The Structure of the FV^3 Solver

Lucas Harris
and the GFDL FV^3 Team

FV3 Workshop in Taiwan

4 December 2017

FV^3 Design Philosophy

- Discretization should be guided by physical principles as much as possible
  - Finite-volume, integrated form of conservation laws
  - Upstream-biased fluxes
- Operators “reverse engineered” to achieve desired properties
- Computational efficiency is **crucial**. Fast models can be good models!
- Solver should be built with vectorization and parallelism in mind
- Dynamics isn’t the whole story! Coupling to physics and the ocean is **important.**

Finite-volume methodology

- In FV^3, all variables are 3D cell- or face-means…not gridpoint values
- We solve not the differential Euler equations but their cell-integrated forms using integral theorems
  - Everything is a flux, including the momentum equation
  - Mass conservation is ensured, to rounding error
  - C-D grid: Vorticity computed *exactly*; accurate divergence computation
  - Mimetic: Physical properties recovered by discretization, particularly Newton's 3rd law

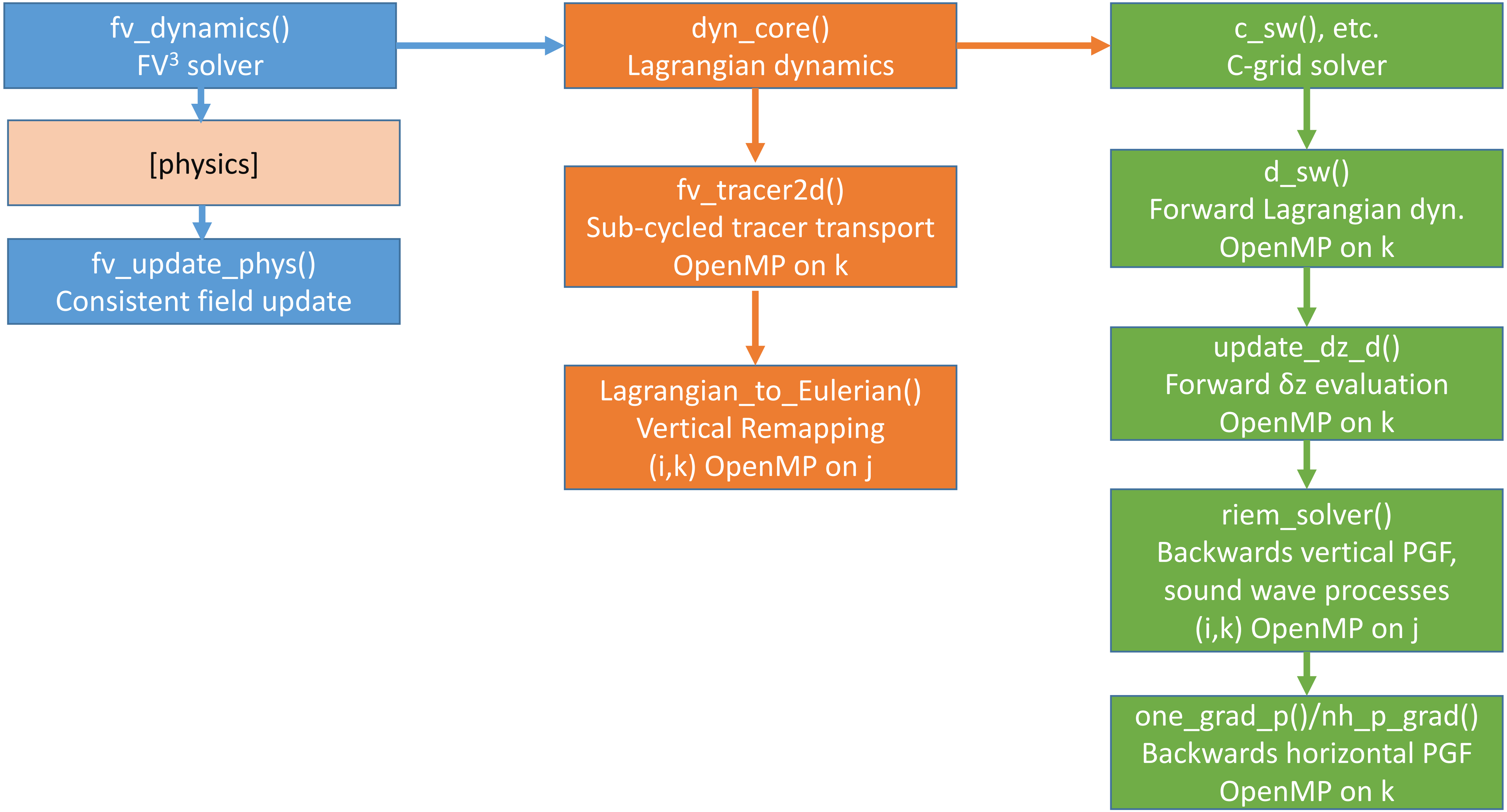
Outline of the FV^3 solver

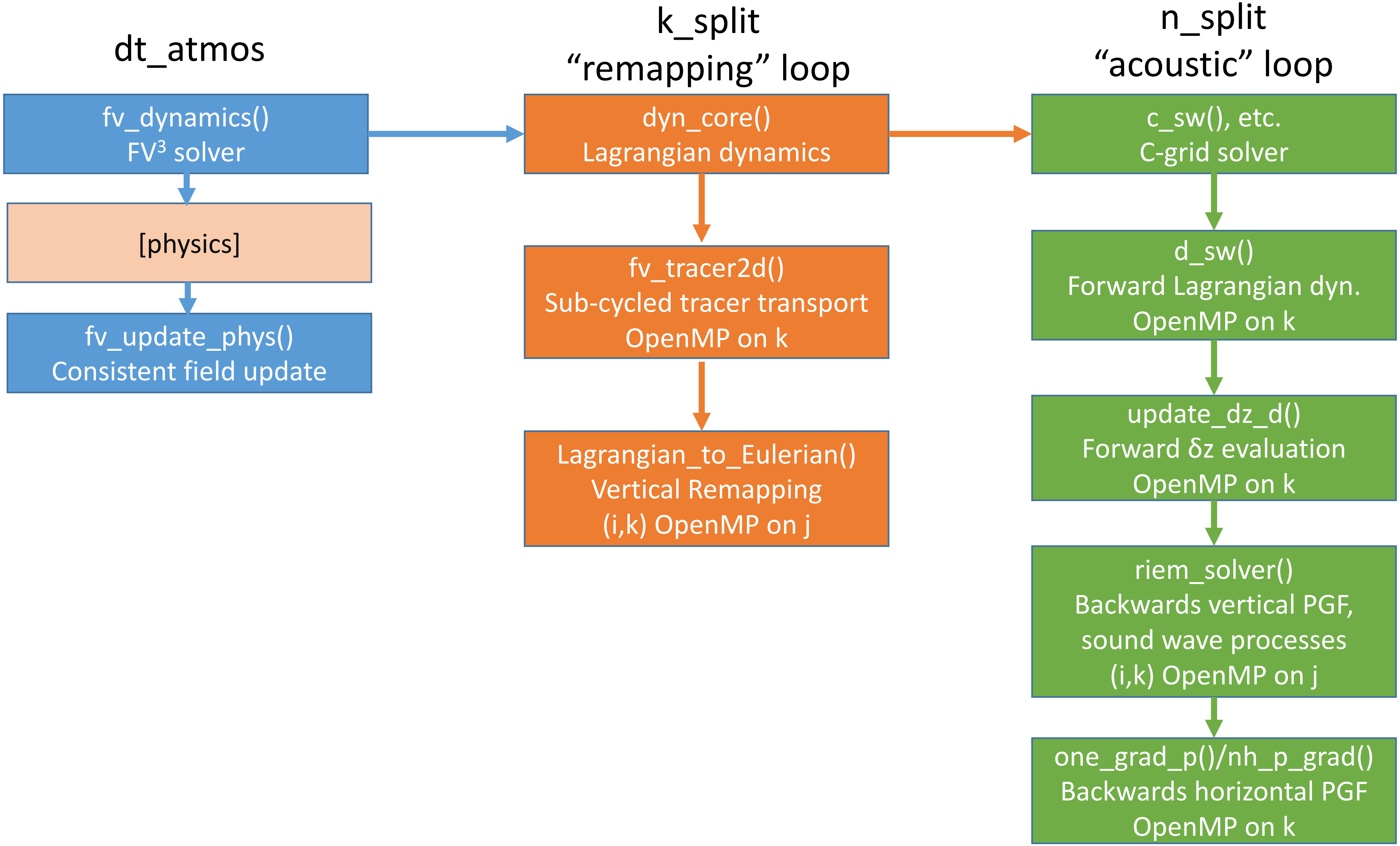
Abstract of FV^3

- Gnomonic cubed-sphere grid for scalability and uniformity
- Fully-compressible vector-invariant Euler equations
- Vertically-Lagrangian dynamics
- C-D grid discretization
- Forward-in-time 2D Lin-Rood advection using PPM operators
- Fully nonhydrostatic with semi-implicit solver
  - Runtime hydrostatic switch

FV^3 time integration sequence

- FV^3 is a forward-in-time solver
  - Flux-divergence terms and physics tendencies evaluated forward-in-time
  - Pressure-gradient and sound-wave terms evaluated backward-in-time for stability
- Lagrangian vertical coordinate: flow constrained along time-evolving Lagrangian surfaces
- Tracers are *sub-cycled* since their stability condition is much less restrictive than the sound- or gravity-wave modes


fv_dynamics()
FV³ solver
[physics]
fv_update_phys()
Consistent field update
dyn_core()
Lagrangian dynamics
fv_tracer2d()
Sub-cycled tracer transport
OpenMP on k
Lagrangian_to_Eulerian()
Vertical Remapping
(i,k) OpenMP on j
c_sw(), etc.
C-grid solver
d_sw()
Forward Lagrangian dyn.
OpenMP on k
update_dz_d()
Forward δz evaluation
OpenMP on k
riem_solver()
Backwards vertical PGF,
sound wave processes
(i,k) OpenMP on j
one_grad_p()/nh_p_grad()
Backwards horizontal PGF
OpenMP on k



dt_atmos
k_split
"remapping" loop
n_split
"acoustic" loop
fv_dynamics()
FV³ solver
[physics]
fv_update_phys()
Consistent field update
dyn_core()
Lagrangian dynamics
fv_tracer2d()
Sub-cycled tracer transport
OpenMP on k
Lagrangian_to_Eulerian()
Vertical Remapping
(i,k) OpenMP on j
c_sw(), etc.
C-grid solver
d_sw()
Forward Lagrangian dyn.
OpenMP on k
update_dz_d()
Forward δz evaluation
OpenMP on k
riem_solver()
Backwards vertical PGF,
sound wave processes
(i,k) OpenMP on j
one_grad_p()/nh_p_grad()
Backwards horizontal PGF
OpenMP on k

Prognostic Variables

δp	Total air mass (including vapor and condensates) Equal to *hydrostatic* pressure depth of layer
θ_v	Virtual potential temperature
u, v	Horizontal D-grid winds in local coordinate (defined on cell faces)
w	Vertical winds
δz	Geometric layer depth
q_i	Passive tracers

Cell-mean pressure, density, divergence, and specific heat are all *diagnostic* quantities
All variables are layer-means in the vertical: **No vertical staggering**

Lagrangian Dynamics in FV^3

What they are

What the equations are

How they are solved

What are Lagrangian Dynamics?

- The Euler equations can be written in Lagrangian or Eulerian forms…
  or Eulerian in the horizontal, and Lagrangian in the vertical
- This constrains the flow along quasi-horizontal surfaces
- Surfaces deform during the integration, representing vertical motion and advection “for free”
- Does require layer thickness to be a prognostic variable

$$
\begin{aligned}
D_L \delta p^* + \nabla \cdot (\mathbf{V} \delta p^*) &= 0 \\
D_L \delta p^* \Theta_v + \nabla \cdot (\mathbf{V} \delta p^* \Theta_v) &= 0 \\
D_L \delta p^* w + \nabla \cdot (\mathbf{V} \delta p^* w) &= -g \delta z \frac{\partial p'}{\partial z}
\end{aligned}
$$

$$
\begin{aligned}
D_L u &= \Omega v - \frac{\partial}{\partial x}\kappa - \frac{1}{\rho}\frac{\partial p^*}{\partial x}\Big|_z \\
D_L v &= -\Omega u - \frac{\partial}{\partial y}\kappa - \frac{1}{\rho}\frac{\partial p^*}{\partial x}\Big|_z
\end{aligned}
$$

$$
\boxed{D_L \phi = \frac{\partial \phi}{\partial t} + \frac{\partial}{\partial z}(w\phi)}
$$

Lagrangian Dynamics: Flux-form advection

$$q^{n+1} = \frac{1}{\pi^{n+1}} \left\{ \pi^n q^n + F\left[q^n + \frac{1}{2} g(q^n) \right] + G\left[q^n + \frac{1}{2} f(q^n) \right] \right\}.$$

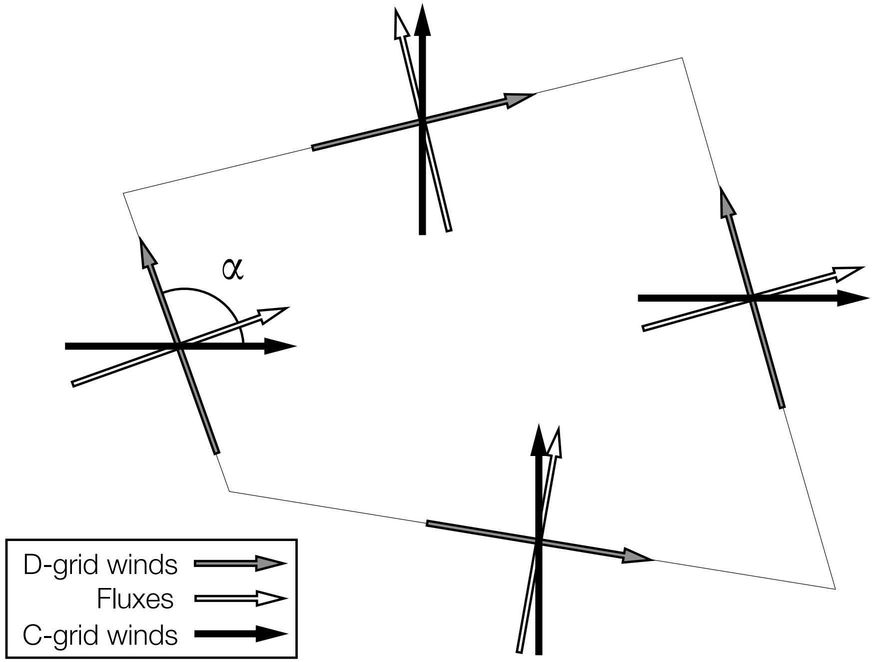
- Advection is not just for passive tracers!
  Nearly everything in FV^3 is a flux
  - Even the momentum advection terms can be expressed as scalar fluxes
- Lin-Rood (1996) 2D advection scheme:
  - Reverse-engineered scheme devised from 1D PPM operators
  - Allows monotonicity and positivity from subgrid reconstructions
  - Monotonicity is “smart” diffusion
- Again: advection is along Lagrangian surfaces

Lagrangian Dynamics:
Tracer advection and sub-cycling

- Tracers can be advected with a longer timestep than the dynamics
- FV^3 permits accumulation of mass fluxes during Lagrangian dynamics. These fluxes are then used to compute the advection of tracers before the vertical remapping
  - Typically one or two tracer timesteps is sufficient for stability.

- Tracer advection is *always* monotone (or at least positive definite) to avoid new extrema. Explicit diffusion is not used.
  - **Question:** what about physical eddy diffusion (cf. PBL scheme?)

Lagrangian Dynamics: C-D grid solver

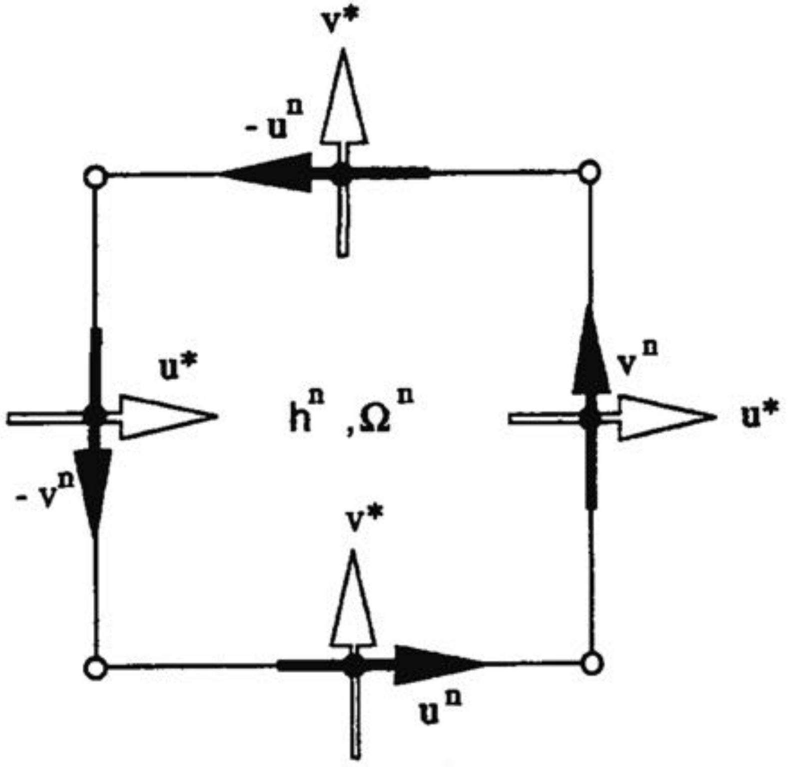


- FV^3 solves for the (purely horizontal) D-grid staggered winds. But solver requires face-normal fluxes
- To compute time step-mean fluxes, the C-grid winds are interpolated and then advanced a half-timestep.
  - A sort of simplified Riemann solver
  - The C-grid solver is the same as the D-grid, but uses lower-order fluxes for efficiency
- Two-grid discretization and time-centered fluxes avoid computational modes

Lagrangian Dynamics: Momentum equation

$$\frac{\partial \mathbf{V}}{\partial t} = -\Omega \hat{k} \times \mathbf{V} - \nabla\left(\kappa + \nu \nabla^2 D\right) - \frac{1}{\rho}\nabla p\Big|_z$$

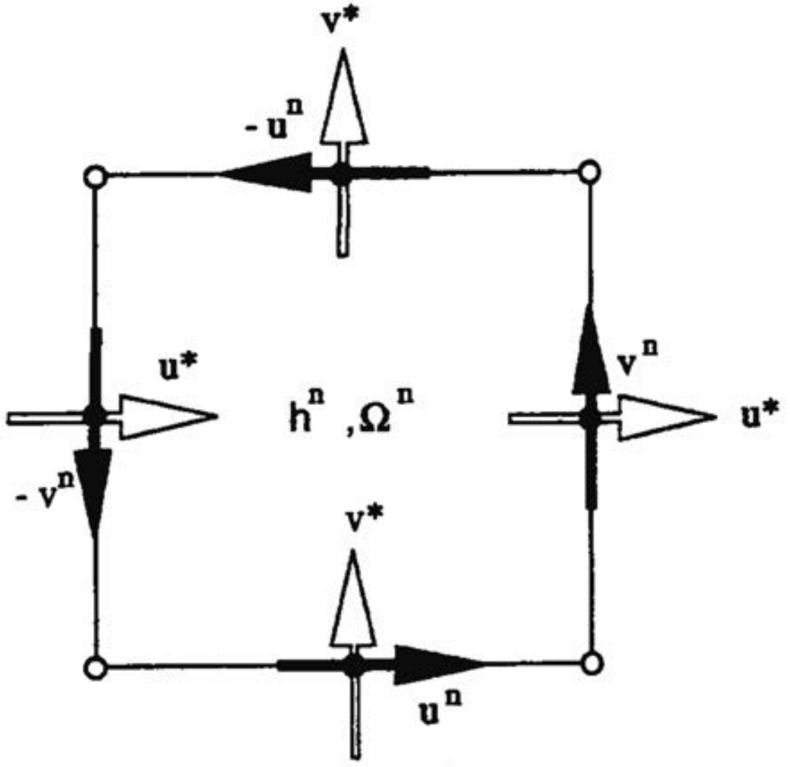
- FV3 solves the flux-form vector invariant equations
- Nonlinear vorticity flux term in momentum equation, confounding linear analyses
- D-grid allows exact computation of absolute vorticity—no averaging!
- Vorticity uses same flux as δp: consistency improves geostrophic balance, and SW-PV advected as a scalar!



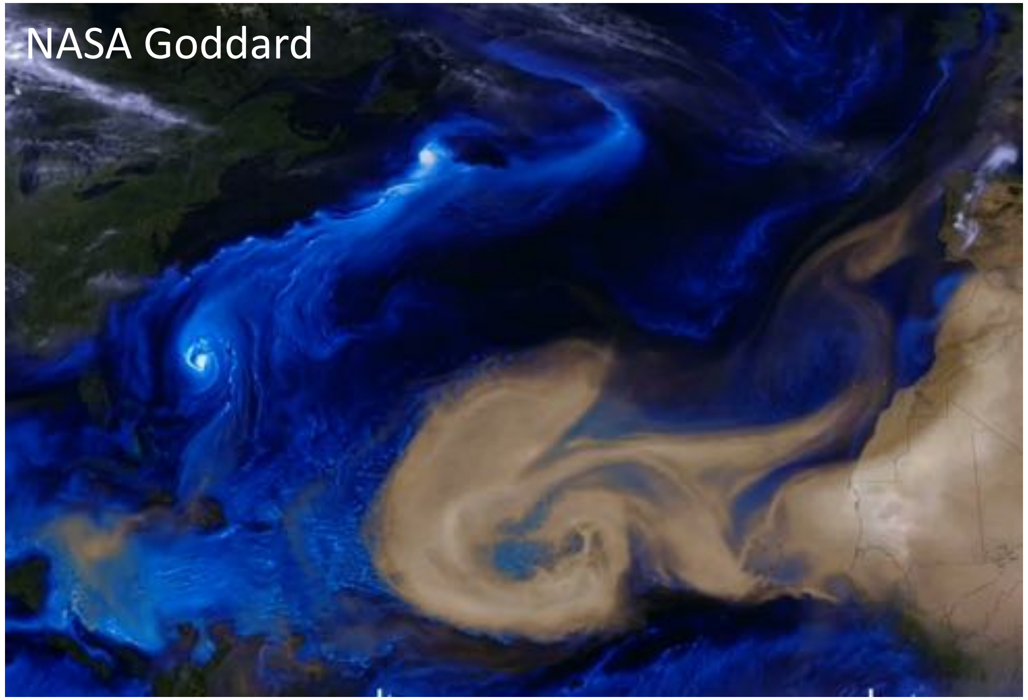
Lagrangian Dynamics: Momentum equation

$$\frac{\partial \mathbf{V}}{\partial t} = -\Omega \hat{k} \times \mathbf{V} - \nabla\left(\kappa + \nu \nabla^2 D\right) - \frac{1}{\rho}\nabla p\Big|_z$$

- FV3 solves the flux-form vector invariant equations
- Nonlinear vorticity flux term in momentum equation, confounding linear analyses
- D-grid allows exact computation of absolute vorticity—no averaging!
- Vorticity uses same flux as **w**: consistency improves **nonlinear** balance, and **updraft helicity** advected as a scalar!



Many flows are vortical
Not just large-scale flows



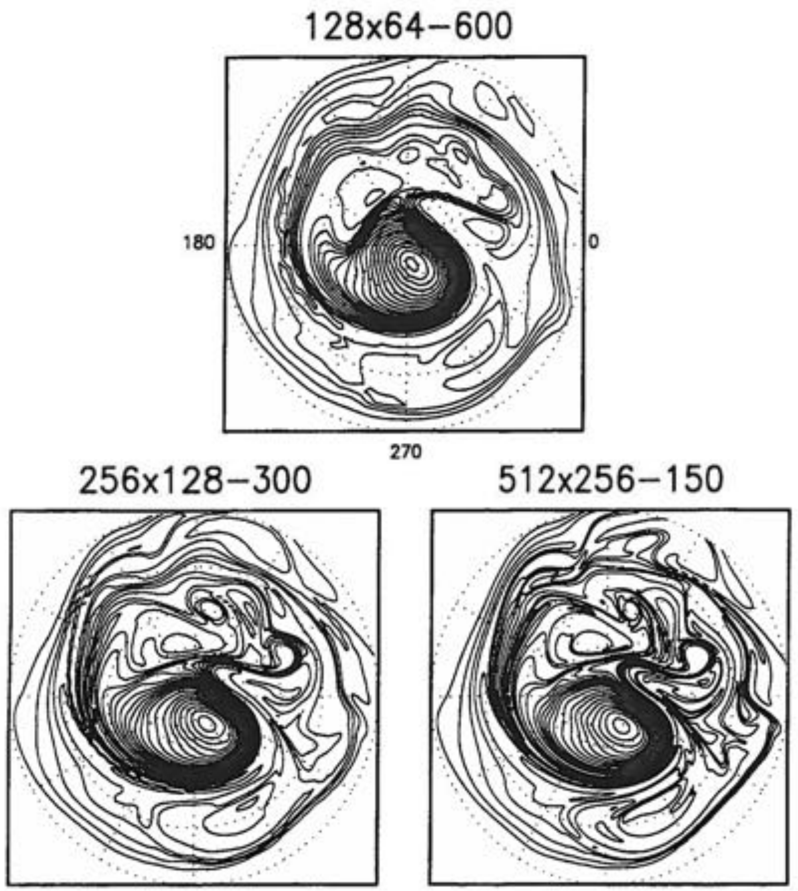
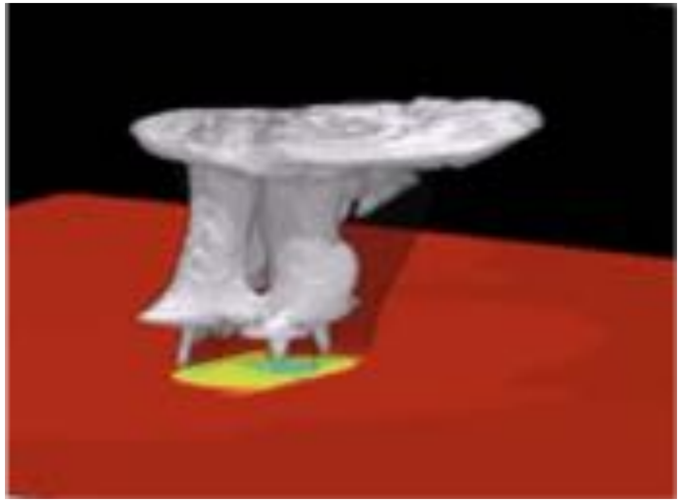
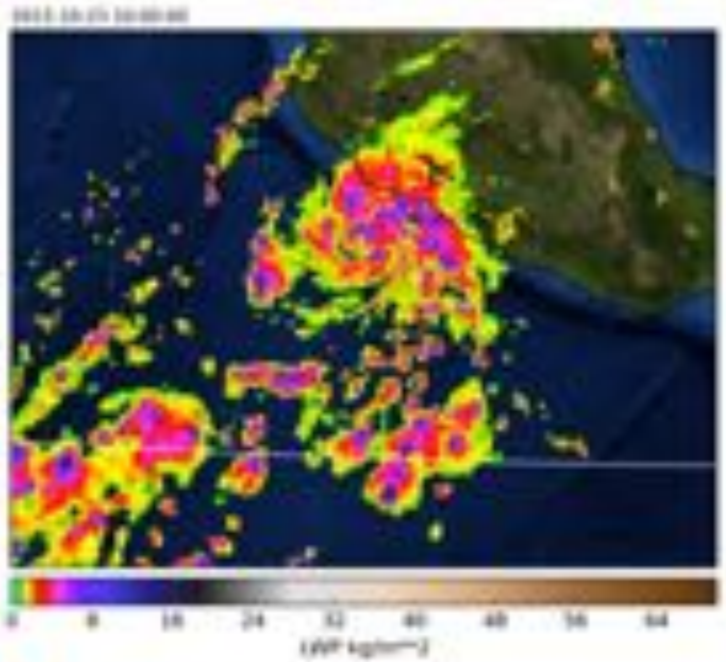


Figure 10. Polar stereographic projection (from the equator to the north pole) of the potential vorticity contours at DAY-24 in the 'stratospheric vortex erosion' test case at three different resolutions.





May I talk to you about…diffusion?

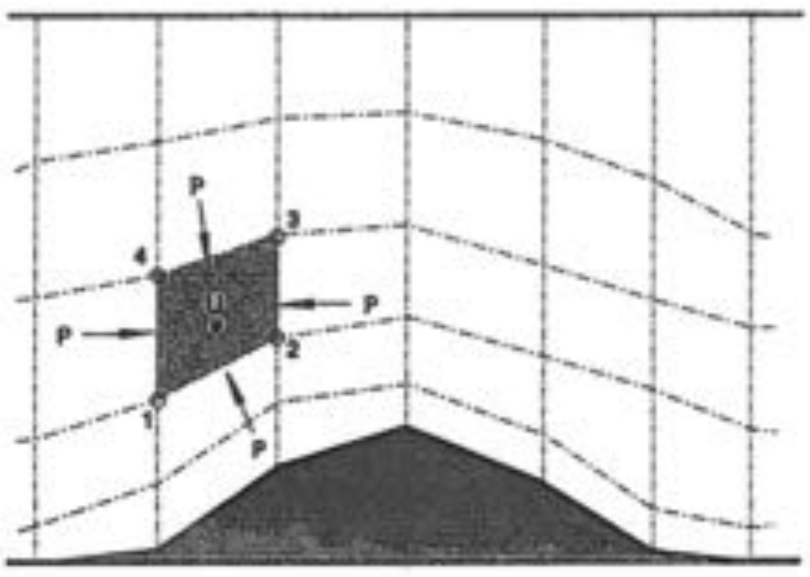
- All dynamical cores require artificial diffusion (implicit or explicit) to remove energy cascading to grid scale
- FV^3 has implicit diffusion from monotone advection, forward-backward timestepping, and vertical remapping
- Except for vertical remapping…all explicit and implicit diffusion is along the Lagrangian surfaces

May I talk to you about…diffusion?

- C-D grid does not explicitly see divergence—so explicit divergence damping is an intrinsic part of the solver
- Non-monotone advection is useful in nonhydrostatic dynamics Noise can be controlled by adding diffusion to the vorticity fluxes
  - Explicitly-dissipated kinetic energy can be converted to **heat**
  - Divergence and vorticity damping can be controlled separately
- A simplified second-order Smagorinsky damping, with a nonlinear flow-dependent coefficient, is also available

Backward horizontal pressure gradient force

- Computed from Newton's second and third laws, and Green's Theorem
- Errors lower, with much less noise, compared to traditional evaluations
    - Purely horizontal
      no along-coordinate projection
    - PGF equal and opposite—3rd law!
      Momentum conserved
    - Curl-free in the absence of density gradients
- Nonhydrostatic and hydrostatic components can be computed separately
    - $\log(p_{hyd})$ PGF more accurate



$$\left(\frac{du}{dt}, \frac{dw}{dt}\right) = \frac{1}{\Delta m}(\Sigma \mathbf{F}_x, \Sigma \mathbf{F}_z)$$

$$\Sigma \mathbf{F} = \int_C P\mathbf{n}\, ds$$

$$\frac{du}{dt} = g\frac{\Sigma \mathbf{F}_x}{\Sigma \mathbf{F}_z} = g/\tan\gamma$$

Backward horizontal pressure gradient force

- Computed from Newton's second and thir
  laws, and Green's Theorem
- Errors lower, with much less noise, compared to traditional evaluations
  - Purely horizontal
    no along-coordinate projection
  - PGF equal and opposite—3rd law!
    Momentum conserved
  - Curl-free in the absence of density gradients
- Nonhydrostatic and hydrostatic componen
  can be computed separately
  - $\log(p_{hyd})$ PGF more accurate

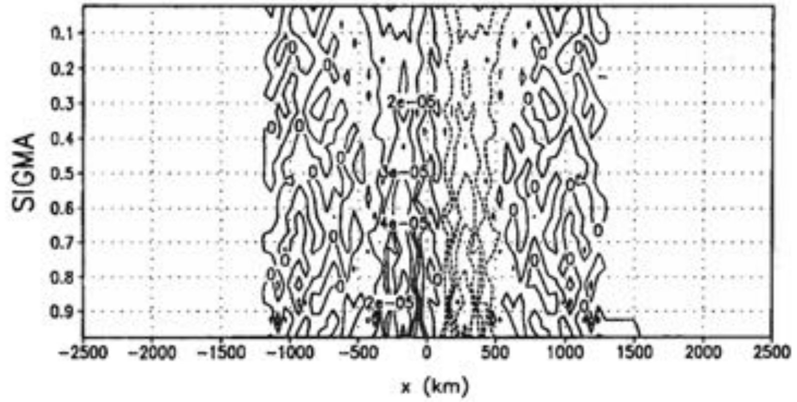


Figure 5. The pressure gradient (m s^{-2}) computed by the Arakawa–Suarez method. Contour interval is 1×10^{-5}.

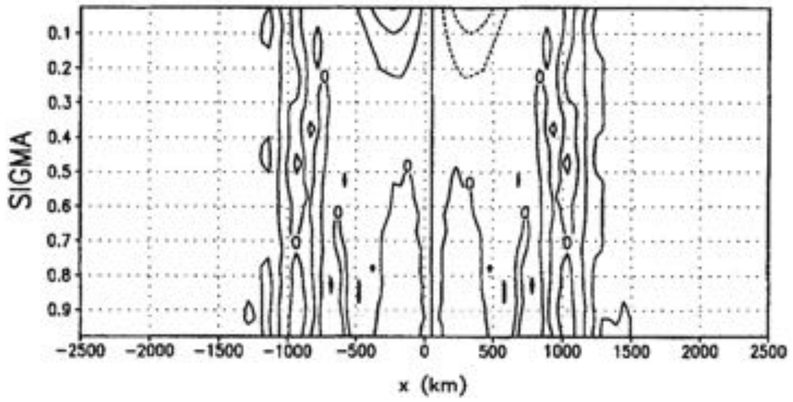
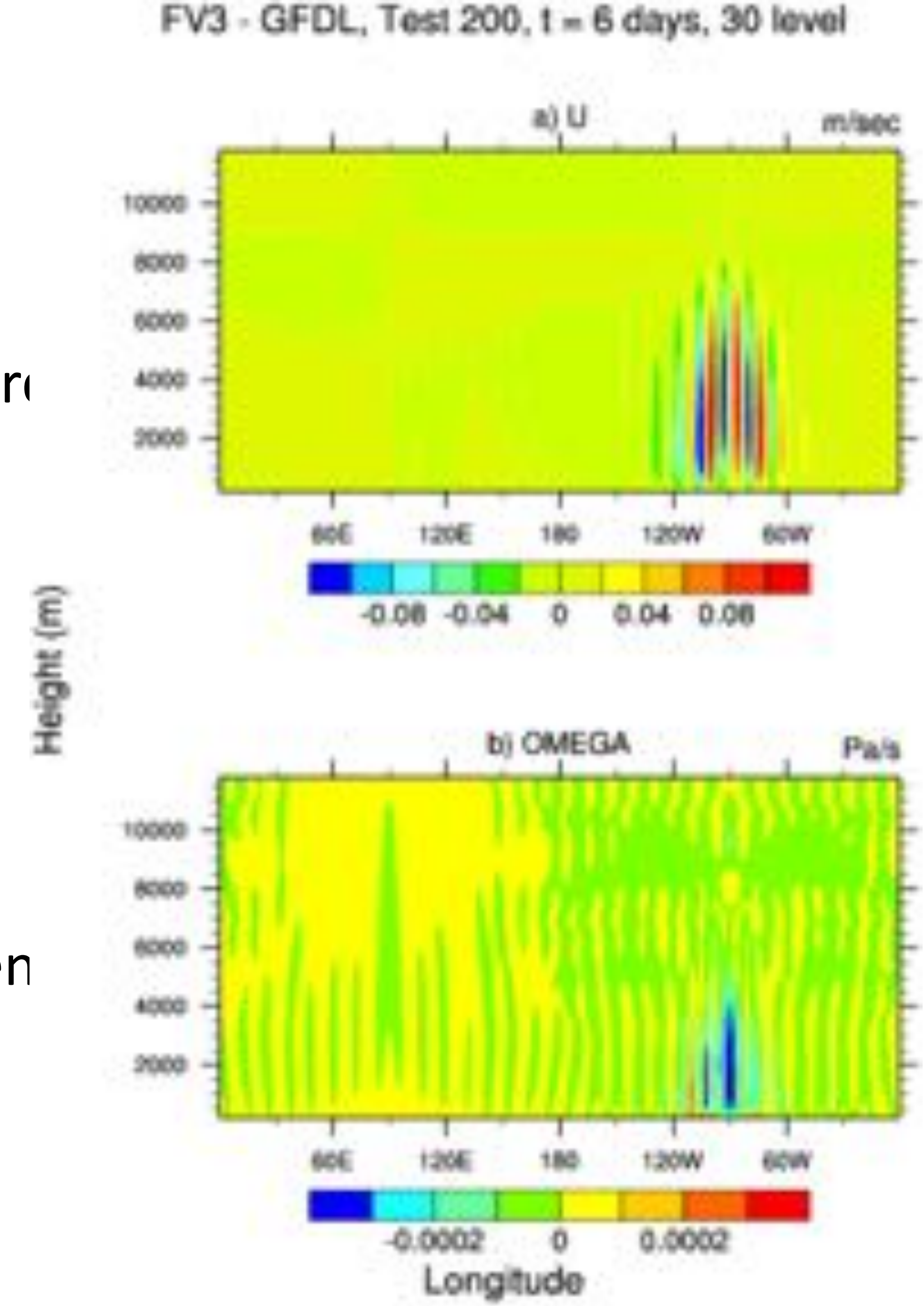


Figure 6. As in Fig. 5, but for the finite-volume method.

Backward horizontal pressure gradient force

- Computed from Newton's second and third laws, and Green's Theorem
- Errors lower, with much less noise, compared to traditional evaluations
    - Purely horizontal
      no along-coordinate projection
    - PGF equal and opposite—3rd law!
      Momentum conserved
    - Curl-free in the absence of density gradients
- Nonhydrostatic and hydrostatic components can be computed separately
    - $\log(p_{hyd})$ PGF more accurate



Vertical processes:
Vertical elastic terms
and Lagrangian vertical coordinate

Semi-implicit solver

- Vertical pressure gradient and layer depth δz is solved by semi-implicit solver
  - Vertically-propagating sound waves weakly damped
- This is all that is needed to make the classic FV hydrostatic algorithm nonhydrostatic
  - Fully compressible and nonhydrostatic! Full Euler equations solved
  - w, δz advected as other variables—consistent!
  - Nonhydrostatic horizontal PGF evaluated same way as hydrostatic

The Lagrangian Vertical Coordinate

- The domain is separated into a number of quasi-horizontal Lagrangian layers (k index)
- All flow is within the layers ("logically" horizontal)
- **No** cross-layer layer flow **or diffusion**
  Vertical motion deforms the layers instead
- Periodically, a highly-accurate conservative remapping is done to avoid layers becoming infinitismally thin ($\delta p \rightarrow 0$)
  - Remapping interval like Lagrangian advection's timestep:
    longer timesteps yield less overall artificial diffusion
  - This is the **only** way cross-layer diffusion is introduced!!

The Lagrangian Vertical Coordinate

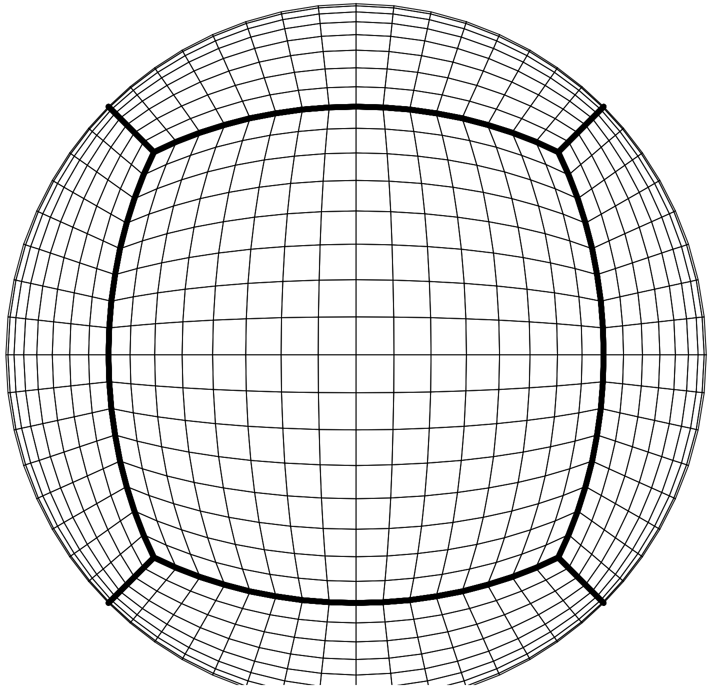
- The mass δp and height δz are prognostic variables.
  Height and pressure of each layer are dynamically computed.
- Lagrangian coordinate works for *any* base coordinate
  - Hybrid-pressure, hybrid-height, and hybrid-isentropic coordinates have been successfully used
- Vertical advection is *implicit* through the vertical movement of layers
  - There is no Courant number restriction or time-splitting!
  - Vertical advection does not need a separate computation.
    Computing δp and δz is sufficient.
- Implicit advection not only saves time but also permits very thin layers without requiring a smaller timestep

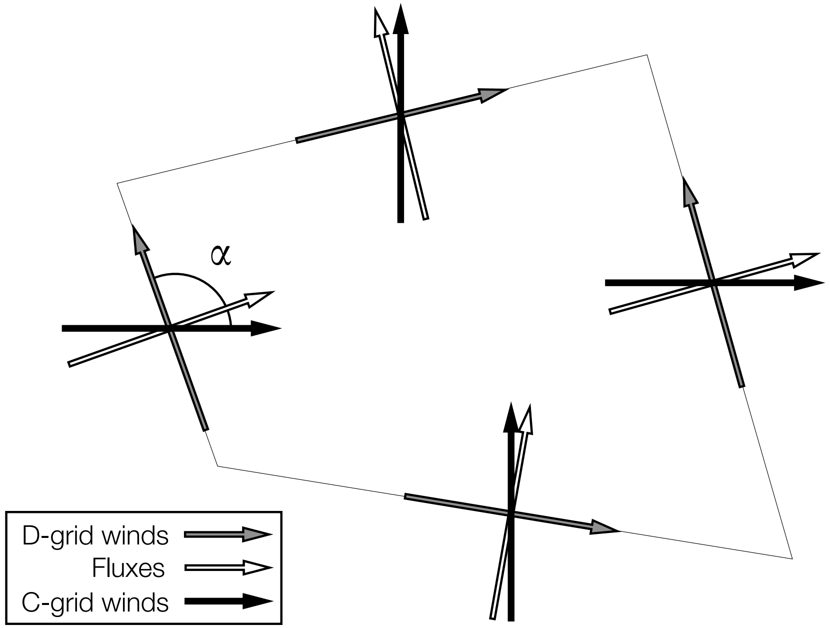
The Cubed-Sphere Grid

The 3 in FV^3

The Cubed-Sphere Grid

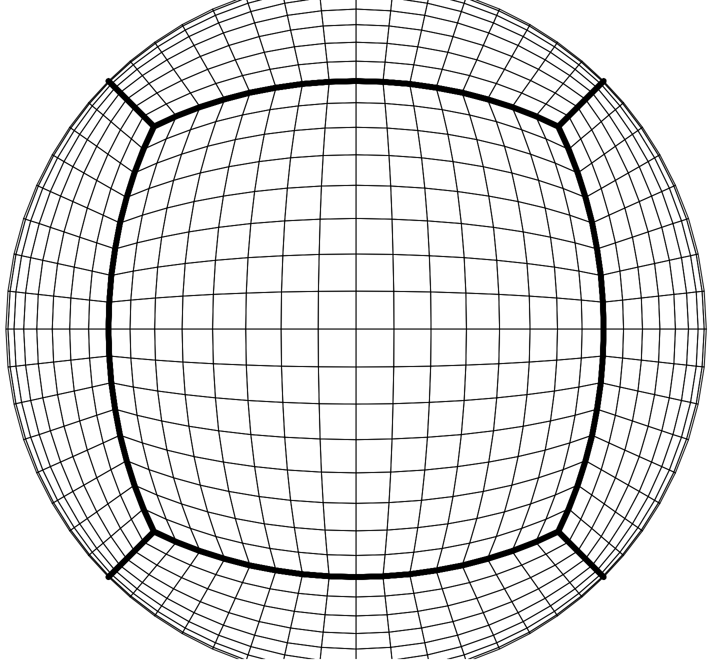
- Gnomonic cubed-sphere grid: coordinates are great circles
- Widest cell only √2 wider than narrowest
- More uniform than conformal, elliptic, or spring-dynamics cubed spheres
  - Tradeoff: coordinate is non-orthogonal, and special handling needs to be done at the edges and corners.

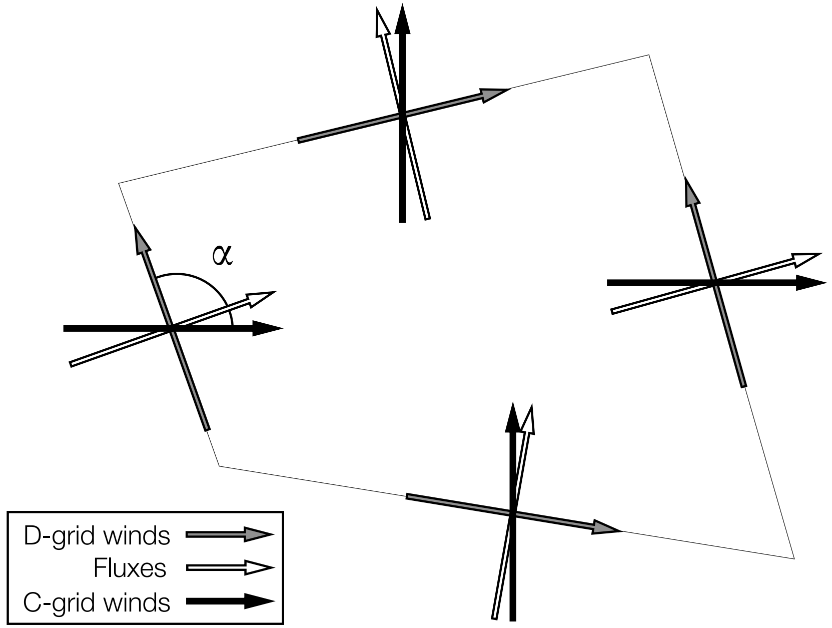




The Cubed-Sphere Grid

- Gnomonic cubed-sphere is non-orthogonal
- Instead of using numerous metric terms, use covariant and contravariant winds
  - Solution winds are covariant, advection is by contravariant winds
  - KE is half the product of the two
- Winds u and v are defined in the local coordinate: rotation needed to get zonal and meridional components





A little bit about initialization

FV^3 has a host of utilities to generate grids, orography, and initial conditions from cold start

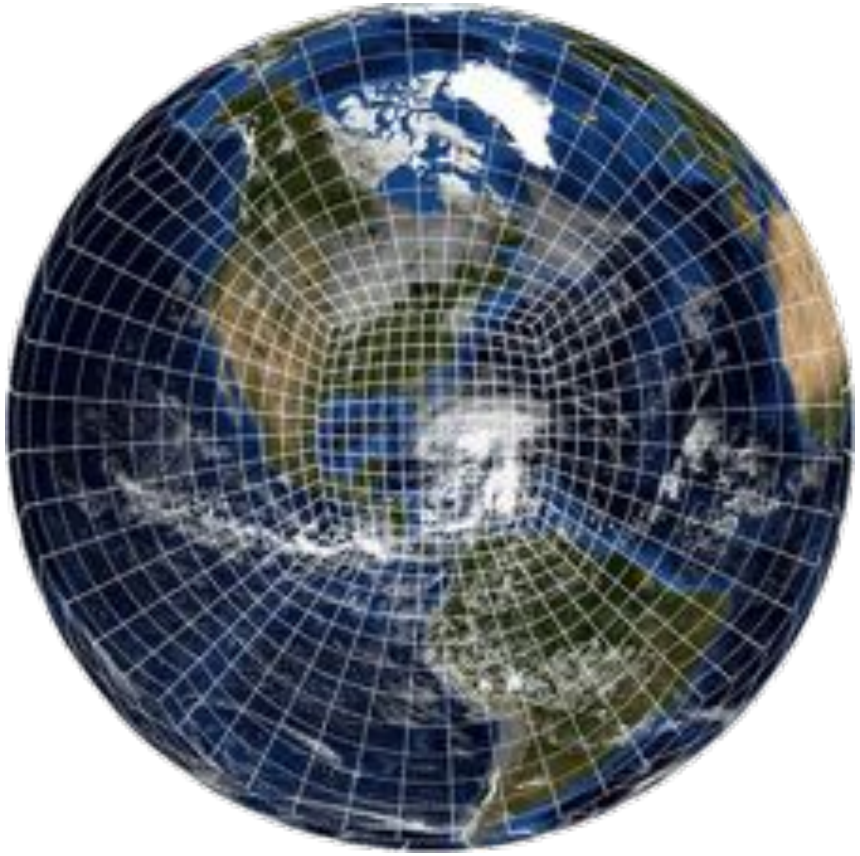
- Any grid can be generated in seconds
- Initialize from NCEP or EC analyses
- Remap from different vertical spacings
- Comprehensive topography generation, including subgrid orography
- Advanced FCT orography filter allows preservation of total topography, peaks, and valleys; limits slope steepness; and prevents nonzero topography in the ocean
- Atmospheric nudging to analyses for simple assimilated initialization, or for aerosol and chemistry studies

Essential Runtime Options

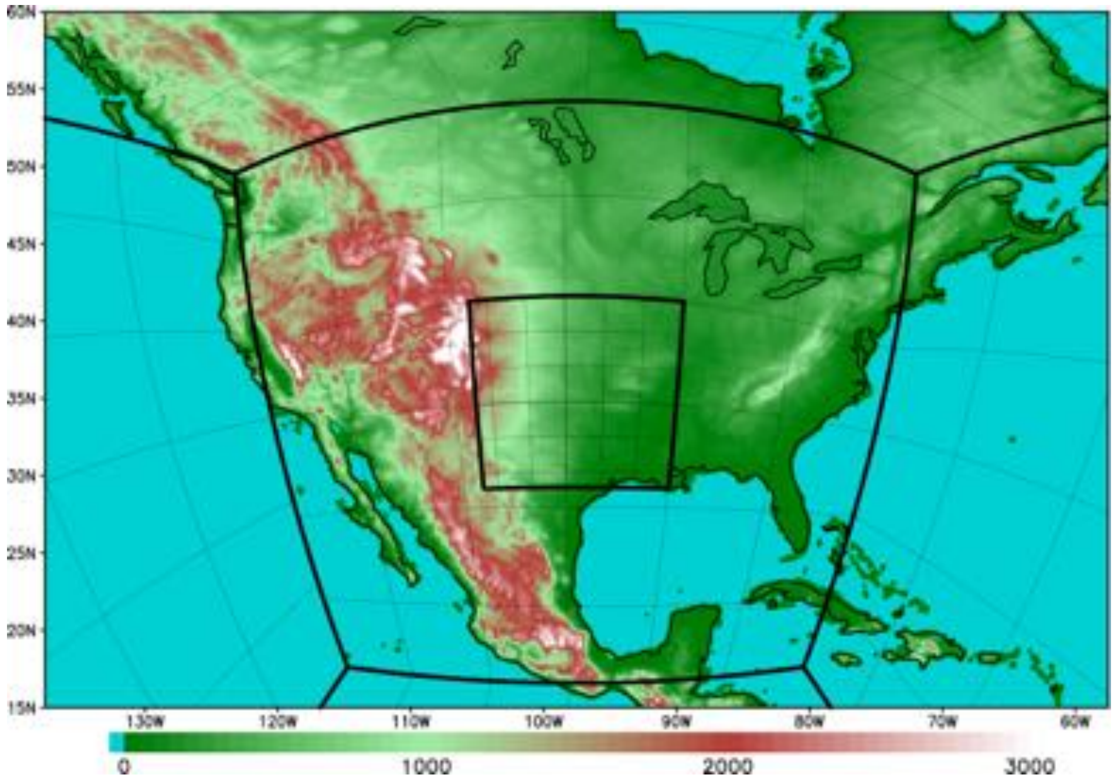
- `dt_atmos`: physics timestep
- `k_split`: Number of vertical remappings per `dt_atmos`
- `n_split`: Number of "acoustic" timesteps per `k_split`
- `hord_xx`: Horizonal advection algorithms
- `kord_yy`: Vertical remapping options
- `d4_bg`, `vtdm4`: explicit damping options for divergence and for fluxes (except scalars), of order 2*(`nord`+1)
- `dddmp`: Smagorinsky damping coefficient
- `d_con`: Converts explicitly-damped KE to heat

Grid refinement (preview)

FV^3 supports both stretching and nesting for grid refinement



Grid stretching is simple and smooth



Grid nesting is efficient and flexible

- FV3 is able to mimic many physical properties, particularly Newton’s laws, mass conservation, and excellent vorticity dynamics
- Lagrangian dynamics is very powerful!
  - Increased parallelism
  - Implicit vertical advection, without computation
  - Much reduced implicit vertical diffusion
  - Improves PBL and surface interaction
- Solver is fully compressible, and so horizontally local
- Fully nonhydrostatic solver maintains excellent hydrostatic flow while consistently implementing nonhydrostatic elastic terms
- Flexible advection, diffusion, and grid structure options