VARIABLE RESOLUTION METHODS IN FV^3

Lucas Harris

and the GFDL FV^3 Team

FV3 Summer School

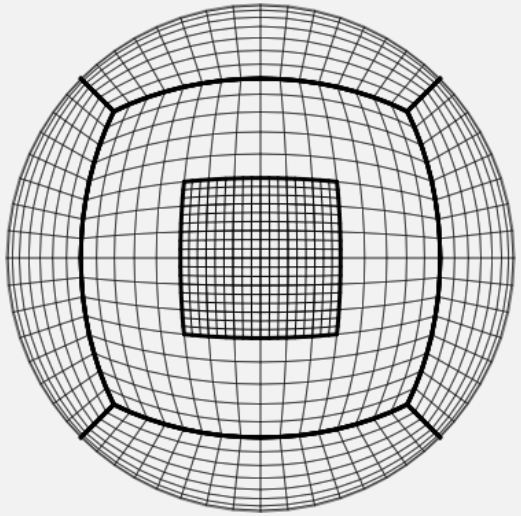
12 June 2018

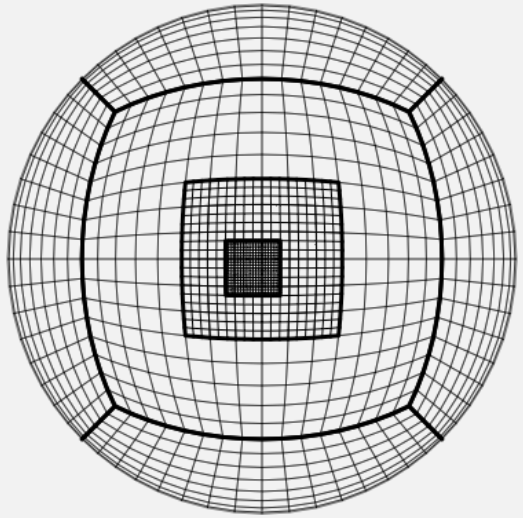
HIGH RESOLUTION MODELING: LIMITED AREA VS. GLOBAL MODELS

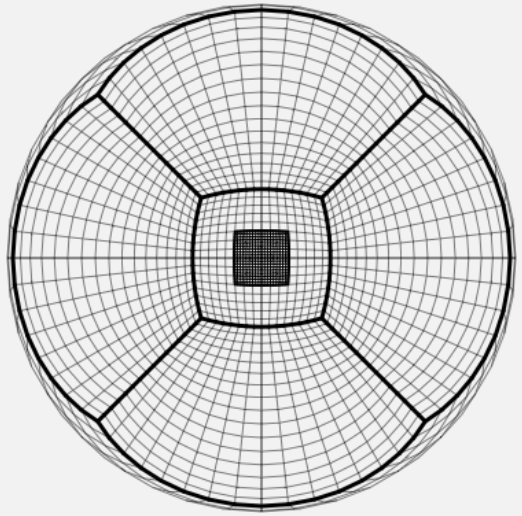
- Stand-alone regional models are commonly used for mesoscale simulation and regional climate modeling. But boundary errors creep in after a few days.
  - Require potentially inconsistent BCs from a global model.
  - No feedback onto large-scale
- Global models have no boundaries and are globally consistent, but global high resolution can be impractical
- **Solution: grid refinement of a global model!**

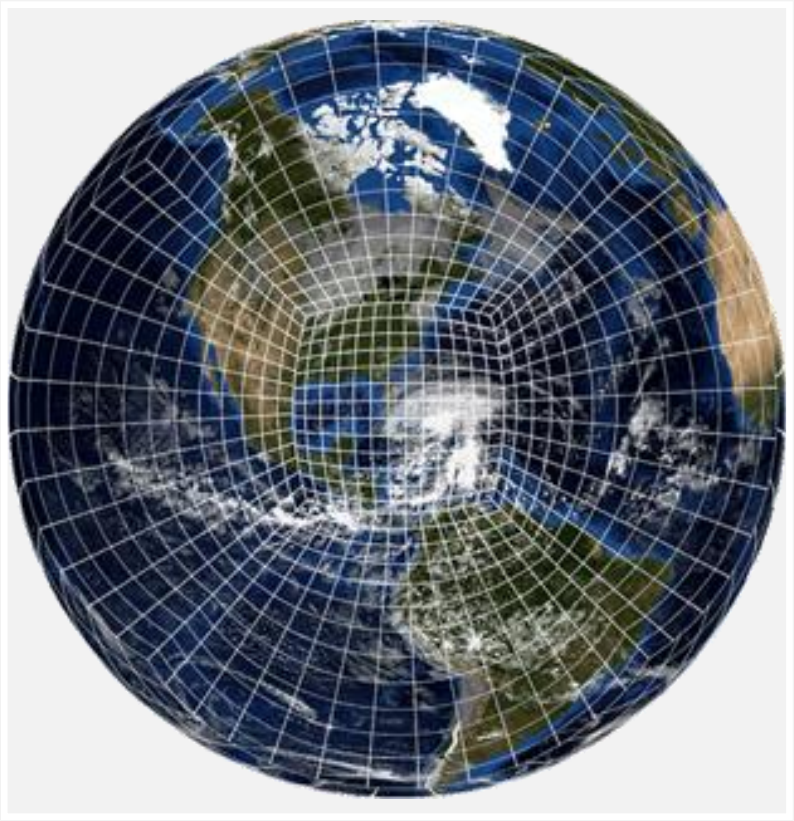
GRID REFINEMENT

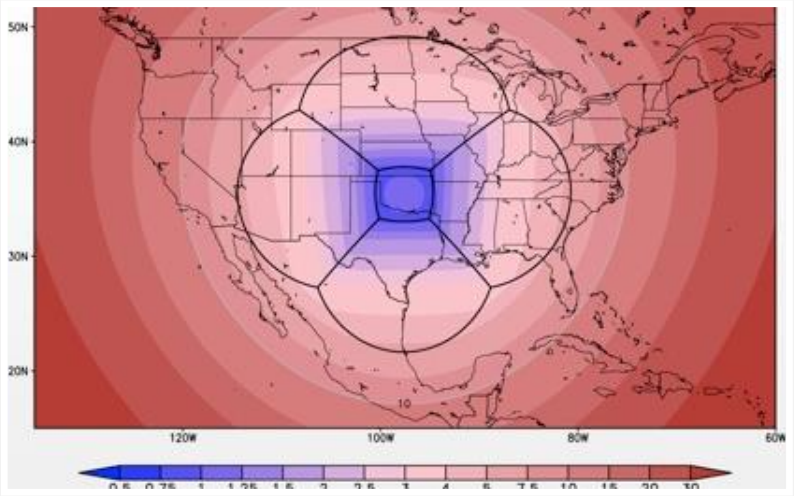
- FV^3 supports both grid stretching and two-way grid nesting
- Grids can be constructed on-the-fly within seconds.
- Both techniques have strengths and weaknesses:
  Combining the two leads to the best results











STRETCHED GRID

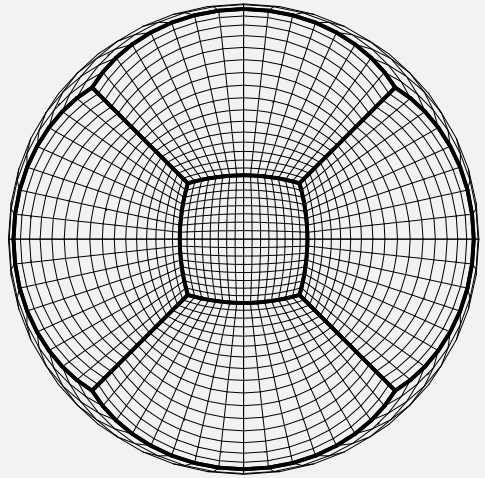
- **The simple, easy way to achieve grid refinement**
- Smooth deformation! Requires no changes to the solver
- No abrupt discontinuity.
- Capable of extreme refinement (80x!!) for easy storm-scale simulations on a full-size earth
- Requires a “compromise” tuning between coarse and fine regions. (A good scale-aware scheme can help here.)

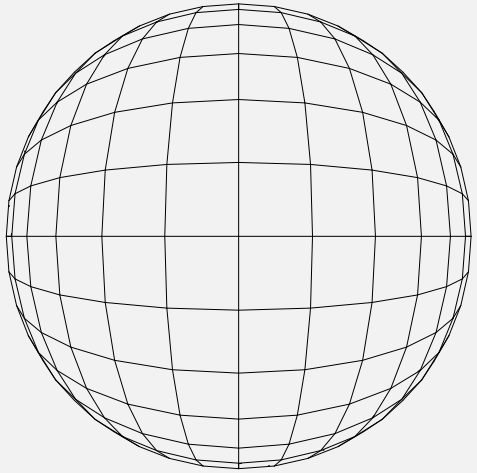
SCHMIDT TRANSFORMATION

$$\sin\vartheta = \frac{D + \sin\theta}{1 + D\sin\theta}$$

$$D = \frac{1 - c^2}{1 + c^2},$$

- Smoothly deforms the cubed sphere into a "truncated pyramid", with the high-resolution face at the top of the "pyramid"
- Transformation is **analytic**. Easily implemented and quickly executed
- The resulting pyramid can then be rotated to an arbitrary target point
  - Transformation does coarsen the opposing side of the sphere
  - Size of high-resolution region decreases with increasing stretching ratio

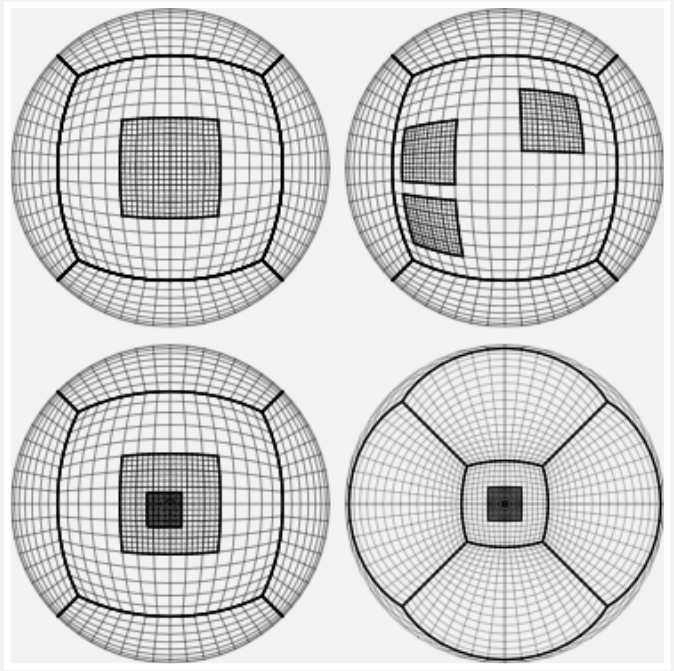


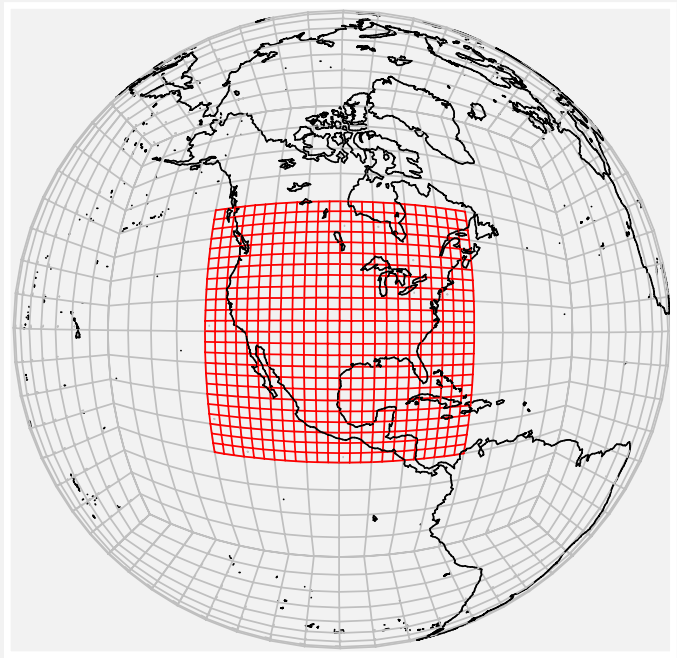


STRETCHED GRID NAMELIST OPTIONS

- do_Schmidt: enables Schmidt transformation for stretching and/or rotation. Other parameters ignored if this is not set.
- stretch_fac: Stretching factor (c). Set to 1.0 if no stretching is desired
- target_lat and target_lon: Center (degrees) of the high-resolution stretched tile (tile 6).

The same parameters used to create the grid in the preprocessing tools must be specified in the global grid's namelist.





TWO-WAY GRID NESTING

- Simultaneous coupled, consistent global and regional solution. No waiting for a regional prediction!
- Different grids permit different parameterizations and timesteps; doesn't need a "compromise" for high-resolution region
- Flexible! Great possibilities for combining nesting and stretching.

GRID NESTING: BOUNDARY CONDITIONS

- Strategy: fill halo (ghost) cells with boundary conditions
  Interior of solver needs no changes
- All variables linearly interpolated in space into nested grid halo.
  Correct upwind BCs "baked in" by FV's upstream-biased fluxes
  - BCs for all solution variables, as well as C-grid winds, and divergence
- Nonhydrostatic solver requires nonhydrostatic pressure, computed using the semi-implicit solver—consistent with interior algorithm

GRID NESTING: BOUNDARY CONDITIONS

- **Concurrent nesting**: BCs extrapolated in time so nest and coarse grids can run **simultaneously**.
  - BCs stepped forward every acoustic timestep
  - New BC data updated at the nest interaction frequency, usually vertical remap frequency
- Extrapolation is formally unstable but is not a problem in practice
  - Extrapolation is limited to ensure positivity for scalars
- Two-time level extrapolation requires saving BCs across restarts to ensure run-to-run reproducibility

GRID NESTING
TOPOGRAPHY AND SMOOTHING

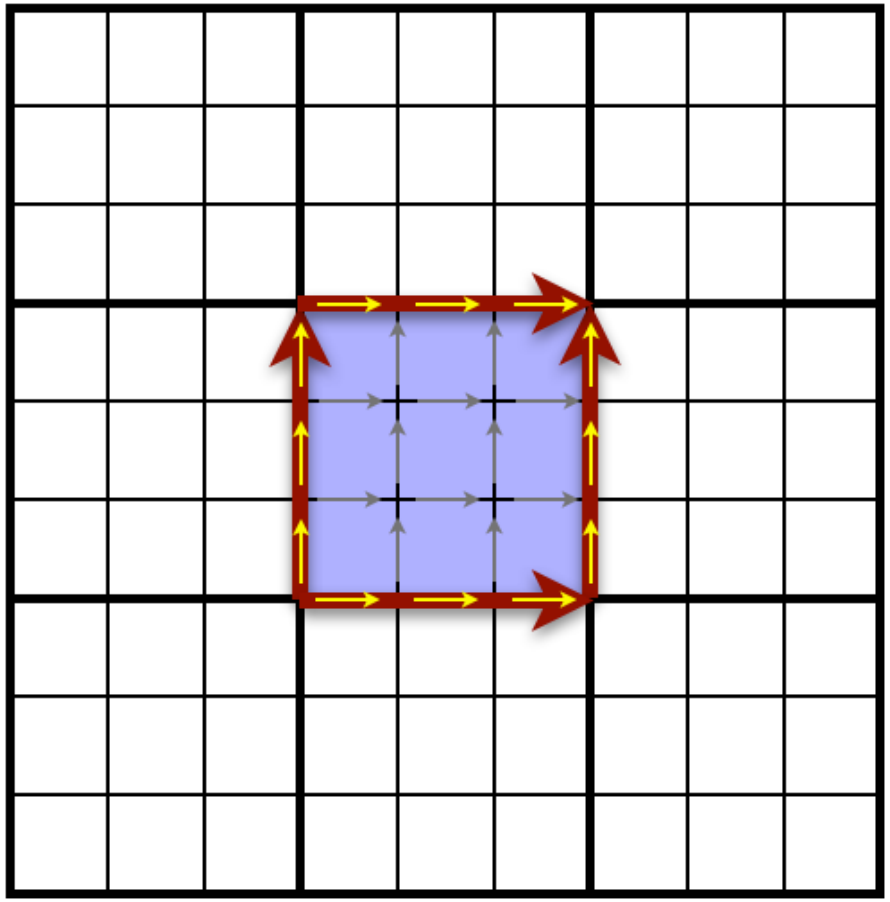
- FV^3 applies **no** additional diffusion or relaxation at the boundaries
  - Linear interpolation introduces some smoothing without creating new extrema
- For consistency, halo topography is linearly interpolated from the coarse grid, the same way as the solution variables
- Topography near the boundary is blended with the interpolated coarse-grid topography

TWO-WAY INTERACTION

- Two-way nesting: coarse grid solution periodically replaced ("updated") by nested-grid solution where the grids coincide
- Essential for small-to-large-scale interaction (e.g. hurricanes, gravity-wave drag, small-scale orographic/coastal processes)
- Theory suggests two-way nesting yields a better nested-grid solution: Harris and Durran (2010), T.T. Warner et al (1997)

TWO-WAY INTERACTION

- Averaging update consistent with FV discretization
  - For consistency, the initialized coarse-grid topography is updated from the nested-grid using the same algorithm
- Cell average on scalars, including δz and w
- In-line average for winds, to conserve vorticity



MASS CONSERVATION AND TWO-WAY NESTING

- Conservation usually requires flux BCs at the nested-grid boundary
  These are difficult to implement with the time-extrapolation BC
- **Our approach:** Update everything except mass (δp) and tracers
- **Very simple!** Works regardless of BC and grid alignment
  - Two-way nesting over-specifies coarse-grid solution
    Less updating, less over-specification

MASS CONSERVATION AND TWO-WAY NESTING

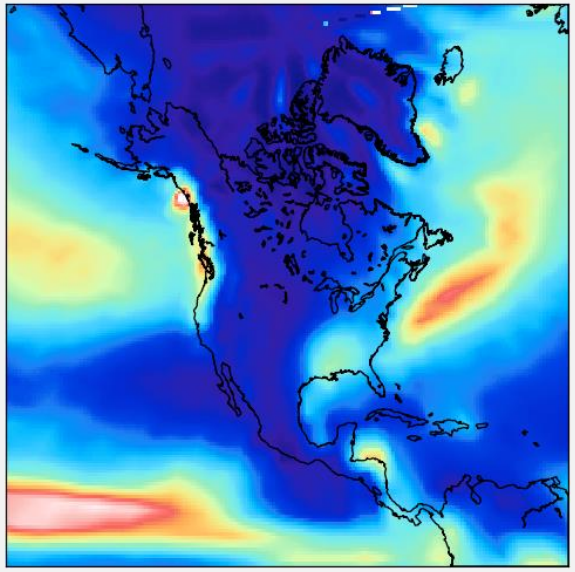
- **Our approach:** Update everything except mass (δp) and tracers
- Because δp is the vertical coordinate, we then need to remap the nested-grid data to the coarse grid's vertical coordinate
- Under development: "Renormalization-update" for tracers uses a layer-by-layer fixer to ensure tracer mass conservation

TWO-WAY VS. ONE-WAY GRID NESTING

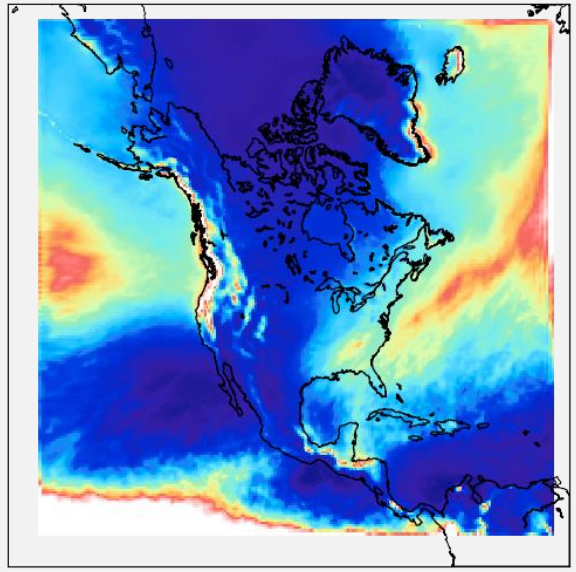
- GFDL HiRAM climate model

  c90 (1°) and
  c90n3 (1° and 1/3°)

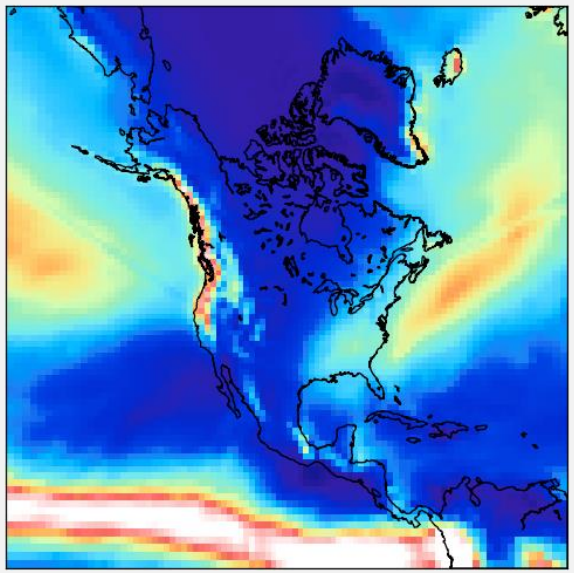
CMAP DJF Observations



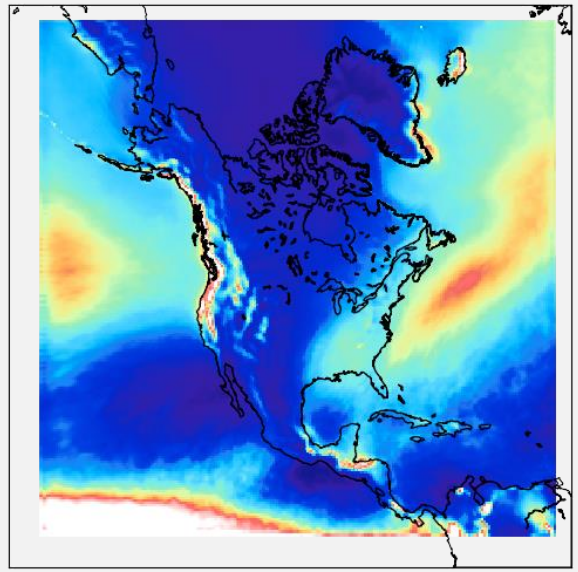
One-way 1/3° climo SST

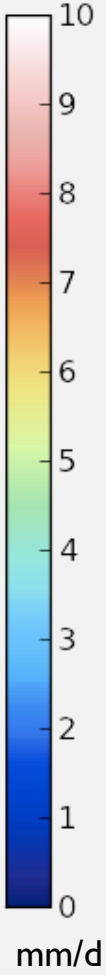


1° uniform AMIP

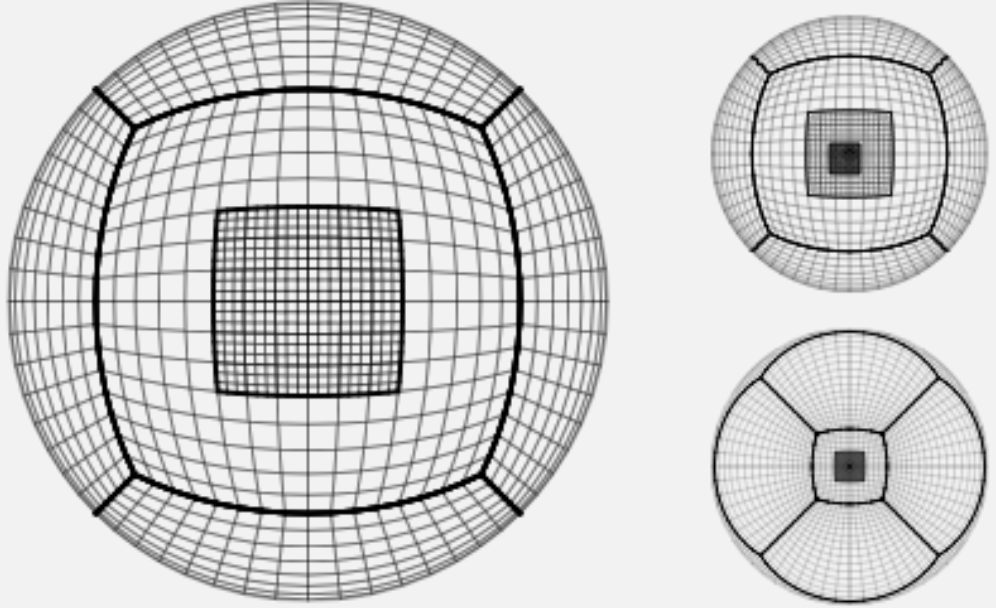


Two-way 1/3° AMIP

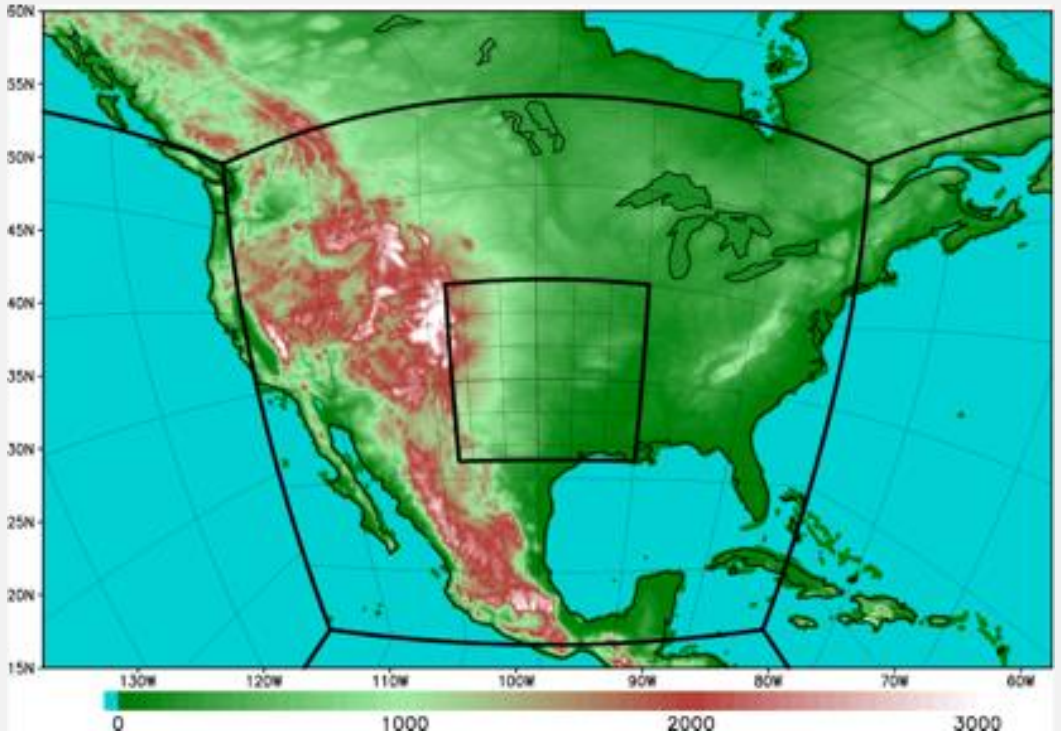


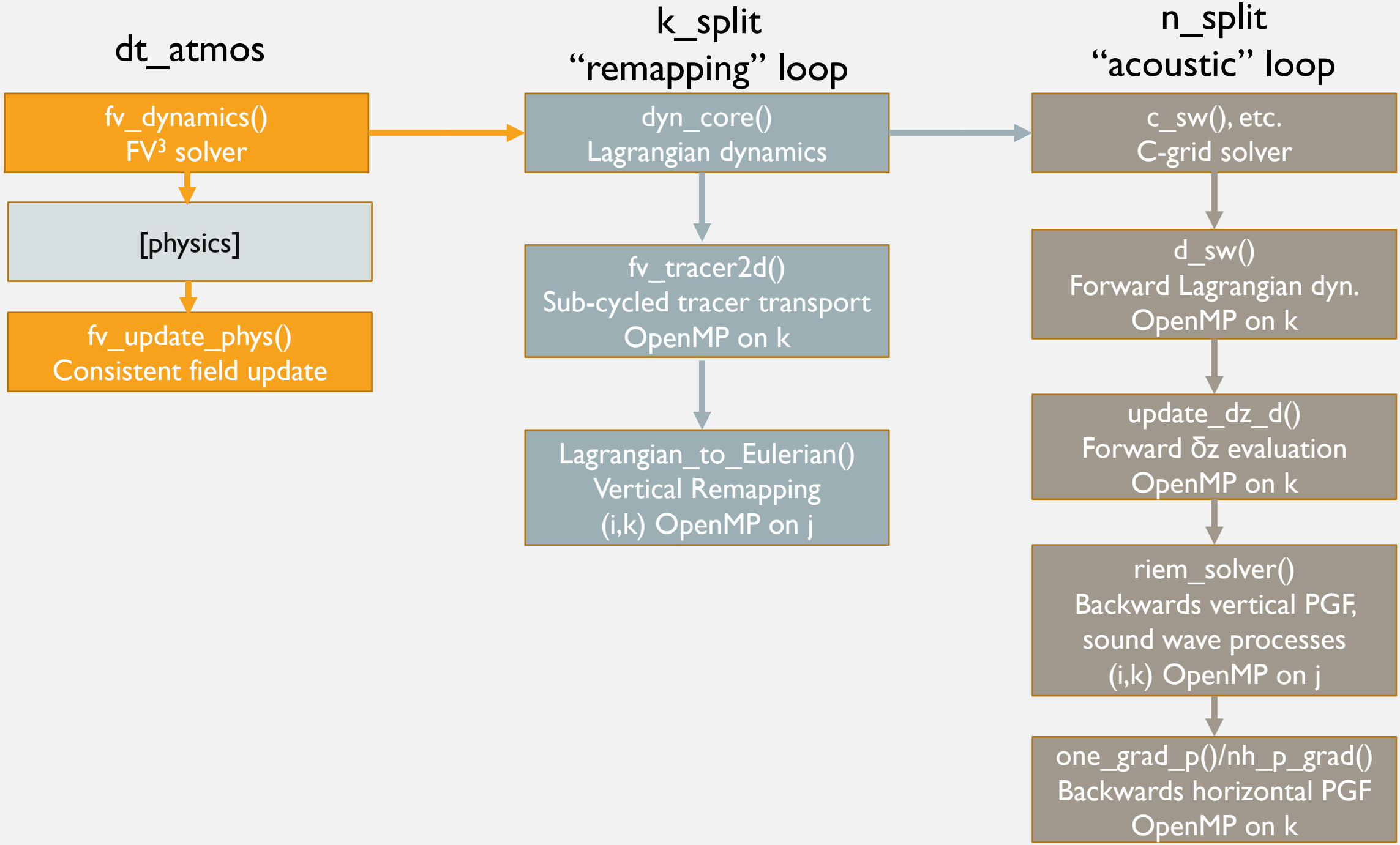


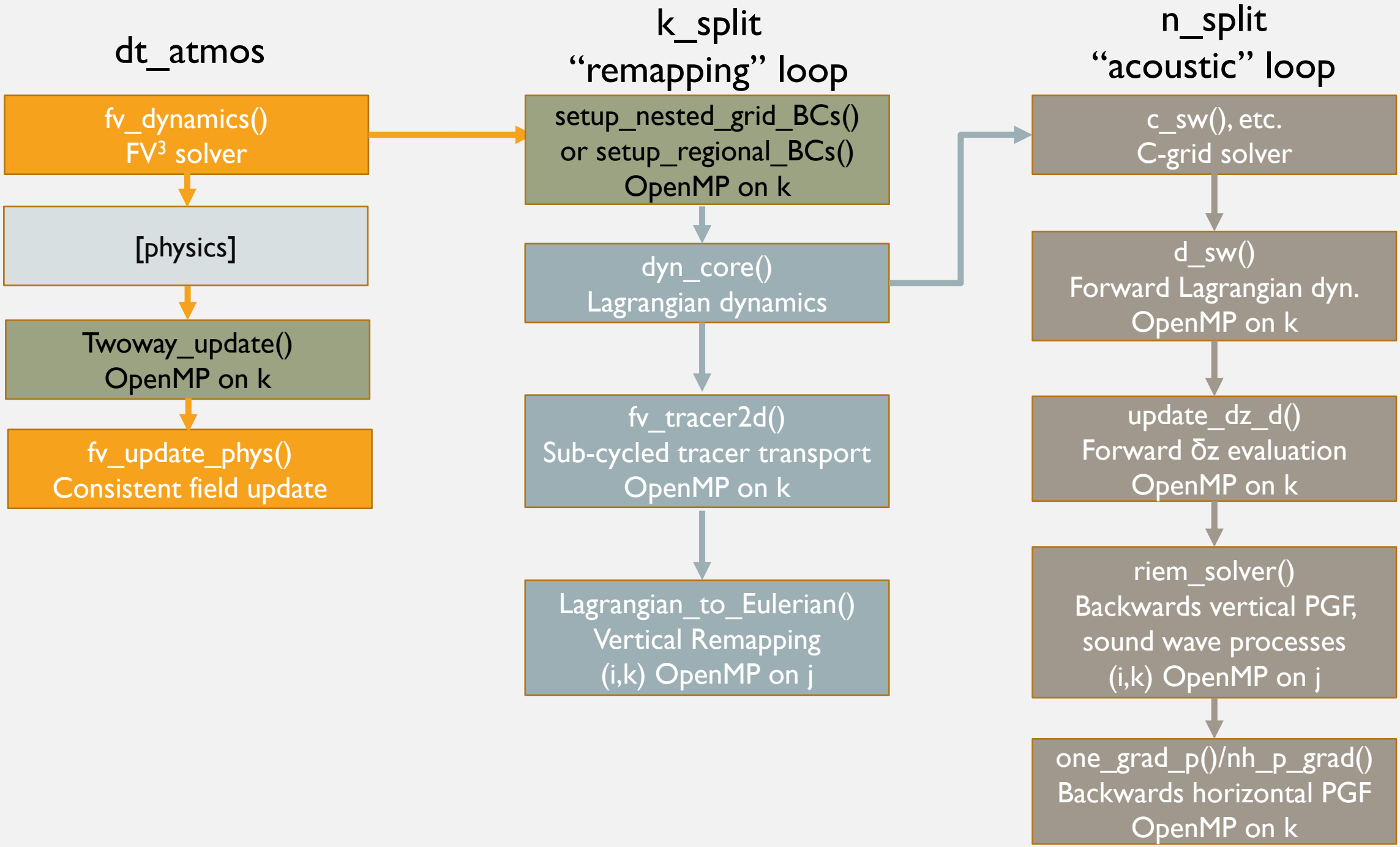
NESTING IMPLEMENTATION IN FV^3



- The nested grid is an additional tile beyond the six for the cubed sphere. There can exist many nests or nests within nests.
- Each nested grid has its own processor list and is integrated concurrently. This greatly aids load balancing between grids.




dt_atmos
fv_dynamics()
FV³ solver
[physics]
fv_update_phys()
Consistent field update
k_split
"remapping" loop
dyn_core()
Lagrangian dynamics
fv_tracer2d()
Sub-cycled tracer transport
OpenMP on k
Lagrangian_to_Eulerian()
Vertical Remapping
(i,k) OpenMP on j
n_split
"acoustic" loop
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
fv_dynamics()
FV³ solver
[physics]
Twoway_update()
OpenMP on k
fv_update_phys()
Consistent field update
k_split
"remapping" loop
setup_nested_grid_BCs()
or setup_regional_BCs()
OpenMP on k
dyn_core()
Lagrangian dynamics
fv_tracer2d()
Sub-cycled tracer transport
OpenMP on k
Lagrangian_to_Eulerian()
Vertical Remapping
(i,k) OpenMP on j
n_split
"acoustic" loop
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

NESTED-GRID WORKFLOW

- Nested grids can be generated online, although orography, land-surface information, and initial conditions have to be generated.
- Standard NCEP tools allow offline generation of grid information and ICs
- Each grid gets its own namelist. Any runtime parameter can be customized for the individual grids
- GFDL fregrid can perform conservative remapping of nested-grid output onto a regional regular latitude-longitude grid. It is also possible to use common software (Python, NCL, IDL, dbrowse, GrADS (?), etc) to plot the native nested grid.

NESTING: NAMELIST OPTIONS

- twowaynest: Enables two-way nesting
- parent_tile: Number of tile on parent domain (if a cubed-sphere grid); set to 6 if using a stretched global grid
- parent_grid_num: number of the parent grid; currently only 1 is supported.
- refinement: Refinement ratio of nested grid. Can be any integer; 2–4 are best.
- ioffset and joffset: Indices of the first ("southwest") cell on the coarse grid that has a nested grid within it.
- nestupdate: Chooses between several different options for two-way updating. Set to 7, which updates u, v, w, and T but not tracers. (Other options not supported.)

NESTED GRID: NAMELIST OPTIONS

- If nesting is used then **nest_nml** must be present and identical in the input.nml for every grid.
- ngrids (sometimes called ntiles): Total number of grids. Currently limited to 2.
- nest_pes: 1D array of the number of processors (MPI ranks) assigned to each grid, in order. The sum must equal the total number of processors assigned to the entire atmosphere model.
- p_split: number of times to call the nested-grid BCs and two-way interaction per physics call. If > 1, reduce k_split accordingly so the acoustic and remapping timesteps remain constant.

CREATING A NESTED GRID

In driver_grid.csh the nested grid is defined on the supergrid, the doubled grid used internally by the FMS exchange grid. The current form of the stretched grid also rotates the cube so the that north and south are flipped.

BEST PRACTICES FOR CHOOSING A SINGLE STATIC NESTED GRID

- For a coarse cN grid and a refined-by-R nest:
  - Choose the size (npx_g2 – 1) x (npy_g2 – 1) of your nest. This should be no more than R*(N – 2).
  - Select target_lat and target_lon to be the center of your nest.
  - You will choose your nest to be centered within the parent tile. This makes the selection of parameters easy.
- After setting up the grid, make sure that your grid parameters in your runscript's namelist matches those used to create the grid.
  - Coarse grid: target_lat, target_lon, stretch_fac (also needed for a stretched grid)
  - Nested grid: npx, npy, refinement, ioffset, joffset

CENTERED NESTED GRID FORMULAS

- Defining N_{nx}= (npx_g2 – 1)/R and N_{ny} = (npy_g2 – 1):
  - ioffset = (N – N_{nx}) / 2 + 1
  - joffset = (N – N_{nx}) / 2 + 1
- Supergrid parameters for driver_grid.csh :
  - istart_nest = ioffset * 2 – 1
  - jstart_nest = joffset * 2 – 1
  - iend_nest = istart_nest + N_{nx} * 2 – 1
  - jend_nest = jstart_nest + N_{ny} * 2 – 1

NESTING AND TOPOGRAPHY

- The pre-processing currently does not support orographic filtering on the nested grid. However the on-line filtering during the initialization does work.
- The simpler, older topographic filtering can be enabled by setting mountain = .true., n_zs_filter = 1, and nord_zs_filter = 4. (The more comprehensive filtering available in the filter step is an experimental option.)
  - Be sure to disable these options if you want to restart the nested grid.
- Nested and coarse grid orography are blended near the boundary of the nest. However steep orography along the nest boundary may lead to instability. Think carefully about the location of your nest boundary.