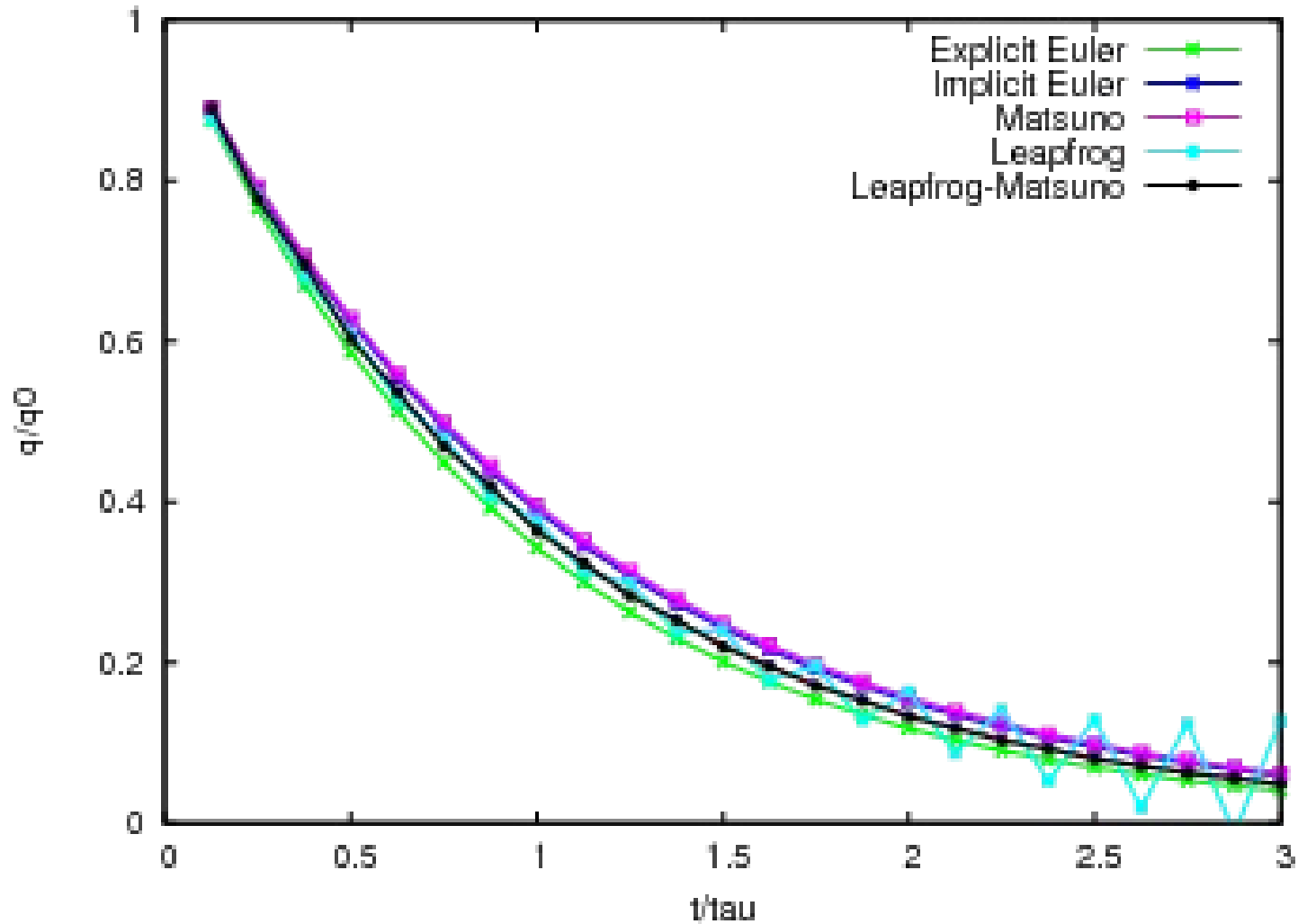
# Time marching schemes



- 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 Matsuno 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) :

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

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

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

=> **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 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:

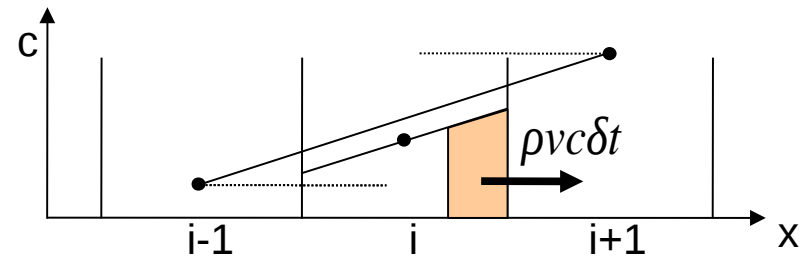
- 1)  $L.U=T$  ,  $L$  and  $U$  are bidiagonal (lower/upper) matrices
- 2) Solve  $L.z=y$  for vector  $z$  (forward substitution step)
- 3) Solve  $U.x=z$  for vector  $x$  (backward substitution step)

# Tracer advection in LMDZ

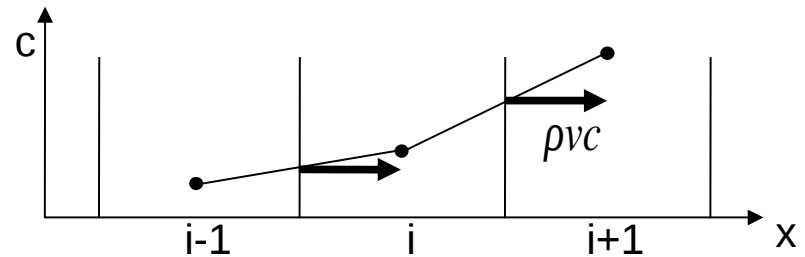
Use of the [Van Leer I scheme](#) (1977), a second order finite volume scheme with slope limiters (e.g. MUSCL, MINMOD) (Hourdin et Armengaud, 1999).

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

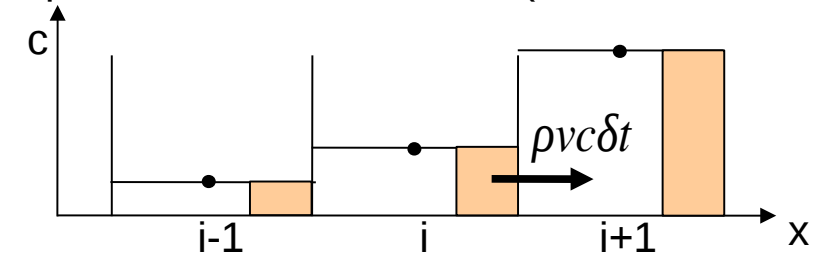
Scheme I by Van Leer (1977)



Centered finite differences (second order)



Upwind first order scheme (Godunov, 1952)



- **CFL requirement**, for an advection velocity  $U_{max}$  :  
 $U_{max} \cdot (dt/dx) = cte$  , with  $cte \sim 0(1)$

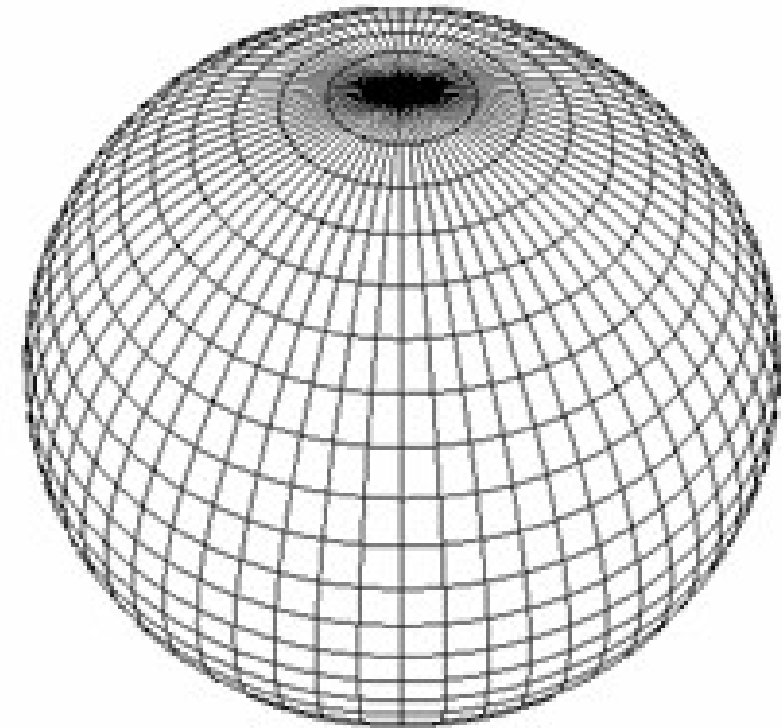
# Tracer advection in LMDZ

- In practice: Tracer names and advection schemes are set in the `traceur.def` file. e.g.:

```
5                <- Total number of tracers
14 14 H2Ov       <- Tracer advection scheme and name
10 10 H2OI       <- Tracer advection scheme and name
10 10 H2Oi       <- ...
10 10 RN
10 10 PB
```

- Scheme “10” : Van Leer scheme
- Scheme “14” : Specific modified scheme for water vapor
- Other (experimental) schemes are coded; see [dyn3d/advtrac.F90](#)

# 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  $\sim 60^\circ$

# 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).

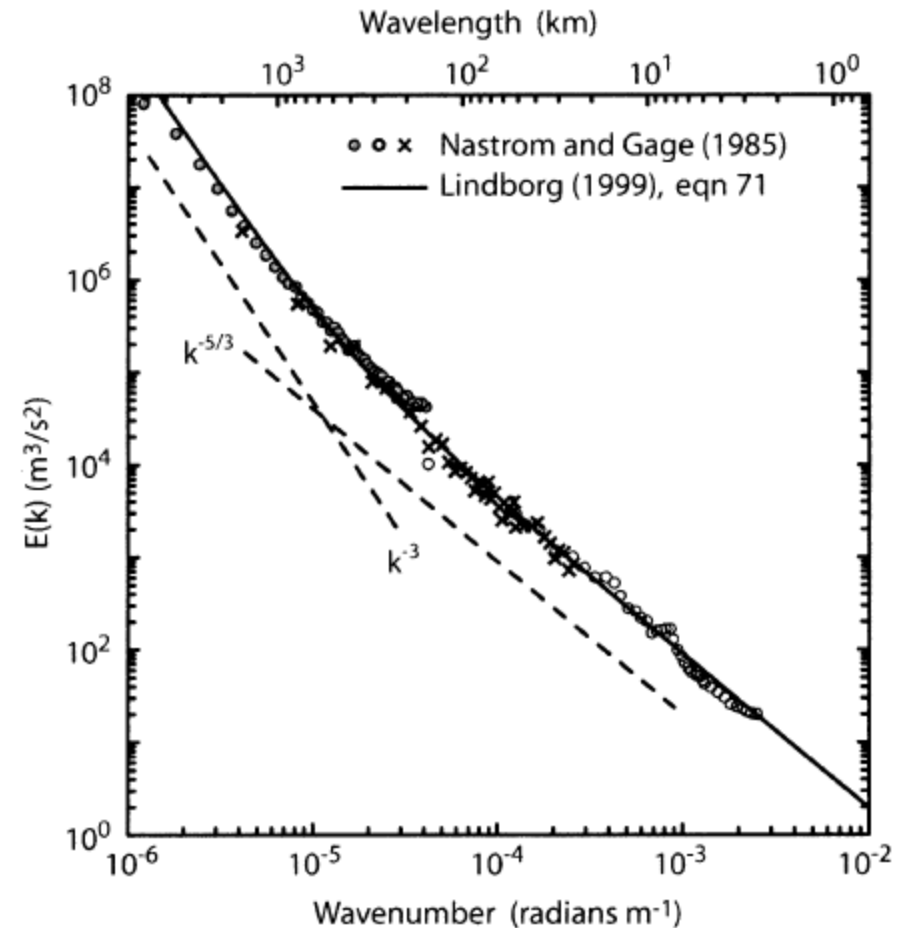


FIG. 1. Nastrom and Gage (1985) spectrum derived from the GASP aircraft observations (symbols) and the Lindborg (1999) functional fit to the MOZAIC aircraft observations.

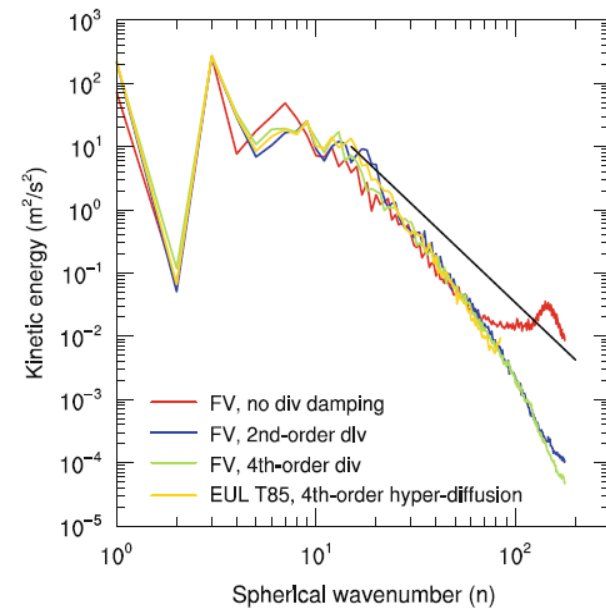
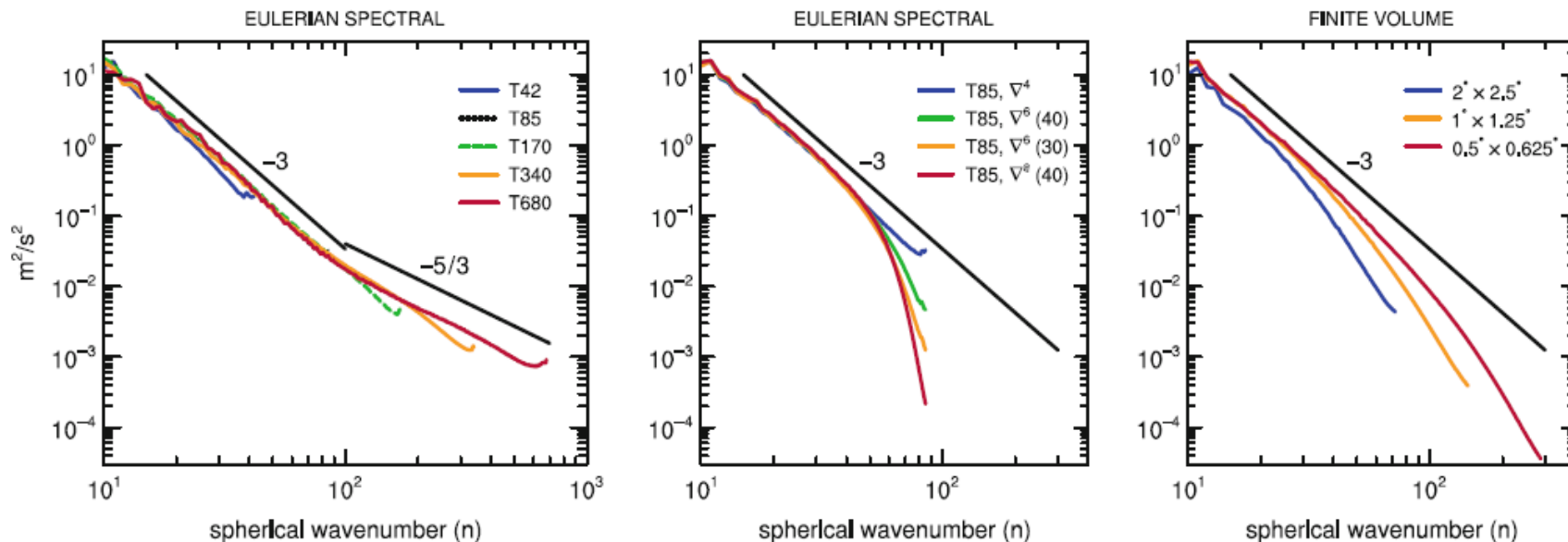
- In order to fulfil the observed energy cascade from resolved scales to unresolved scales in GCMs, a **dissipation term** is added:

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



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

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 aquaplanet simulations from (left) CAM Eulerian spectral dynamical core with  $\nabla^4$  diffusion for different resolutions, (center) T85L26 Eulerian spectral dynamical with  $\nabla^4$ ,  $\nabla^6$  and  $\nabla^8$  diffusion, and (right) CAM Finite Volume (FV) dynamical core for different  $lat \times lon$  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

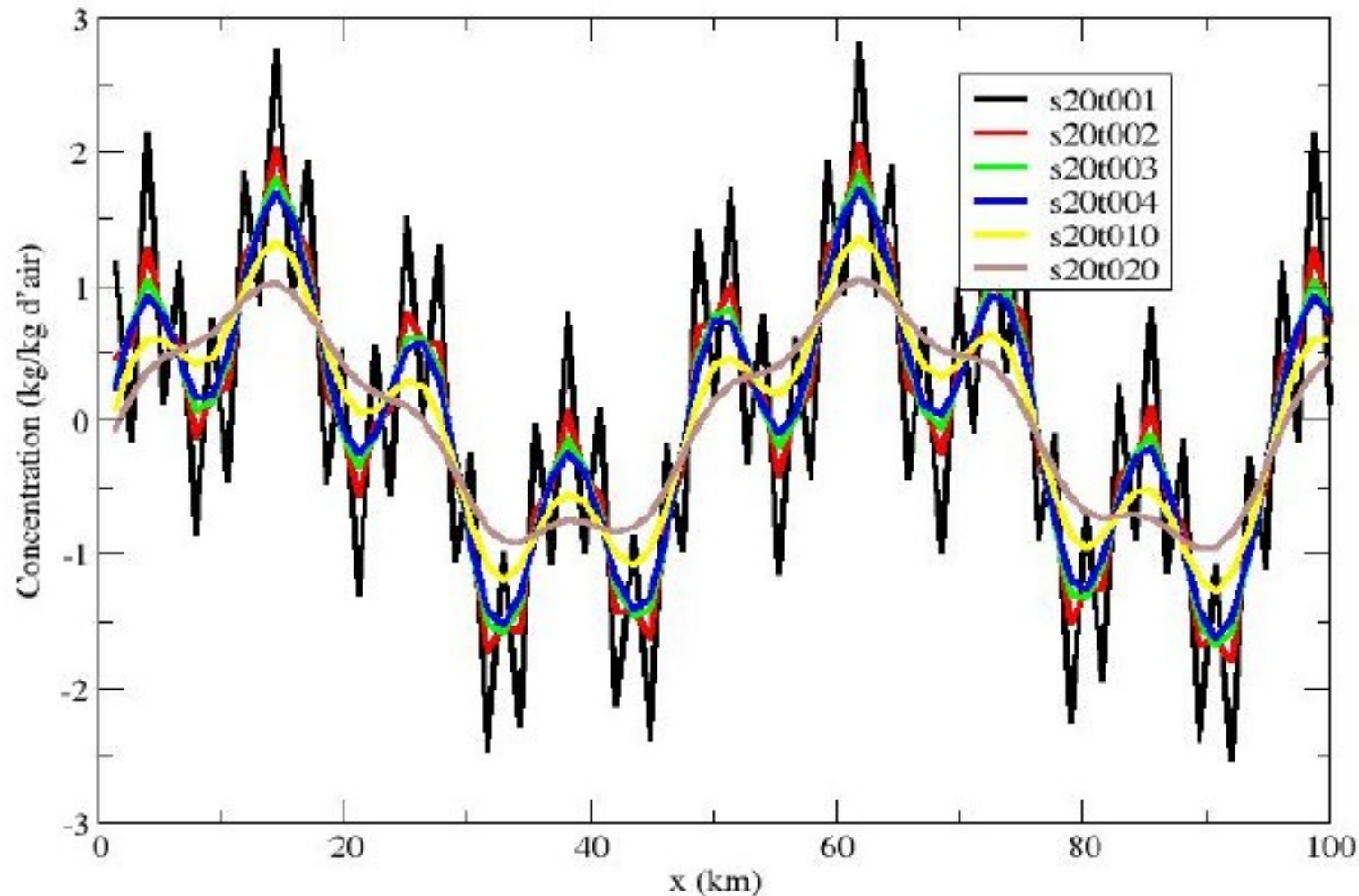
$$A_k(t) = a_k(t) \cdot \sin(kx)$$

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

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

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)

# Some examples of typical values for various resolutions

From test simulations using various horizontal resolutions (Foujols et al.)

[http://forge.ipsl.jussieu.fr/igcmg/wiki/ResolutionIPSLCM4\\_v2](http://forge.ipsl.jussieu.fr/igcmg/wiki/ResolutionIPSLCM4_v2)

Tableau récapitulatif des différences entre résolutions et avec le couplé standard

	<b>R44</b>	<b>R97</b>	<b>R99</b>	<b>R149</b>	<b>R1414</b>	<b>R1914</b>
<b>Résolutions</b>						
atmosphère	44x43x19	96x71x19	96x95x19 (*)	144x96x19	144x142x19	192x142x19 (*)
océan	ORCA2	ORCA2	ORCA2	ORCA2	ORCA2	ORCA2
<b>Etat initial</b>						
océan	2L20 - 20 ans 30 12 1859	2L20 -30 12 1859	2L20 -30 12 1859	2L20 - 30 12 1859	2L20 - 30 12 1859	2L20 - 30 12 1859
atmosphère	10 ans forcé AMIP 1980-1989 <b>F44A</b>	10 ans forcé AMIP 1980-1989 <b>F97A</b>	10 ans forcé AMIP 1980-1989 <b>F99A</b>	10 ans forcé AMIP 1980-1989 <b>F149A</b>	10 ans forcé AMIP 1980-1989 <b>F1414A</b>	10 ans forcé AMIP 1980-1989 <b>F1914A</b>
coupleur	défaut	défaut	défaut	défaut	défaut	défaut
atmosphère (idem avec dynzon)	10 ans forcé AMIP 1980-1989 <b>F44AZ</b>	10 ans forcé AMIP 1980-1989 <b>F97AZ</b>	10 ans forcé AMIP 1980-1989 <b>F99AZ</b>	10 ans forcé AMIP 1980-1989 <b>F149AZ</b>	10 ans forcé AMIP 1980-1989 <b>F1414AZ</b>	10 ans forcé AMIP 1980-1989 <b>F1914AZ</b>
coupleur	défaut	défaut	défaut	défaut	défaut	défaut
<b>Paramètres</b>						
day_step	240	480	480 (=96x71)	720	720	960 (**)
nitergdiv	1					
iphysiq	5	10	10 (=)	15	15	20 (**)
tetagdiv	36000.	5400. (**)	5400. (=)	5400. (**)	3600.	3600. (=144x142)
tetagrot	18000.	7200. (**)	7200. (=)	7200. (**)	5400.	5400. (=)
tetatemp	18000.	7200. (**)	7200. (=)	7200. (**)	5400.	5400. (=)
cvl_corr	1.0	1.0	1.0 (=)	1.0	1.0 (**)	1.0 (=)

# Rules of thumb for run.def parameters

- Time steps in LMDZ:

dynamical time steps:  $dtvr = \text{daysec} / \text{day\_step}$

physics time step:  $dtphys = \text{iphysiq} * dtvr$

dissipation time step:  $dtdiss = \text{dissip\_period} * dtvr$

tracer advection time step:  $dtvrtrac = \text{iapp\_trac} * dtvr$

- Constraints to be aware of:

$dtvr$  limited by CFL for waves:  $C_{\text{max}}.dt < \min(dx, dy)$

$dtrtrac$  limited by advection CFL:  $U_{\text{max}}.dt < \min(dx, dy)$

$iphysiq$ ,  $dtvrtrac$ ,  $dissip\_period$  should be multiples of  $iperiod$



# Rules of thumb for run.def parameters

- Constraints to be aware of (continued):

dissipation time step should be much smaller than dissipation timescales:

$dtdiss \ll tetatgdiv, tetagrot, tetatemp$

- Changing time step with resolution on a regular grid:

$day\_step(\max(iim, jjm)=N) \sim day\_step(\max(iim, jjm)=M) * M/N$

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

$day\_step(\text{zoom}) \sim day\_step(\text{regular}) * \max(\text{grossismx}, \text{grossismy})$