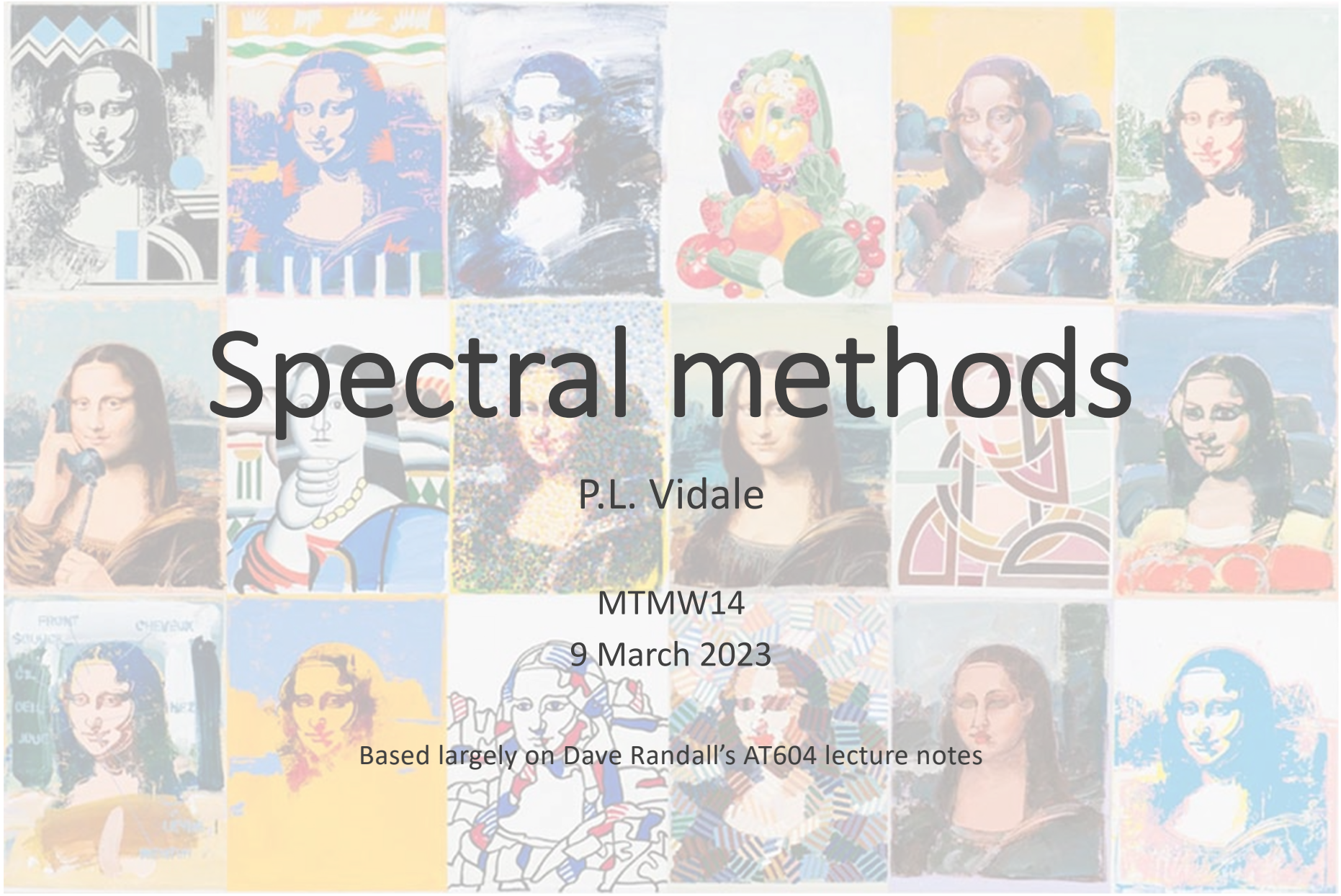


# Spectral methods

P.L. Vidale

MTMW14

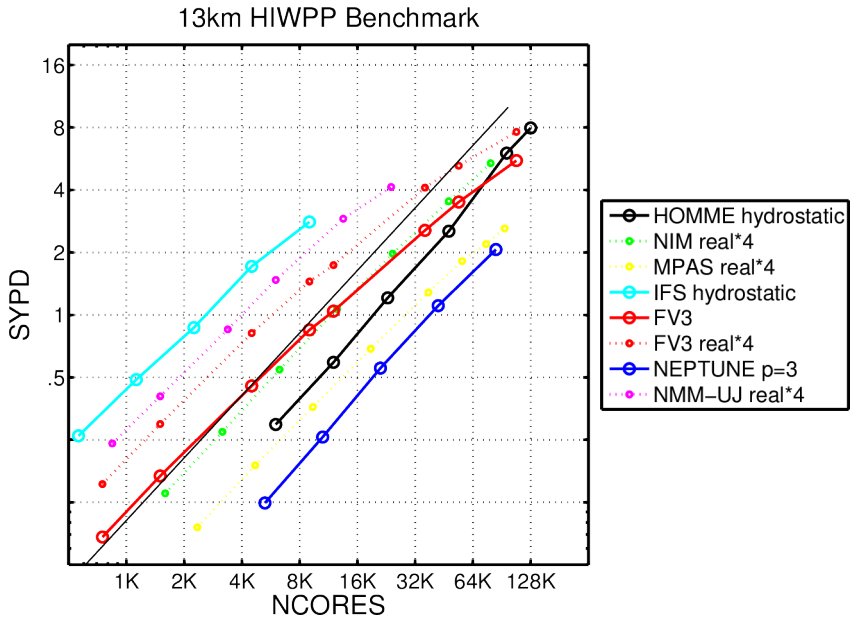
9 March 2023

Based largely on Dave Randall's AT604 lecture notes

# Spectral models: very much alive, despite their death having been announced for 20 years.

## NGGPS Scaling Study

Spectral models are nowadays fast, efficient, **scalable**, and able to perform scale separation by design.




Thomas Schultess (ETH and CSCS) showed how the IFS can take up the entire Piz Daint (one of the top 10 supercomputers) and scale well at 1km

Nils Wedi carried on to Summit…

**http://www.nws.noaa.gov/ost/nggps/dycoretesting.html**

# Recent breakthroughs with the ECMWF IFS

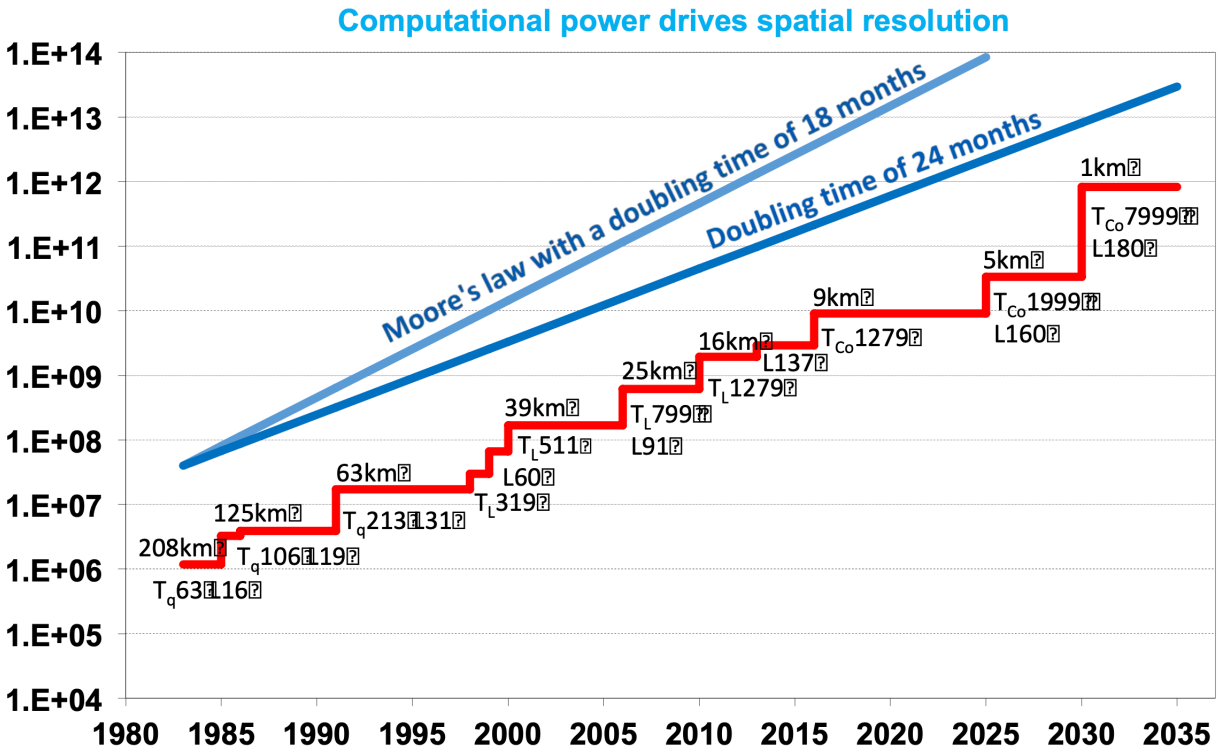



Figure 2: "Understanding grows only logarithmically with the number of floating point operations." (John P. Boyd). The progress in the degrees of freedom (vertical levels x grid columns x prognostic variables) of the ECMWF operational global atmosphere model in comparison to Moore's law, adjusted by ($P^{3/4}$) to account for the change in time-resolution also required in a 3-dimensional dynamical system when increasing spatial resolution. ECMWF's actual progress up to 2018 doubles performance every 24 month. As illustrated, the ambitious goal of reaching a 1 km horizontal resolution with 180 levels in 2030 requires a faster progress. The numbers indicate the average grid-point distance in kilometers and the corresponding spectral resolution and levels used.

## Reflecting on the Goal and Baseline for Exascale Computing: A Roadmap Based on Weather and Climate Simulations


**Thomas C. Schulthess**
ETH Zurich, Swiss National Supercomputing Centre

**Peter Bauer**
European Centre for Medium-Range Weather Forecasts

**Nils Wedi**
European Centre for Medium-Range Weather Forecasts

**Oliver Fuhrer**
MeteoSwiss

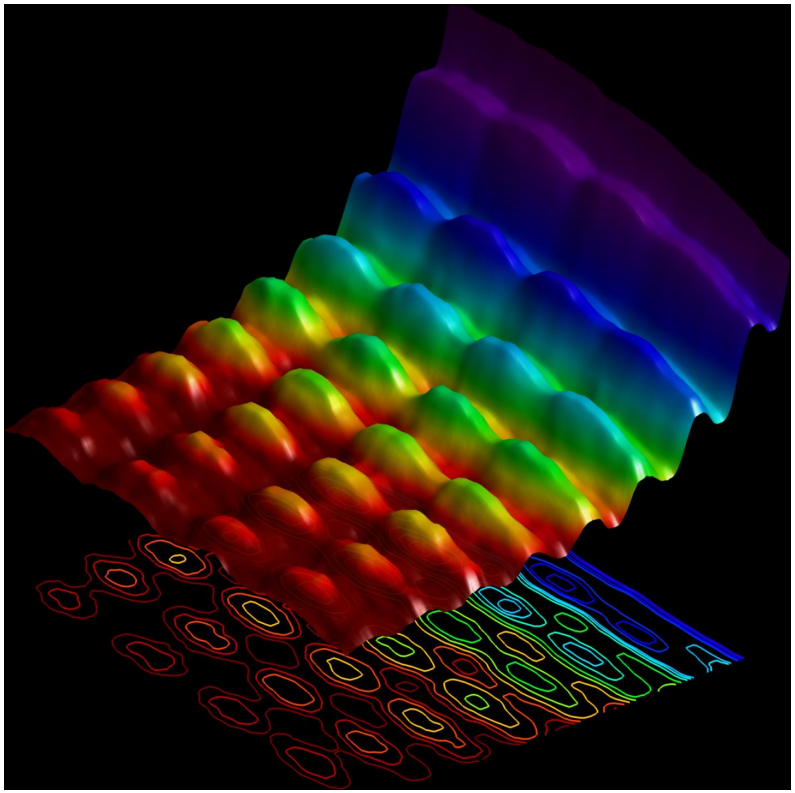
**Torsten Hoefler**
ETH Zurich

**Christoph Schär**
ETH Zurich


**Table 1. Ambitious target configuration for global weather and climate simulations with km-scale horizontal resolution accounting for physical Earth-system processes, and with today's computational throughput rate.**

| | |
|---|---|
| Horizontal resolution | 1 km (globally quasi-uniform) |
| Vertical resolution | 180 levels (surface to ~100 km) |
| Time resolution | 0.5 min |
| Coupled | Land-surface/ocean/ ocean-waves/sea-ice |
| Atmosphere | Non-hydrostatic |
| Precision | Single or mixed precision |
| Compute rate | 1 SYPD (simulated years per wall-clock day) |

# Waves or particles?



This photo was published a few days ago in Science. Fabrizio Carbone (EPFL) was able, for the first time, to capture the dual nature of light.

So far we have been dealing in this course with discrete representations of functions, for instance as points or volumes.

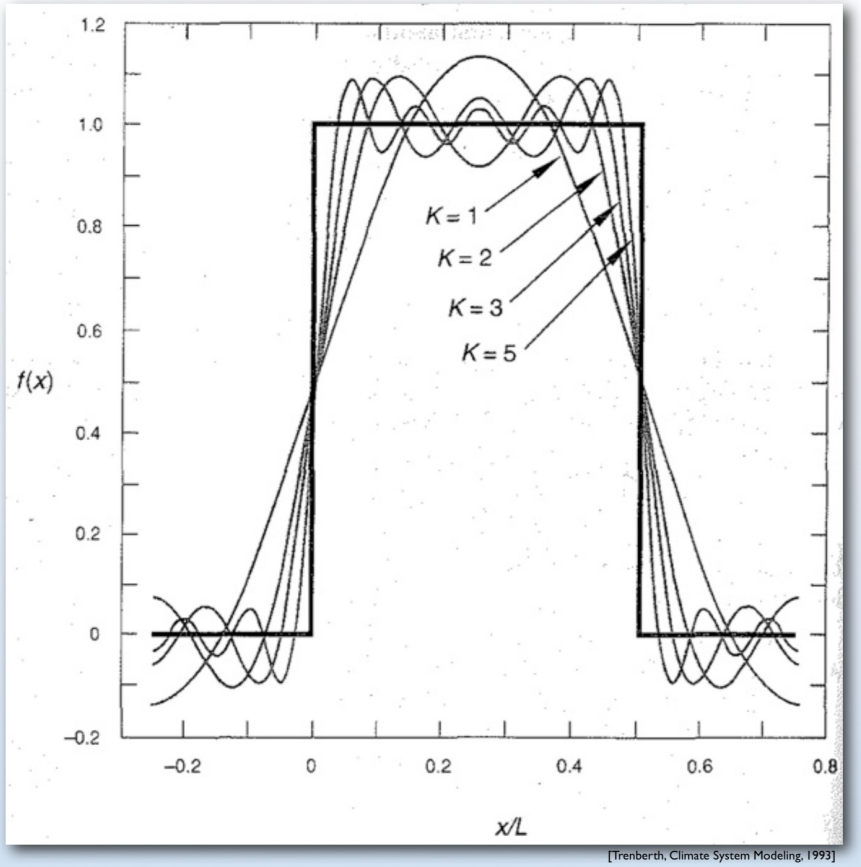
Every continuous function in the function space can be represented as a linear combination of basis functions.

Fourier series are an example of basis functions. From Fourier's theorem, any continuous function can be represented as a sum of *sin* and *cos* functions:

$$q(x,t) = \sum_{k=-\infty}^{\infty} \widehat{q_k}(t) e^{ikx}$$

# Back to our square wave example

Improved Spectral Representation of Steep Gradients by Increasing Wave Numbers




[Trenberth, Climate System Modeling, 1993]

Credit AWI, Germany

# In practice, when using a Fourier basis:

- Start with a real and integrable function *q(x,t)*:
- Use the definition:

$$q(x,t) = \sum_{k=-\infty}^{\infty} \widehat{q_k}(t) e^{ikx} \qquad (1)$$

Representation of q at point location depends on combination of an infinite number of waves

- Where the complex coefficients can be evaluated:

$$\widehat{q_k}(t) = \frac{1}{L} \int_{x-\frac{L}{2}}^{x+\frac{L}{2}} q(x,t) e^{-ikx} dx \qquad (2)$$

Wave characteristics depend on sampling q over a domain of size L

- (1) and (2) form a *transform pair*

# Why bother?

- For starters, gradients are exact and easy. For instance, in physical space (*x* domain):

$$q(x,t) = \sum_{k=-\infty}^{\infty} \widehat{q_k}(t) e^{ikx} \quad (1) \qquad \Rightarrow \qquad \frac{\partial q(x,t)}{\partial x} = \sum_{k=-\infty}^{\infty} ik\, \hat{q}_k(t) e^{ikx}$$

- In practice, for a finite spectral model, the *transform pair* is:

**n** wave numbers at each grid point j

$$q(x_j,t) \cong \sum_{k=-n}^{n} \hat{q}_k(t) e^{ikx_j} \quad (6)$$

$$-n \le k \le n$$

**N is the TRUNCATION**

**M** grid points used to find spectral coefficients

$$\hat{q}_k(t) \cong \frac{1}{M} \sum_{j=1}^{M} q(x_j,t)\, e^{-ikx_j} \quad (7)$$

**M is the RESOLUTION**

$$\frac{\partial q(x_j,t)}{\partial x} \cong \sum_{k=-n}^{n} ik\hat{q}_k(t) e^{ikx_j}$$

Gradient computed using waves

# Truncation, accuracy and computational costs

- The truncation is exactly reversible if M is large. This is equivalent to model resolution in finite difference models
- What are the costs involved? Substitute (6) into (7)

$$\hat{q}_k(t) = \frac{1}{M} \sum_{j=1}^{M} \left\{ \left[ \sum_{l=-n}^{n} \hat{q}_l(t)\, e^{ilx_j} \right] e^{-ikx_j} \right\}, \quad -n \le k \le n$$

$$-n \le l \le n$$

- Both wave numbers bound by n. Max wave number: $\pm 2n$
  → We need 2n+1 complex conjugates
- The Fourier representation is equivalent to discretising *q(x,t)* onto **2n+1 grid points**

# Waves and particles again

We have:

- Truncation n → max wave number: 2n
- 2n+1 complex conjugates $\hat{q}_k(t)$

This means that for a grid of M points to represent amplitudes and phases of all waves up to $k = \pm n,$ it is necessary that $M \geq 2n+1$

**The truncation in the representation of the waves and the grid resolution are related.** This is important, because,

1. as we will see, physical parameterisations operate on the grid, not in spectral space. The model algorithm uses the transform pair to jump from the grid into spectral space and back again.
2. There are non-linear terms…

# Equivalence of spectral and finite difference methods

$$\frac{\partial q}{\partial x}(x_l,t) \cong \sum_{k=-n}^{n}\left[\frac{ik}{M}\sum_{j=1}^{M} q(x_j,t)\; e^{-kx_j}\right] e^{ikx_l}$$

Now reverse the two $\sum$

$$\frac{\partial q}{\partial x}(x_l,t) \cong \sum_{j=1}^{M} \alpha_j^l \; q(x_j,t)$$

where $\alpha_j^l \equiv \frac{i}{M}\sum_{k=-n}^{n} k \;\; e^{ik\left(x_l - x_j\right)}$

This can be **interpreted as a finite difference approximation**.

A member of the families we have seen before, but now it involves all points in the domain

# Let's try to build a simple model:

## 1D linear advection in spectral space

$$\frac{\partial q}{\partial t} + c\frac{\partial q}{\partial x} = 0 \qquad \sum_{k=-n}^{n} \frac{d\hat{q}_k}{dt} e^{ikx_j} + c\sum_{k=-n}^{n} ik\hat{q}_k e^{ikx_j} = 0$$

Since we can invoke linear independence:

$$\frac{d\hat{q}_k}{dt} + ikc\hat{q}_k = 0 \quad \text{for all} \quad -n \le k \le n$$

**We have our prognostic model!!**

remember that $\frac{dq_0}{dt} \equiv 0$

# 1D linear advection: spectral and finite difference

$$\frac{d\hat{q}_k}{dt} + ikc\hat{q}_k = 0$$

**SPECTRAL**

$$\frac{d\hat{q}_k}{dt} + ikc\left[\frac{\sin(k\Delta x)}{k\Delta x}\right]\hat{q}_k = 0$$

**FINITE DIFFERENCE**

Which model will give better results?

# More advantages for spectral models: a boundary value problem

$$\nabla^2 q = f(x,y) \qquad \text{in 1D: } \frac{d^2 q}{dx^2} = f(x)$$

with periodic B.C.s:

$$\sum_{k=-n}^{n} (-k^2)\hat{q}_k e^{ikx} = \sum_{k=-n}^{n} \hat{f}_k e^{ikx}$$

$$\Rightarrow \hat{q}_k = -\frac{f}{k^2} \qquad \text{for } -n \le k \le n, \text{ not valid for } k = 0$$

This is an exact solution for all modes that are included.

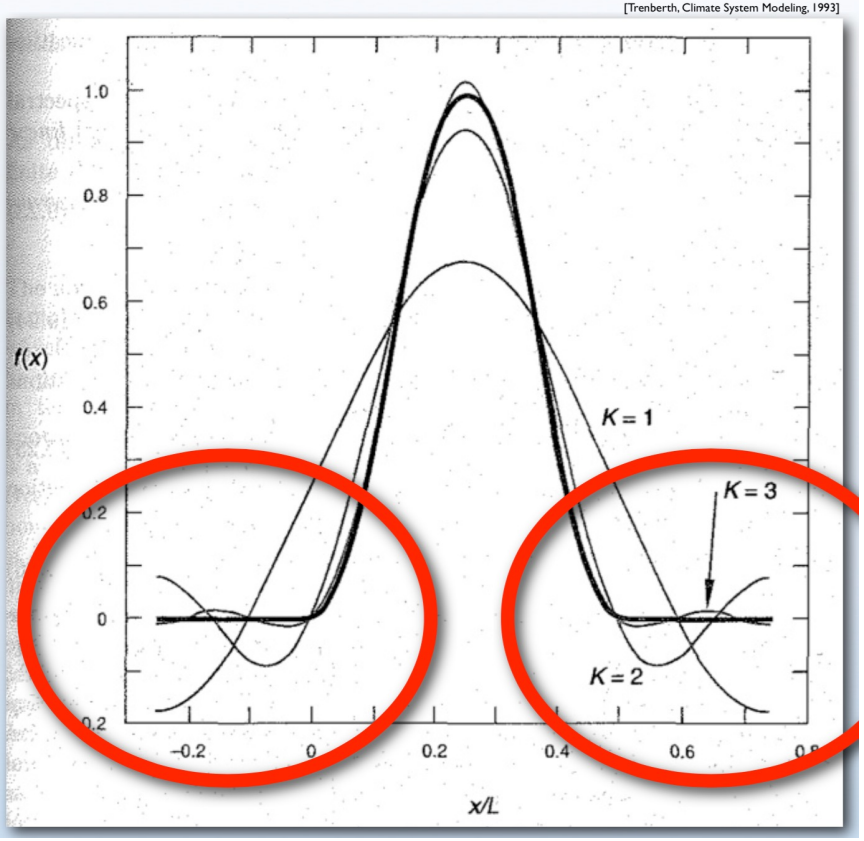
Ends up not being exact because some modes are left out.

# Disadvantages I: treatment of orography
# This is more typical of older spectral models

## "Overshooting" of a Spectral Representation of Steep Gradients

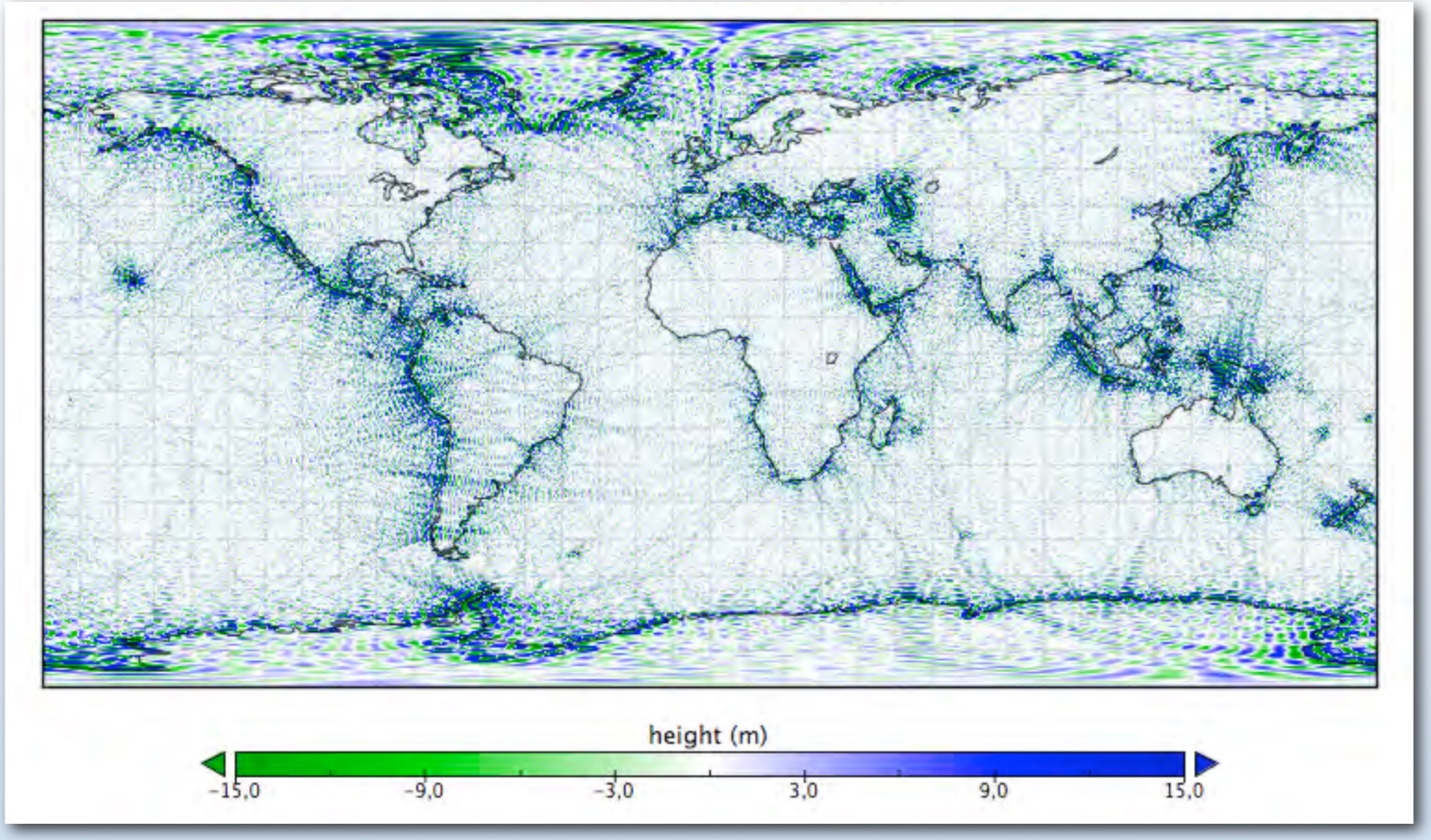
We said that we can represent anything, e.g. a square wave, but "overshooting" is one of the many prices we pay when we use spectral methods.

[Trenberth, Climate System Modeling, 1993]




Credit AWI, Germany

# Example: "Overshooting" of Topography Gradients after Transformation into Spectral Space



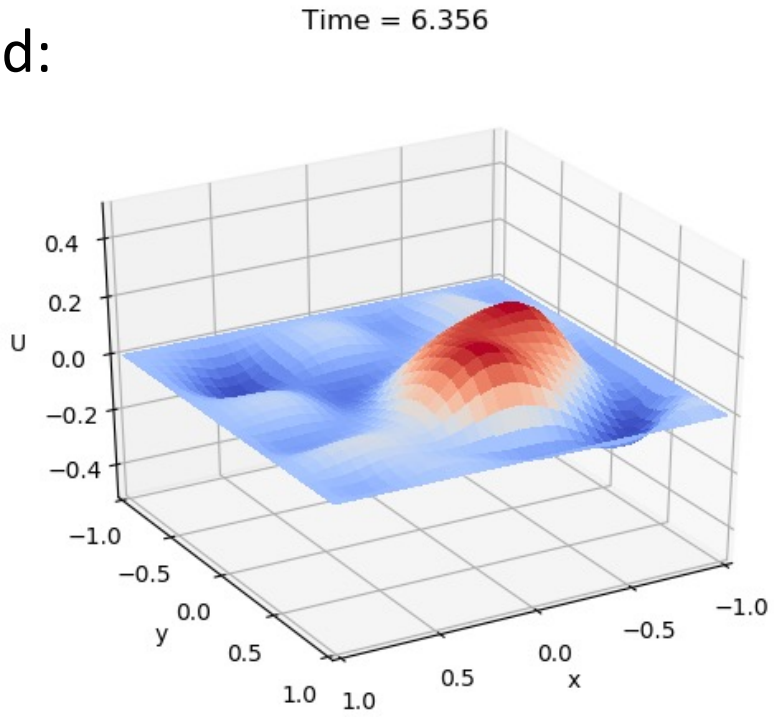

Orography Difference: Original - Spectral Transformation

Credit AWI, Germany

# Brief interval: SWE re-visited with finite differences vs spectral method

For a reasonably smooth initial condition, I used:

| | Finite Difference | Spectral | Spectral |
|---|---|---|---|
| Resolution | M=50 grid points in x,y | T=12 wavenumbers | T=24 |
| Time step | 0.2s | 0.2s | 0.2s |
| Time to solution | 30s | 15s | 59s |
| | | But the spectral method converges quickly with low wavenumber, so T=24 is not needed for this problem!!!<br>Also, I am not using any of the typical spectral tricks… | |




From python Spectral_wave2_32.py

# A bit more trouble when waves interact…

- Let us go back to advection, but this time let us consider the full (non-linear) problem of momentum advection over a periodic domain:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \quad \Rightarrow \quad \sum_{k=-n}^{n} \frac{d\hat{u}_k}{dt} e^{ikx} = -\left( \sum_{l=-n}^{n} \hat{u}_l e^{ilx} \right) \left( \sum_{m=-n}^{n} im\hat{u}_m e^{imx} \right)$$

- But now we have products in this form: $e^{ilx} e^{imx}$
  - Both l and m are in the range –n … n. **Meaning that?**
- In fact, for a given Fourier mode *k*:

$$\frac{d\hat{u}_k}{dt} = -\sum_{l=-\alpha}^{\alpha} \sum_{m=-\alpha}^{\alpha} im\left[\hat{u}_l \hat{u}_m e^{i(l+m)x}\right] e^{-ikx}, \text{ for } -n \le k \le n$$

# Computing individual terms in spectral space is computationally expensive, even in 1D

- α must be large enough to cover all possible combinations of *l,m* inside the range *–n,n*
- The number of points inside each triangular (e**X**cluded) region is:

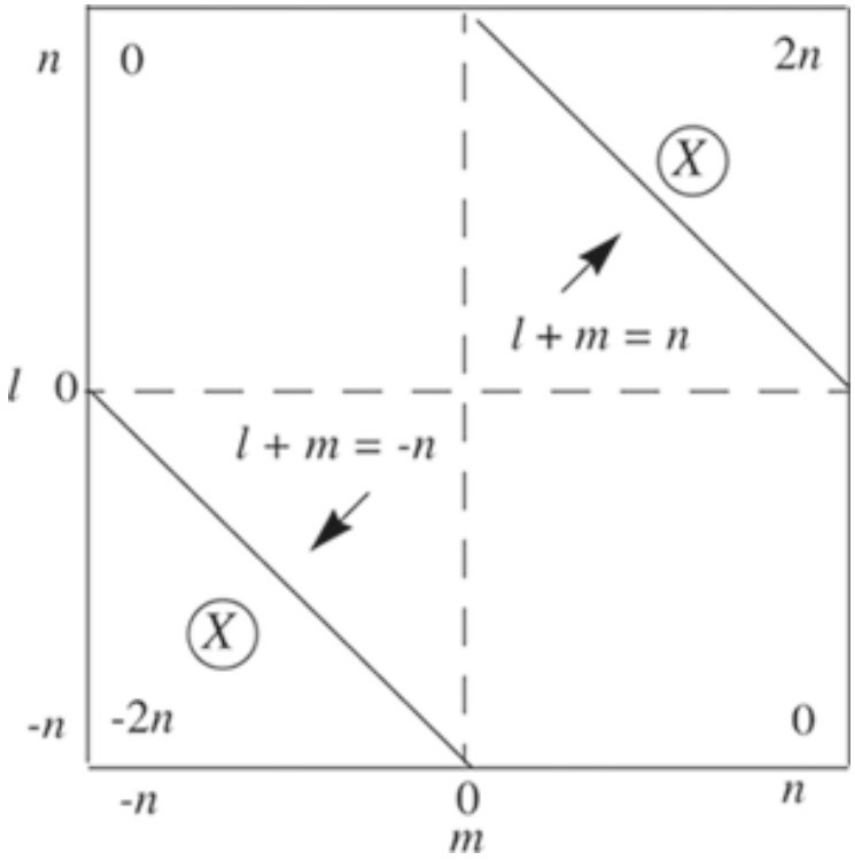
$$1+2+3\ldots+(n-1)=\frac{n(n-1)}{2}$$

- The number of points *retained* is therefore:

$$(2n+1)^2-2\left[\frac{n(n-1)}{2}\right]=3n^2+5n+1$$

Which is the number of points that must be evaluated.

- This is $O(n^2)$ and it is very bad news, also considering that it originated with just a quadratic term. Computational cost will increase very rapidly. This is much worse in 2D.
- What can we do?



# Combining grid and spectral methods

- What if we evaluated $u$ and $\frac{\partial u}{\partial x}$ on a grid? If we had a sufficiently large grid, no aliasing!

$$\left(u\frac{\partial u}{\partial x}\right)_k = \frac{1}{M}\sum_{j=1}^{M}\left[u(x_j)\frac{\partial u}{\partial x}(x_j)e^{-ikx_j}\right], \quad -n \le k \le n$$

**SPECTRAL** **GRID**

- Now use definitions to write terms as Fourier series (and benefit from exact derivatives)

$$\left(u\frac{\partial u}{\partial x}\right)_k = \frac{1}{M}\sum_{j=1}^{M}\left[\left(\sum_{l=-n}^{n}\hat{u}_l e^{ilx_j}\right)\left(\sum_{m=-n}^{n} im\hat{u}_m e^{imx_j}\right)e^{-ikx_j}\right], \quad -n \le k \le n$$

**SPECTRAL** **GRID, using SPECTRAL METHOD**

# How many grid points do we need for the non-linear momentum advection?

- We have 3 Fourier modes, *k,l,m,* so zonal wave numbers up to *± 3n*. Therefore we need *3n+1* complex coefficients.
- But $u\frac{\partial u}{\partial x}$ is real, so we only have *3n+1* independent real numbers.
  - Therefore *M≥3n+1*
- 50% more expensive than the linear case, but not as bad as $n^2$
- Note that the extra 50% of information on the grid is thrown away when we transform back (5)

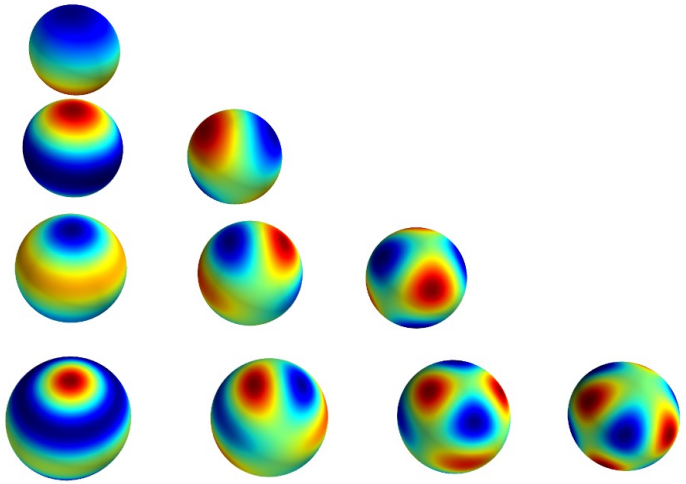
**Our spectral model so far**:

1. Initialise spectral coefficients $\hat{u}_k$ for $-n \le k \le n$
1. Evaluate $u$ and $\frac{\partial u}{\partial x}$ on a grid with M points.

   The derivative is computed using the spectral method
2. Form the product $u\frac{\partial u}{\partial x}$ on the grid
3. Now transform back into wave-number space, so as to obtain coefficients: $\widehat{\left(u\frac{\partial u}{\partial x}\right)}_k$

5. Predict (do the time step) $\frac{d\hat{u}_k}{dt} = -\widehat{\left(u\frac{\partial u}{\partial x}\right)}_k$
6. Go back to step 2.

# Spectral methods on the sphere



- First developed by Silberman (1954)

$$F(\lambda,\varphi) = \sum_{m=-\infty}^{\infty} \sum_{n=|m|}^{\infty} F_n^m Y_n^m(\lambda,\varphi),$$

$\lambda$ is lon; $\varphi$ is lat; $m$ is the zonal wavenumber

where:

$$Y_n^m(\lambda,\varphi) = e^{im\lambda} P_n^m(\sin\varphi)$$

are spherical harmonics and $P_n^m(\sin\varphi)$ are the associated Legendre functions of the first kind

The spherical harmonics $Y_n^m$ are eigenfunctions of the Laplacian on the sphere:

$$\nabla^2 Y_n^m = \frac{-n(n+1)}{a^2} Y_n^m$$ where $a$ is the radius of the sphere

# In practice: spectral models on the sphere

An approximation to F can be made by using truncated sums:

$$\bar{F} = \sum_{m=-M}^{M} \sum_{n=|m|}^{N(m)} F_n^m Y_n^m$$

$m$ is the zonal wavenumber

how to choose $N(m)$?

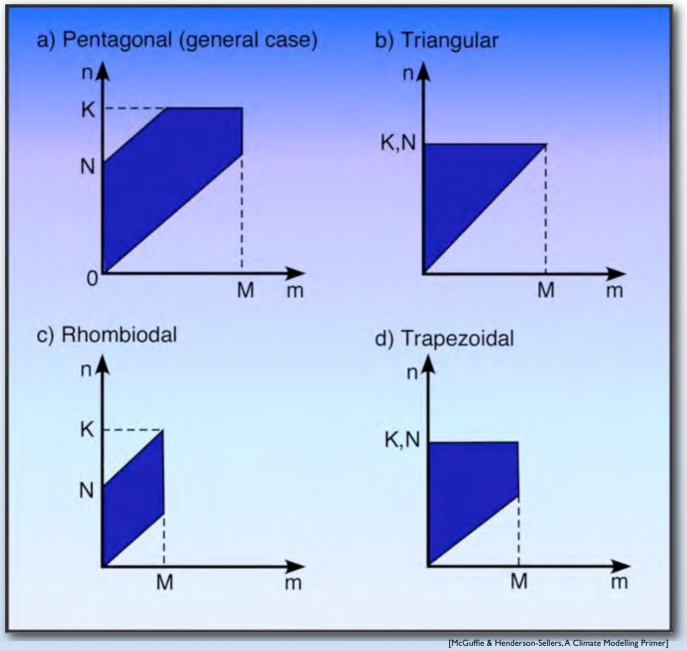
This is known as the problem of *truncation.*

Two well-known possibilities are:

*rhomboidal truncation* ***N=M+|m|*** and

*triangular truncation* ***N=M***

Different Truncation Types used in spectral AGCM



# Spherical Harmonic Transform

- Similar to what we did with Fourier transforms, we can design a spherical harmonic transform, which consists in the combination of a Fourier transform and a Legendre transform.
- The Legendre transform is formulated using a method called *Gaussian quadrature*.
  - Say that we want to evaluate: $I \equiv \int_{-1}^{1} f(x)\,dx$ numerically
  - If *f(x)* is defined at a finite number of points $x_j$, then $I \cong \sum_{i=1}^{N} f(x_i) w_i, \quad w_i$ are weights (44)
- Special case: *f(x)* is a weighted sum of Legendre polynomials, as in a Legendre transform. We want the transform to be exact.
- Gauss showed that (44) gives the exact value of *I* if we choose the points $x_j$ to be the roots of the highest Legendre polynomial used.
- **In practice, we choose (Gaussian) latitudes to be those roots.** Expensive (iterative), but we only need to perform this calculation once.

# Spherical grids

- For either truncation, rhomboidal or triangular, choosing M fixes the expansion. That is then how we can say R15 or T106. The number of coefficients is:
  - Rhomboidal: $(M+1)^2 + M^2 + M$
  - Triangular: $(M+1)^2$
- So, the transform suitable for a spherical geometry needs many more points than the spectral transform. For triangular truncation:
  - Grid points around a latitude circle: $\geq 3M+1$
  - Grid points on a meridian: $\geq (3M+1)/2$
  - **TOTAL**: $\geq (3M+1)^2/2$ a factor of 2.25 between grid and spectral representations.
- Computing physics on the non-aliasing grid is quite wasteful, but unavoidable.
- Until recently we did not have fast Legendre transforms, but ECMWF have made major breakthroughs in this area. Without FLTs, operation count would be $O(N^3)$, while finite difference methods are $O(N^2)$.

# Summary of spectral methods

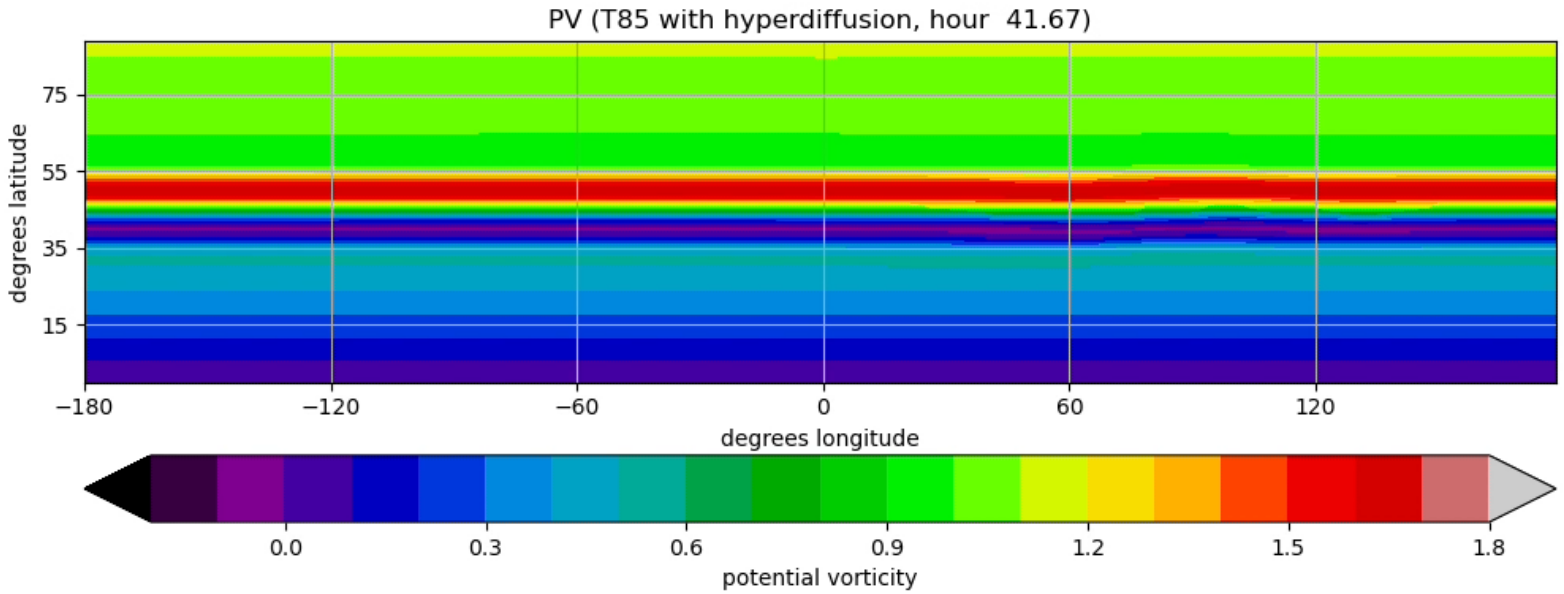
**Strengths**

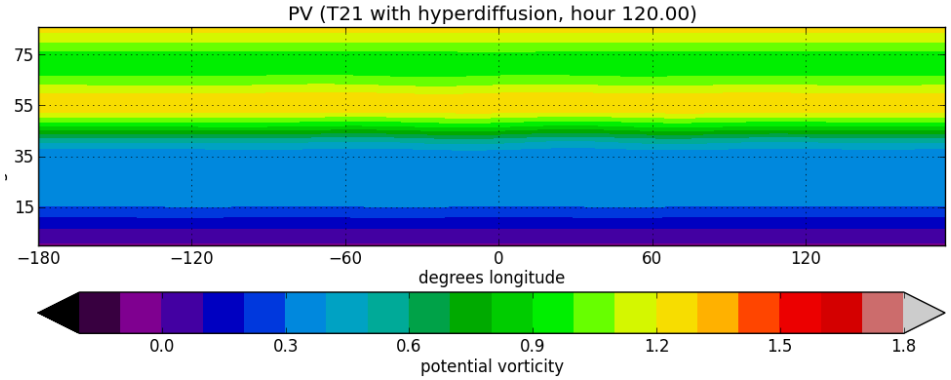
- Eliminate pole problem, especially with triangular truncation
- Provides exact phase speed for linear waves and advection by constant current
- **Converges very rapidly and gives good results even with just few modes**
- Semi-implicit time stepping is implemented very easily

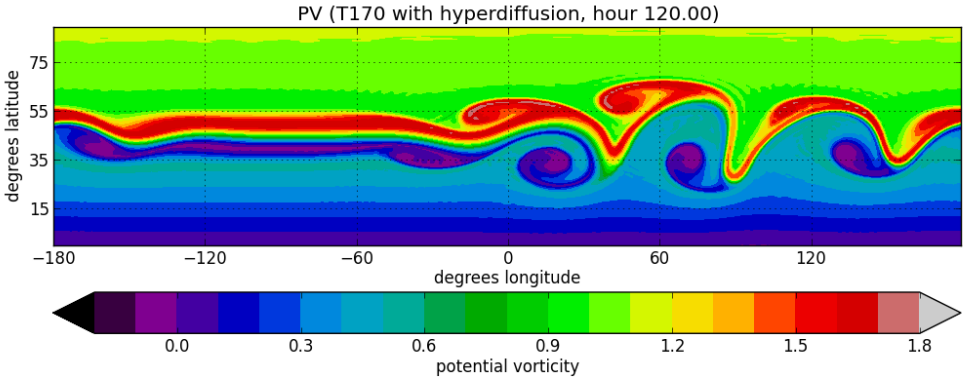
**Weaknesses**

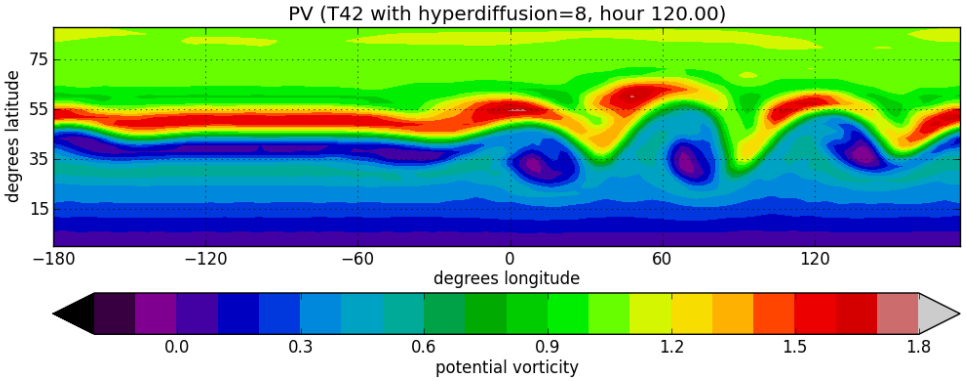
- Do not conserve anything
- Failure to conserve requires damping, else BOOM
- “Bumpy” oceans
- Truncation in transform may mean that physics do not always have the intended effects
- Moisture advection problematic
- Until FLT, not practical at high-resolution

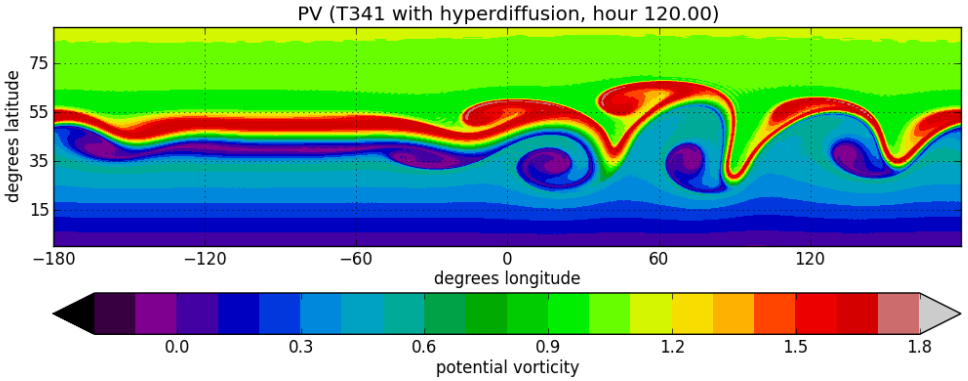
# Galewsky's barotropic instability problem with a global spectral model (T85)

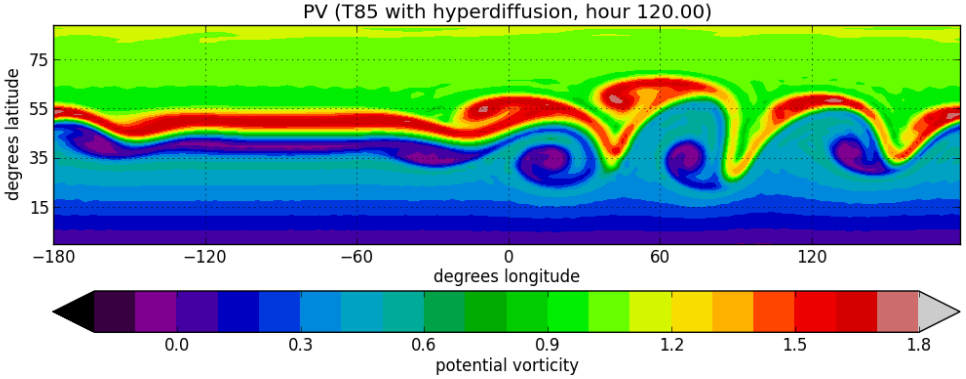





The Galewski problem on truncations T21 to T341 at time 120 hours.

T42 already captures the main characteristics of the flow.

*Results from a global, spectral SWE model that was written in Python and can be run on a laptop*

# A spectral SWE model

## 1) Setup grid and initial conditions

```
    # setup up spherical harmonic instance, set lats/lons of grid
    x = Spharmt(nlons, nlats, ntrunc, rsphere, gridtype="gaussian")
#    x = Spharmt(nlons, nlats, ntrunc, rsphere, gridtype="regular")
    lons, lats = np.meshgrid(x.lons, x.lats)
    f = 2.*omega*np.sin(lats)   # coriolis

    # zonal jet.
    vg = np.zeros((nlats, nlons), np.float)
    u1 = (umax/en)*np.exp(1./((x.lats-phi0)*(x.lats-phi1)))
    ug = np.zeros((nlats), np.float)
    ug = np.where(np.logical_and(x.lats < phi1, x.lats > phi0), u1, ug)
    ug.shape = (nlats, 1)
    ug = ug*np.ones((nlats, nlons), dtype=np.float)     # broadcast to shape (nlats, nlonss)
    # height perturbation.
    hbump = hamp*np.cos(lats)*np.exp(-((lons-np.pi)/alpha)**2)*np.exp(-(phi2-lats)**2/beta)

    # initial vorticity, divergence in spectral space
    vrtspec, divspec = x.getvrtdivspec(ug, vg)
    vrtg = x.spectogrd(vrtspec)
    divg = x.spectogrd(divspec)

    # create hyperdiffusion factor
    hyperdiff_fact = np.exp((-dt/efold)*(x.lap/x.lap[-1])**(ndiss/2))

    # solve nonlinear balance eqn to get initial zonal geopotential,
    # add localized bump (not balanced).
    vrtg = x.spectogrd(vrtspec)
    tmpg1 = ug*(vrtg+f)
    tmpg2 = vg*(vrtg+f)
    tmpspec1, tmpspec2 = x.getvrtdivspec(tmpg1, tmpg2)
    tmpspec2 = x.grdtospec(0.5*(ug**2+vg**2))
    phispec = x.invlap*tmpspec1 - tmpspec2
    phig = grav*(hbar + hbump) + x.spectogrd(phispec)
    phispec = x.grdtospec(phig)

    # initialize spectral tendency arrays
    ddivdtspec = np.zeros(vrtspec.shape+(3,), np.complex)
    dvrtdtspec = np.zeros(vrtspec.shape+(3,), np.complex)
    dphidtspec = np.zeros(vrtspec.shape+(3,), np.complex)
    nnew = 0
    nnow = 1
    nold = 2
```

## 2) Time integration loop

Bring prognostics to grid

Tendencies on grid

Tendencies in spectral space

Adams-Bashfort Time step

Physics in spectral space

Update

```
    # time loop.
    time1 = time.time()
    for ncycle in range(itmax+1):
        t = ncycle*dt
        # get vort, u, v, phi on grid
        vrtg = x.spectogrd(vrtspec)
        ug, vg = x.getuv(vrtspec, divspec)
        phig = x.spectogrd(phispec)
        print("t=%6.2f hours: min/max %6.2f, %6.2f" % (t/3600., vg.min(), vg.max()))

        # compute tendencies.
        tmpg1 = ug*(vrtg+f)
        tmpg2 = vg*(vrtg+f)
        ddivdtspec[:, nnew], dvrtdtspec[:, nnew] = x.getvrtdivspec(tmpg1, tmpg2)
        dvrtdtspec[:, nnew] *= -1

        tmpg = x.spectogrd(ddivdtspec[:, nnew])
        tmpg1 = ug*phig
        tmpg2 = vg*phig
        tmpspec, dphidtspec[:, nnew] = x.getvrtdivspec(tmpg1, tmpg2)
        dphidtspec[:, nnew] *= -1

        tmpspec = x.grdtospec(phig+0.5*(ug**2+vg**2))
        ddivdtspec[:, nnew] += -x.lap*tmpspec

        # update vort, div, phiv with third-order adams-bashforth.
        # forward euler, then 2nd-order adams-bashforth time steps to start.
        if ncycle == 0:
            dvrtdtspec[:, nnow] = dvrtdtspec[:, nnew]
            dvrtdtspec[:, nold] = dvrtdtspec[:, nnew]
            ddivdtspec[:, nnow] = ddivdtspec[:, nnew]
            ddivdtspec[:, nold] = ddivdtspec[:, nnew]
            dphidtspec[:, nnow] = dphidtspec[:, nnew]
            dphidtspec[:, nold] = dphidtspec[:, nnew]
        elif ncycle == 1:
            dvrtdtspec[:, nold] = dvrtdtspec[:, nnew]
            ddivdtspec[:, nold] = ddivdtspec[:, nnew]
            dphidtspec[:, nold] = dphidtspec[:, nnew]

        vrtspec += dt*(
            (23./12.)*dvrtdtspec[:, nnew] - (16./12.)*dvrtdtspec[:, nnow]
            + (5./12.)*dvrtdtspec[:, nold])
        divspec += dt*(
            (23./12.)*ddivdtspec[:, nnew] - (16./12.)*ddivdtspec[:, nnow]
            + (5./12.)*ddivdtspec[:, nold])
        phispec += dt*(
            (23./12.)*dphidtspec[:, nnew] - (16./12.)*dphidtspec[:, nnow]
            + (5./12.)*dphidtspec[:, nold])
        # implicit hyperdiffusion for vort and div.
        vrtspec *= hyperdiff_fact
        divspec *= hyperdiff_fact
        # switch indices, do next time step.
        nsav1 = nnew
        nsav2 = nnow
        nnew = nold
        nnow = nsav1
        nold = nsav2
```

# Equivalent grid resolution of spectral models

We will see later in the lecture that, in 1D, we need at least 2N+1 grid points to represent the waves used in a truncation with N wavenumbers.

The real resolution needs 1D lie between 2N+1 and N^2, as also seen in the paper by Renee Laprise (1992).

However, what is the real, effective resolution of (spectral) models? We have a paper under review (Klaver et al. 2019) to answer that question.

Laprise: 4 ways to compute the resolution of a spectral model. Let us take T31 truncation

1: average distance between latitudes on Gaussian grid L1 = 425km

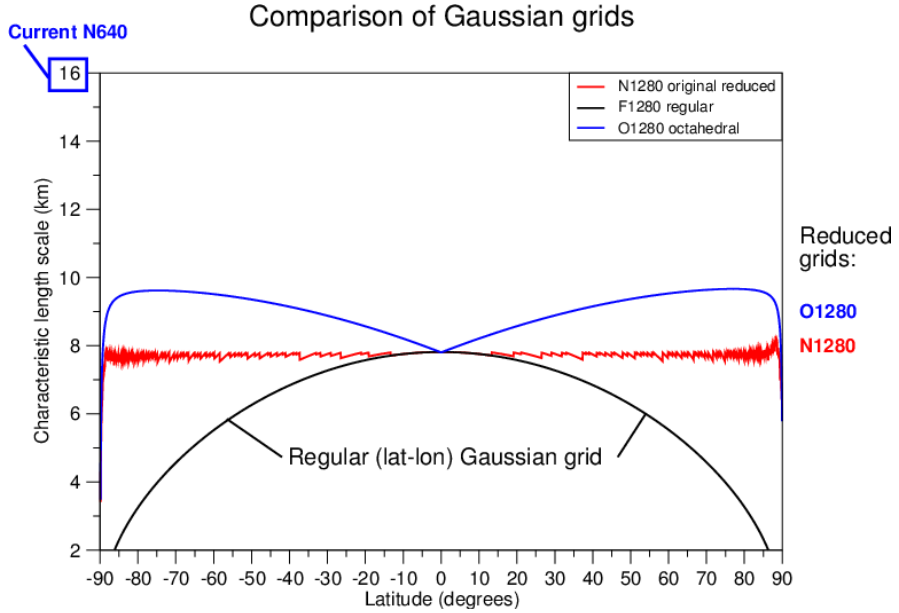
2: half the wavelength of the shortest resolved zonal wave at the equator: L2 = 650km

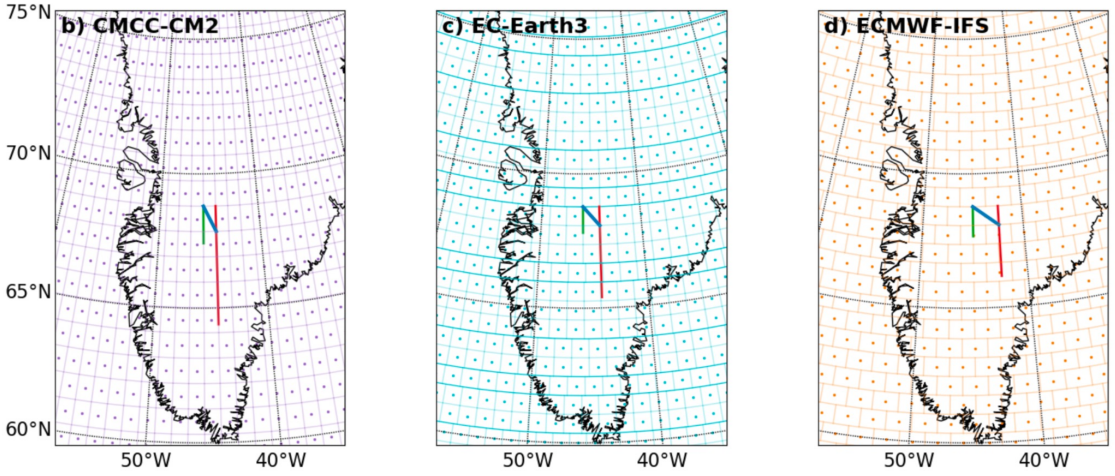
3: Number of real variables per unit area: L3 = 725km

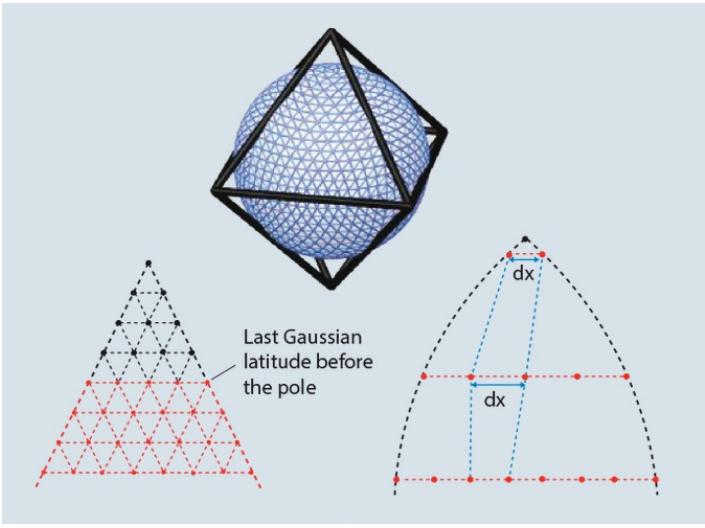
4: equivalent total wave number associated with the Laplacian operator (for highest mode): L4 = 900km

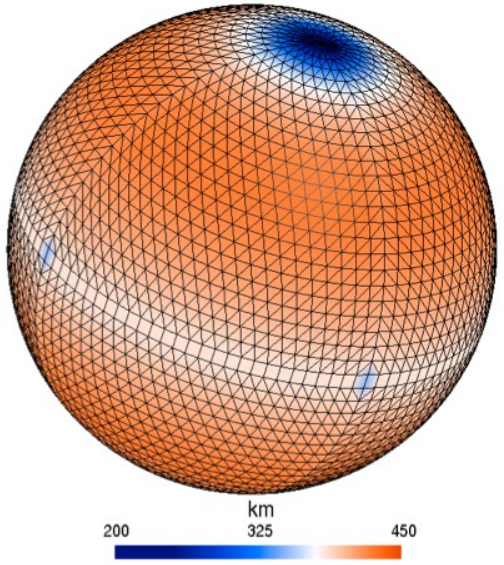
*For reference, in lat/lon models, N31 = 425km*

# Associated grids in physical space



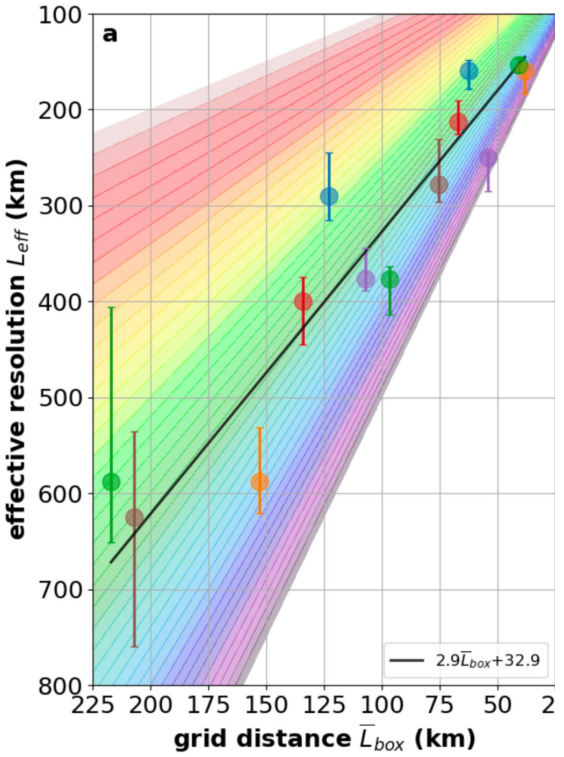




N24 octahedral Gaussian grid

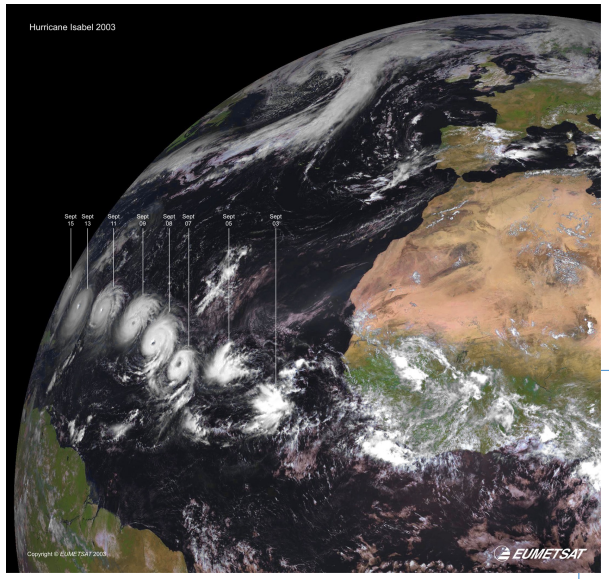
# Effective resolution of PRIMAVERA models

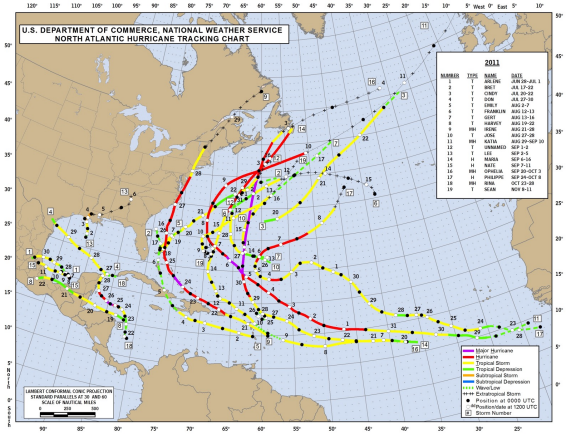
| Model-Version | Grid Type | Native Grid | $\overline{L}_{box}$ (km) | Resolution $L_{eff}$ (km) | $L_{eff}/\overline{L}_{box}$ |
|---|---|---|---|---|---|
| HadGEM3-GC31-LM | grid point | 145 x 192 | 217 | 525-770 (588) | 2.4-3.5 (2.7) |
| HadGEM3-GC31-MM | grid point | 325 x 432 | 96.7 | 340-390 (377) | 3.5-4.1 (3.9) |
| HadGEM3-GC31-HM | grid point | 769 x 1024 | 40.8 | 145-160 (153) | 3.6-3.9 (3.7) |
| CMCC-CM2-HR4 | grid point | 192 x 288 | 153 | 555-645 (588) | 3.6-4.2 (3.8) |
| CMCC-CM2-VHR4 | grid point | 768 x 1152 | 38.2 | 135-165 (159) | 3.5-4.4 (4.2) |
| ECMWF-IFS-LR | spectral | reduced TCO255 | 123 | 265-335 (290) | 2.2-2.7 (2.4) |
| ECMWF-IFS-HR | spectral | reduced TCO511 | 62.6 | 140-170 (159) | 2.2-2.7 (2.5) |
| EC-Earth3 | spectral | reduced TL255 | 107 | 365-410 (377) | 4.0-4.7 (4.6) |
| EC-Earth3-HR | spectral | reduced TL511 | 54.2 | 215-255 (250) | 3.4-3.8 (3.5) |
| MPIESM-1-2-HR | spectral | TQ127 | 134 | 355-425 (400) | 2.7-3.2 (3.0) |
| MPIESM-1-2-XR | spectral | TQ255 | 66.9 | 200-235 (213) | 3.0-3.5 (3.2) |
| CNRM-CM6-0 | spectral | reduced TL127 | 207 | 490-715 (625) | 2.4-3.5 (3.0) |
| CNRM-CM6-0-HR | spectral | reduced TL359 | 75.3 | 260-325 (278) | 3.4-4.3 (3.7) |




Klaver et al. *Atmospheric Science Letters* 2020 (https://doi.org/10.1002/asl.952)

# Finding the storms



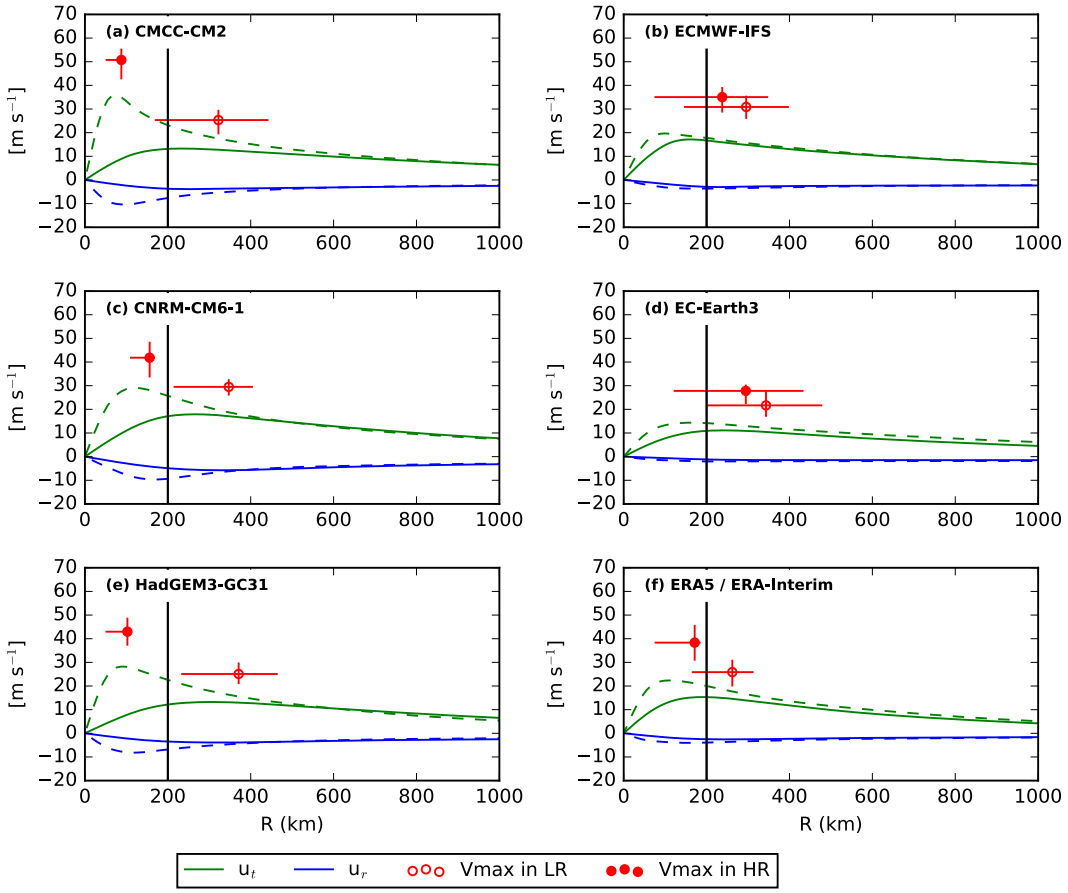

## TRACK

### Configurations

| Tropical Cyclones and Easterly Waves | Extra-Tropical Cyclones (Synoptic scale) | Mesocyclones |
|---|---|---|
| 6 hourly data. | 6 or 3 hourly data. | ideally 3 hourly data but can use 6 hourly. |
| spectral filtering, T42 for single level vorticity; T63 for vertically averaged vorticity. | spectral filtering, choice can depend on field, T42 for vorticity (any level), T63-T106 for MSLP or Z. | spectral filtering T40-100 for vorticity, T40-200 for MSLP. |
| Planetary wave filtering. | Planetary wave filtering. | intensity threshold ~2.0^10^(-5) |
| intensity threshold, usually set low, ~0.5x10^(-5). | intensity threshold, usually set low, ~1.0^10^(-5) for vorticity; 1hPa for MSLP. | for extra-tropics, identification on polar stereographic projection. |
| cylindrical projection (60S, 60N). | projection for identification, usually polar stereographic but can choose. | tracking parameters, displacement distances, track smoothness, applied adaptively. |
| tracking parameters, displacement distances, track smoothness, applied adaptively. | tracking parameters, displacement distances, track smoothness, applied adaptively. | |
| post tracking filtering, 2 days. | post tracking filtering, 2 day, 1000km but can be modified by user. | |
| **warm core test**, add vorticity at T63 for multiple levels up to 200hPa. | | |

# Extra slides

# Simulation of the primary and secondary circulations of hurricanes




PRIMAVERA GCMs: composites of Tropical Cyclone structures at high and low resolution

The inner core region is better simulated at high resolution. This is clearer in finite difference and finite volume models (left column, top and bottom) than it is in spectral models (right column and left-centre, CNRM