Novel Grid Capabilities in GFDL's FV3

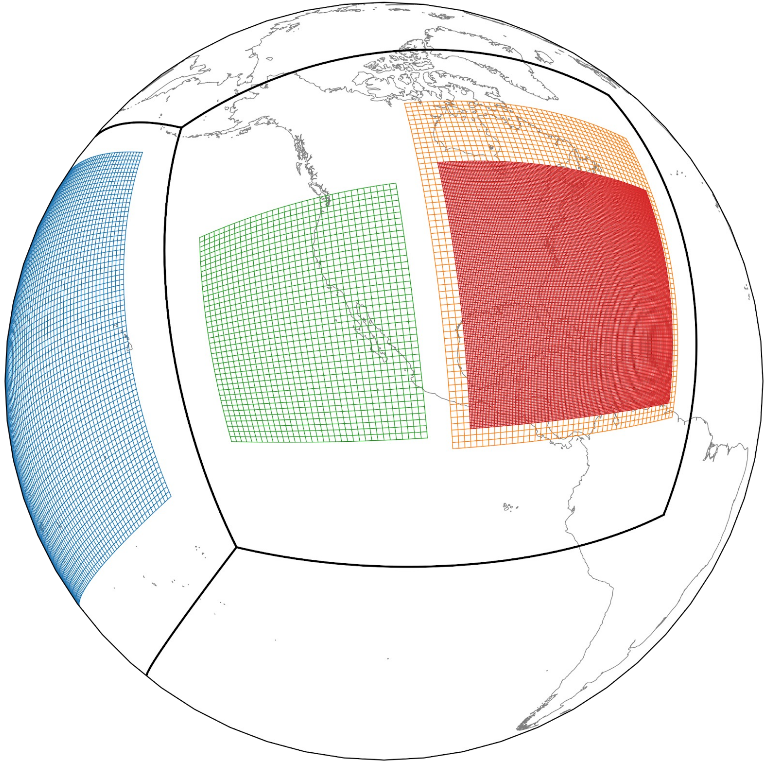
Joseph Mouallem

Thanks to Lucas Harris, Rusty Benson, the FV3 and MSD teams

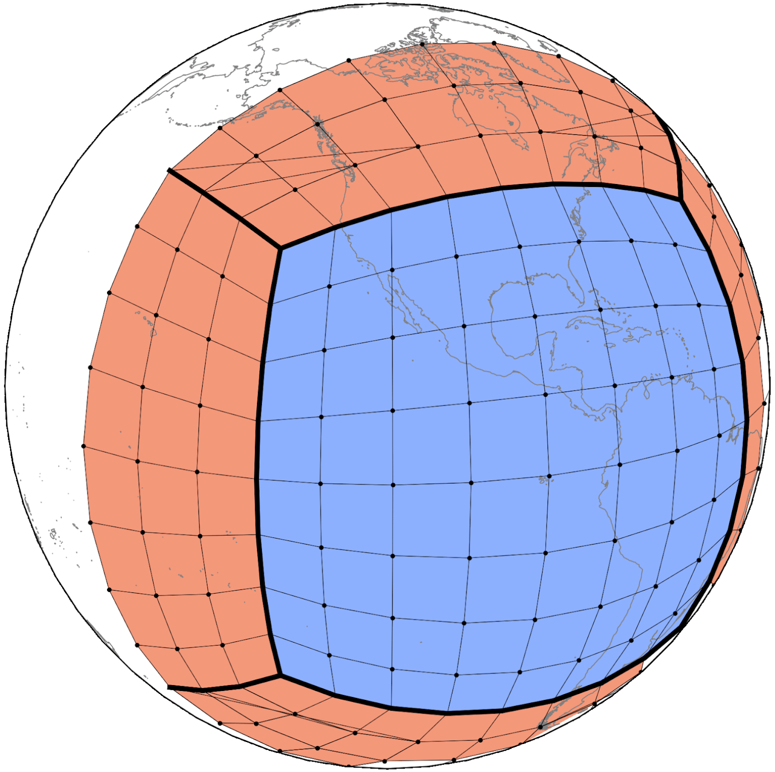
Cooperative Institute for Modeling the Earth System, Princeton University, Princeton, NJ, USA,
NOAA/Geophysical Fluid Dynamics Laboratory, Princeton, NJ, USA

November 14, 2024

Novel grid capabilities in FV3

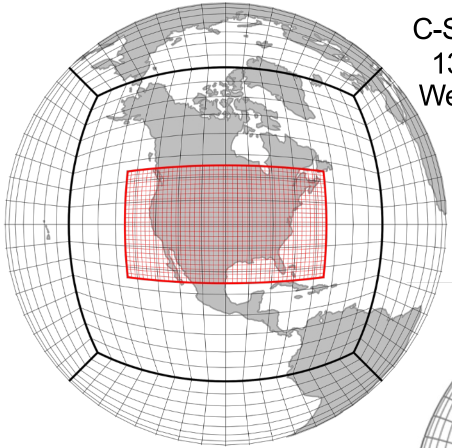


Grid nesting

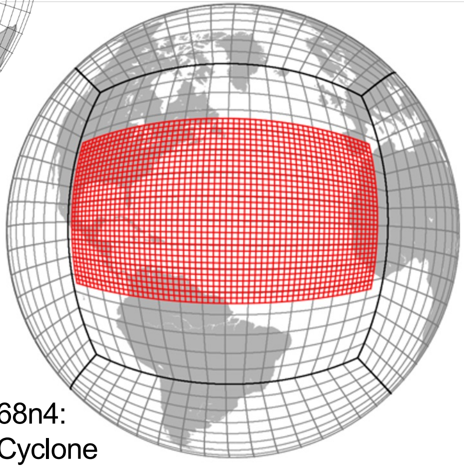


Duo-Grid

FV3 variable resolution techniques

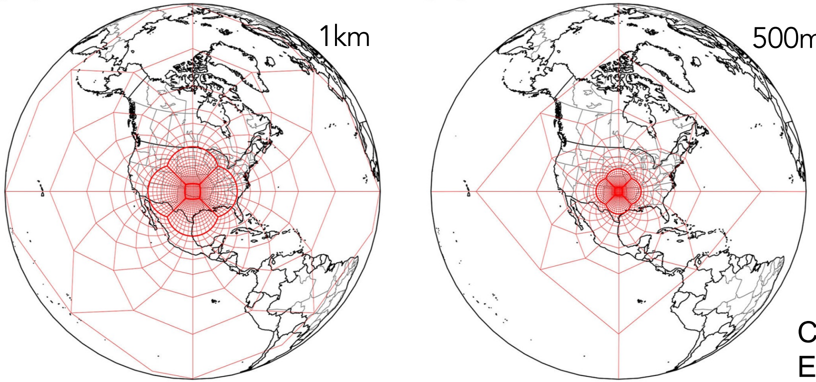


C-SHiELD: C768n5:
13-2.5km Severe
Weather Prediction

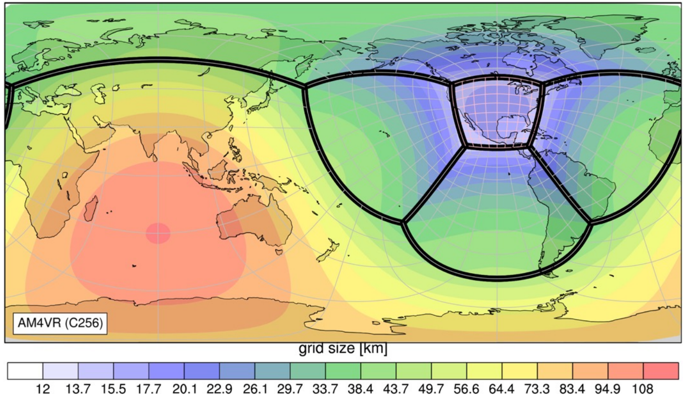


T-SHiELD: C768n4:
13-3km Tropical Cyclone

Grid nesting



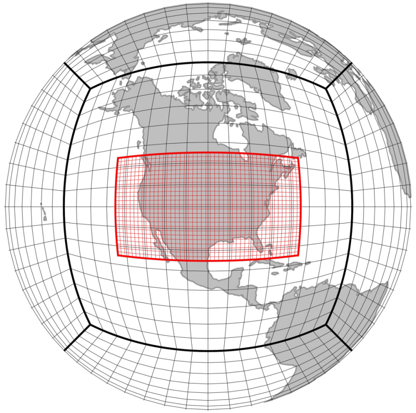
Cheng et al., ESS, 2024

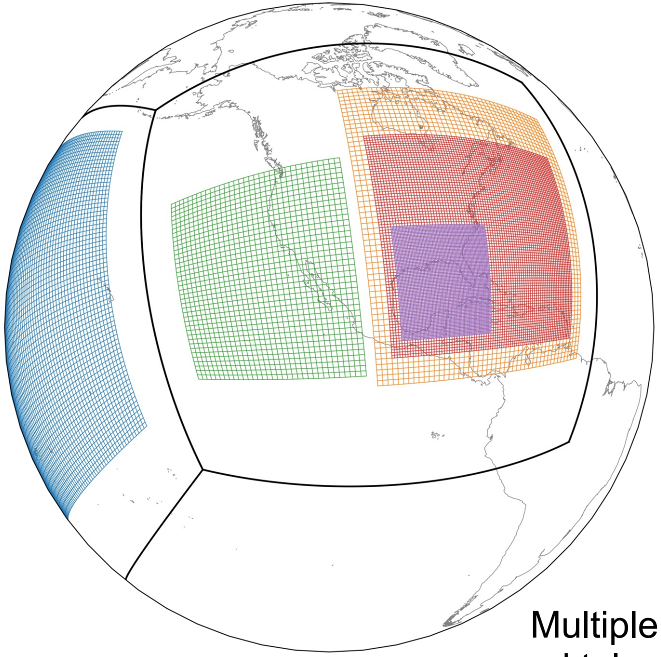


Lin et al., JAMES, 2024

Grid stretching

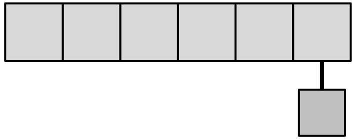
Grid nesting:





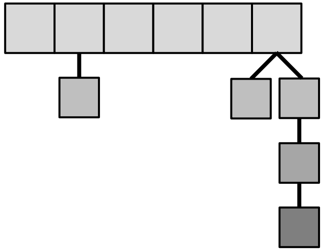
Single nest

Harris&Lin 2013, MWR



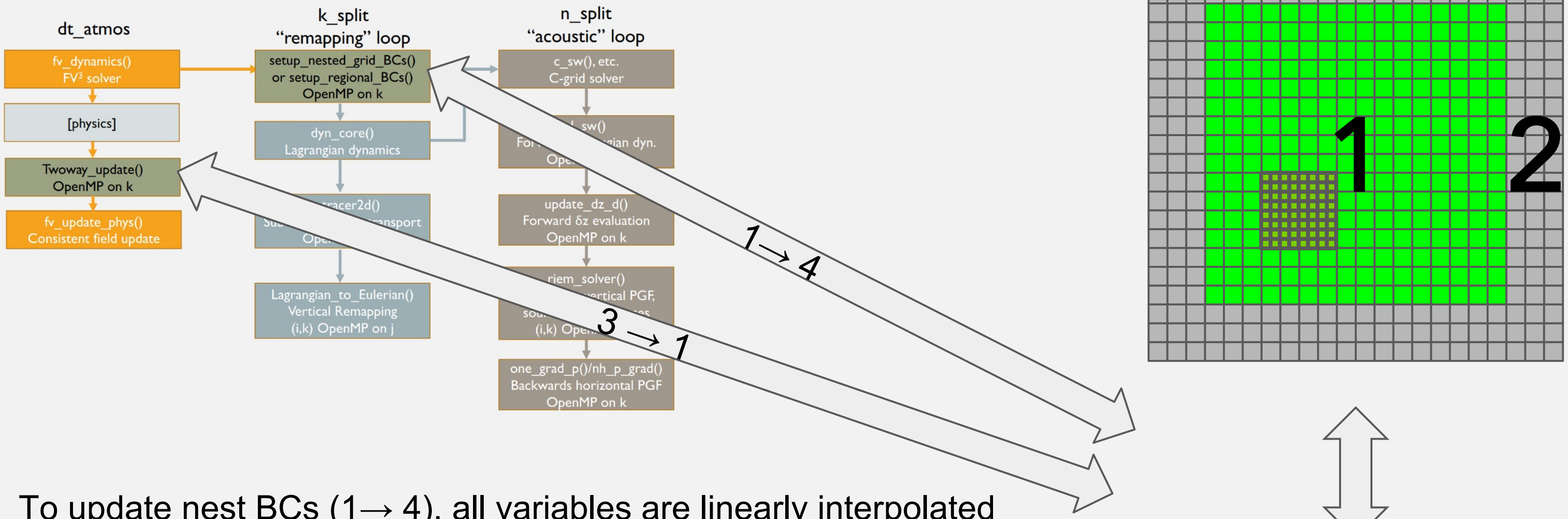
Multiple same level
and telescoping nests

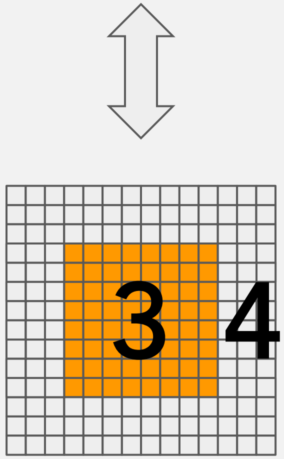
Mouallem et al. 2022, GMD



As many as you want!

Grid nesting in FV3:





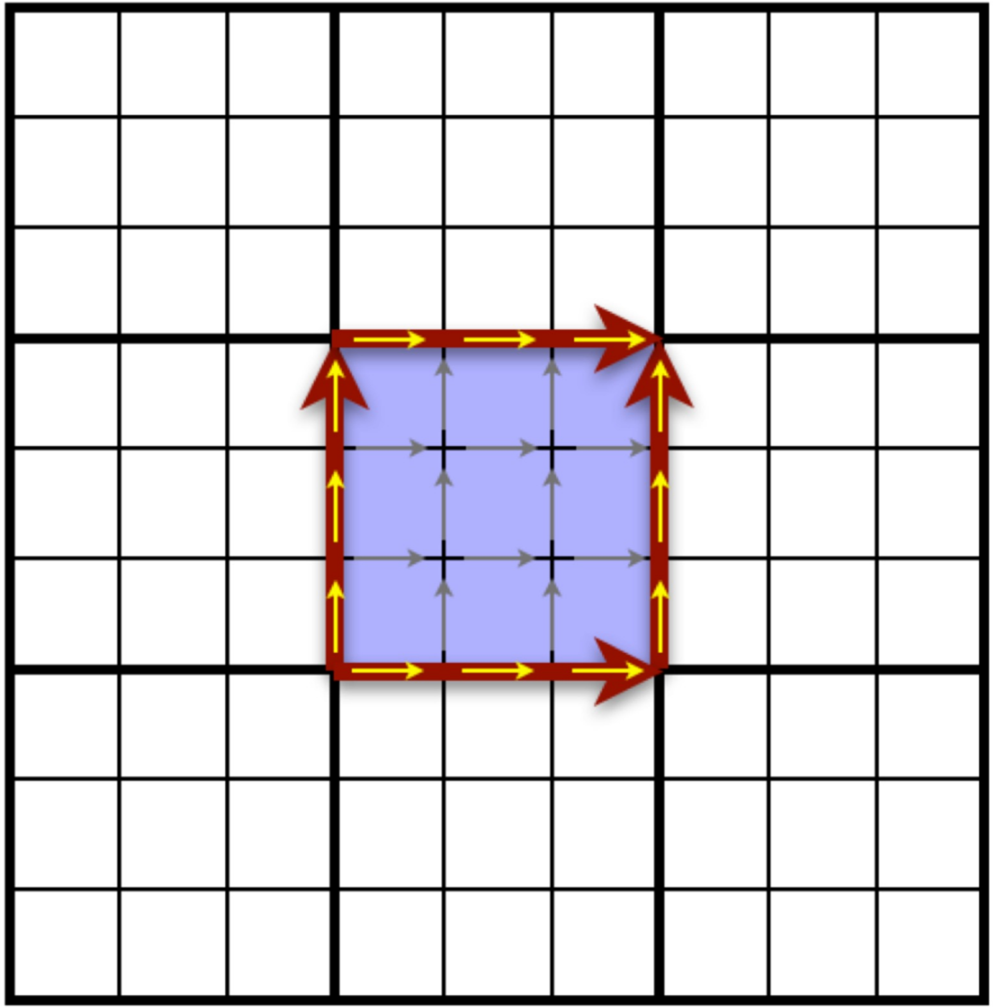
- To update nest BCs (1→ 4), <u>all variables</u> are linearly interpolated in space
- Concurrent nesting: Parent and nest run simultaneously on different sets of processors.
- Nest BCs are also linearly extrapolated in time every acoustic timestep (n_split) then updated from the coarse grid at the end of the full timestep before the vertical remapping timestep (k_split).
- Only the temperature and the three wind components are used for the twoway updates. Therefore, there is no violation of mass conservation during this process on the coarse grid.

Two way updates:

- For cell-mean scalars, the value in the shaded coarse-grid cell (heavy lines) is replaced by the area-weighted average of the values on the coinciding nested-grid cells (thin lines).

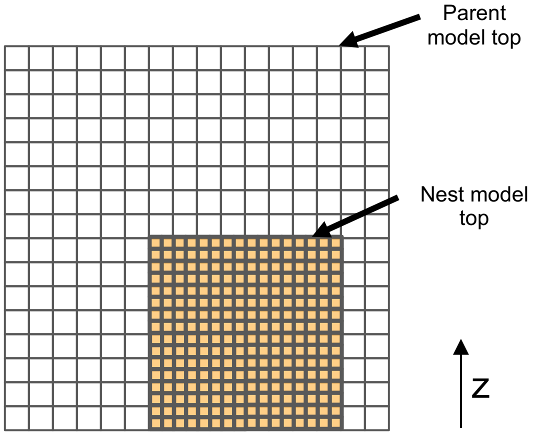
- The winds tangential to this coarse-grid cell (red arrows) are updated using the length-weighted average of coinciding nested-grid cell boundaries (yellow arrows). This conserves vorticity.

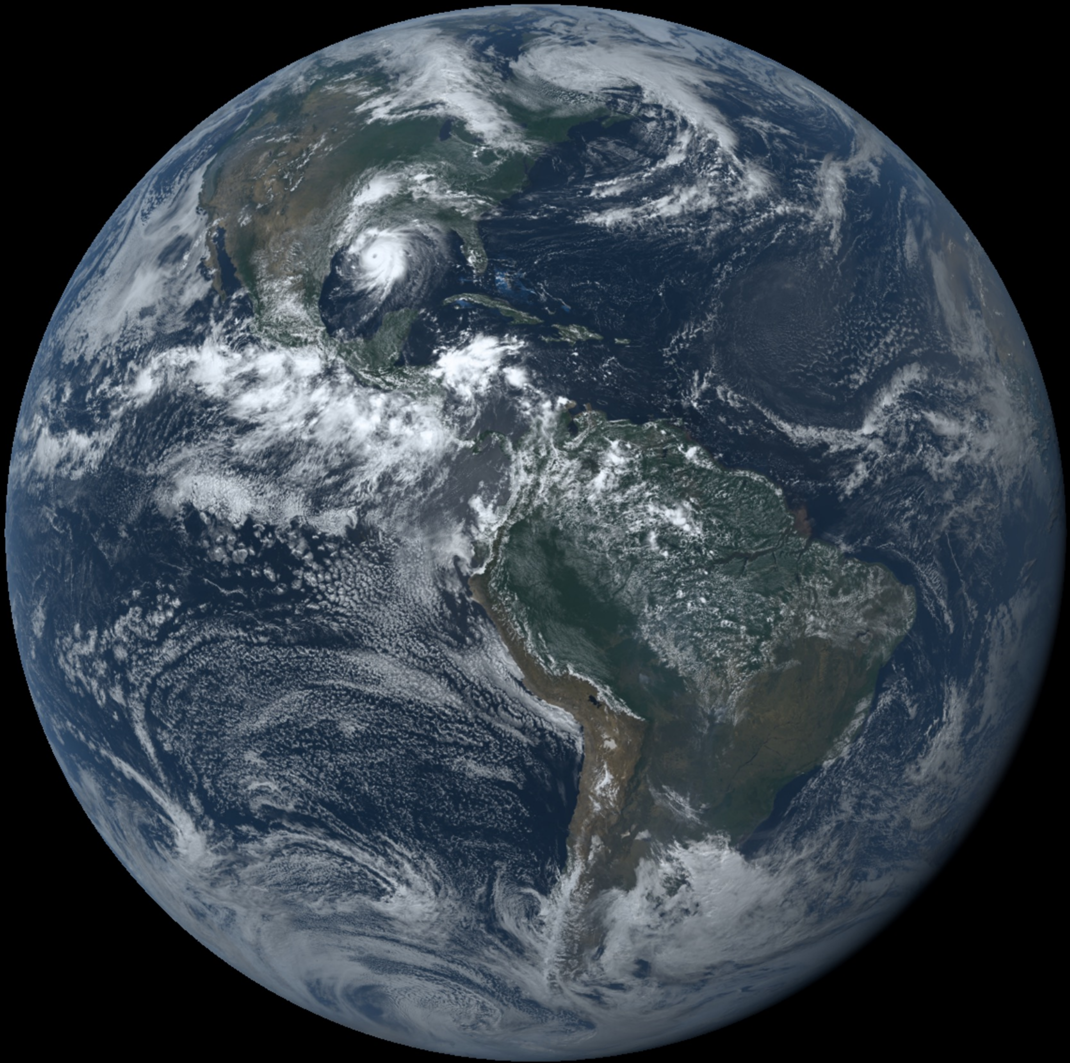
Note: Cubed-sphere grid cells are not squares.
Could be applied to any non-orthogonal quadrilateral grid

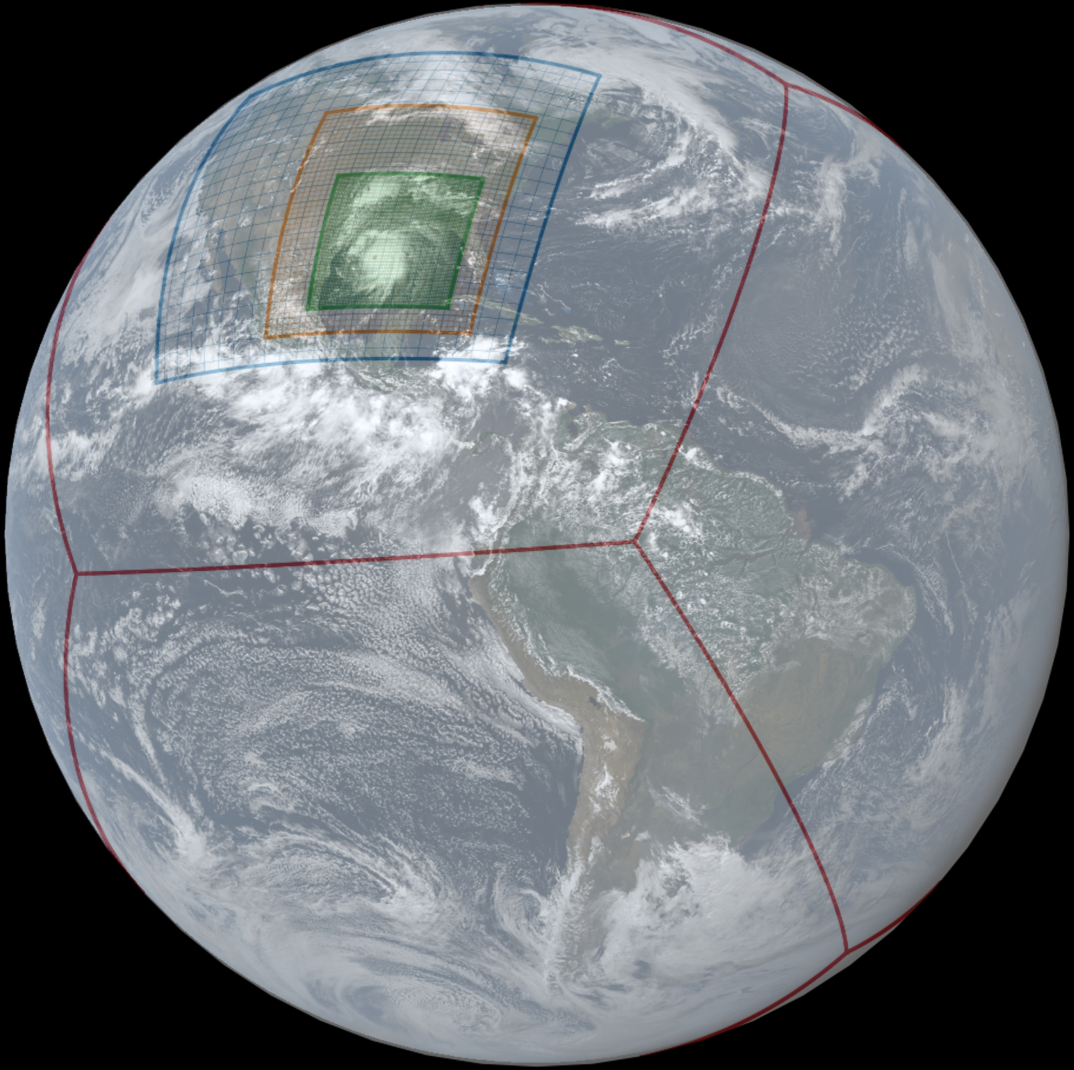


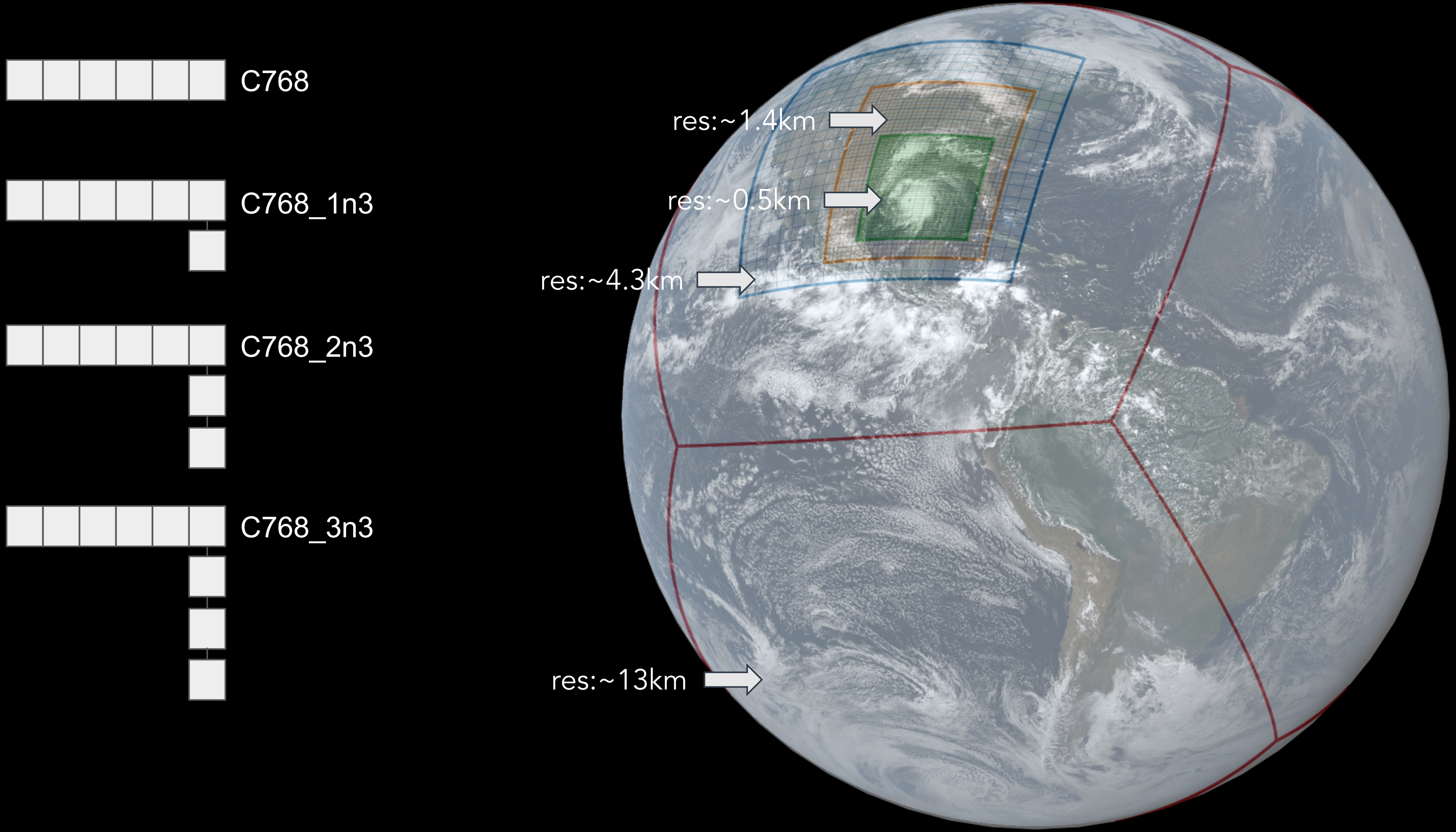
Grid nesting in FV3:

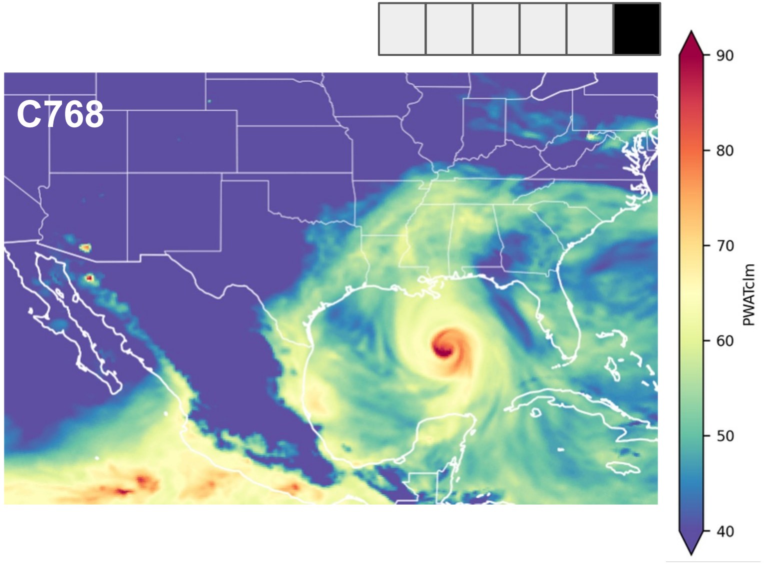
- Vertical nesting is allowed => the nested grid can have a lower model top. (T-SHiELD: 63 parent layers, top at 64 Pa; 75 nested layers, top at 200 Pa)
- Nests can not cross a cubed sphere edge or corner. FMS supports an edge crossing, but it is not implemented in FV3 yet.
- Each nested grid runs on a specific list of processors, allowing concurrent time stepping which eases the computational load of each grid.
- Each nested grid has its own input name list, and can be configured differently (timestep, parametrization, …).
- Two-way updates can be turned on and off on each nested grid.
- No extra relaxation at the nest boundary conditions.
- Nests could be implemented in a stretched parent grid.
- **Bit-for-bit Reproducibility** is conserved when changing processors layouts on the parent and nested grids.

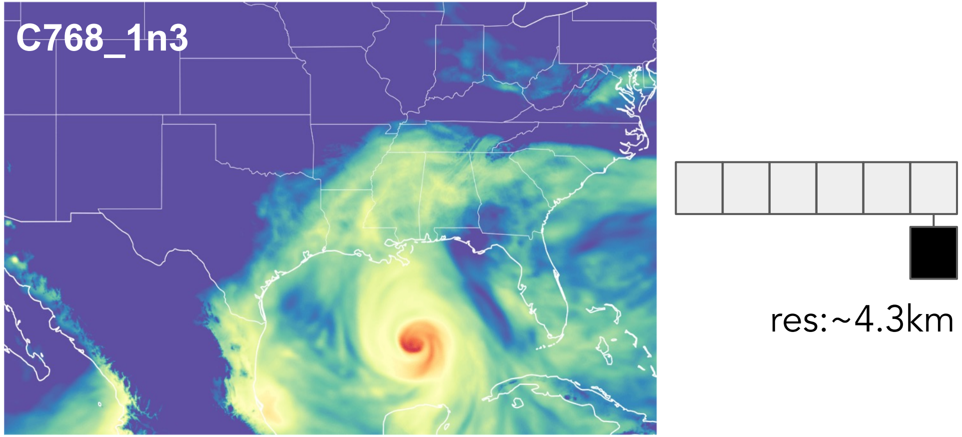


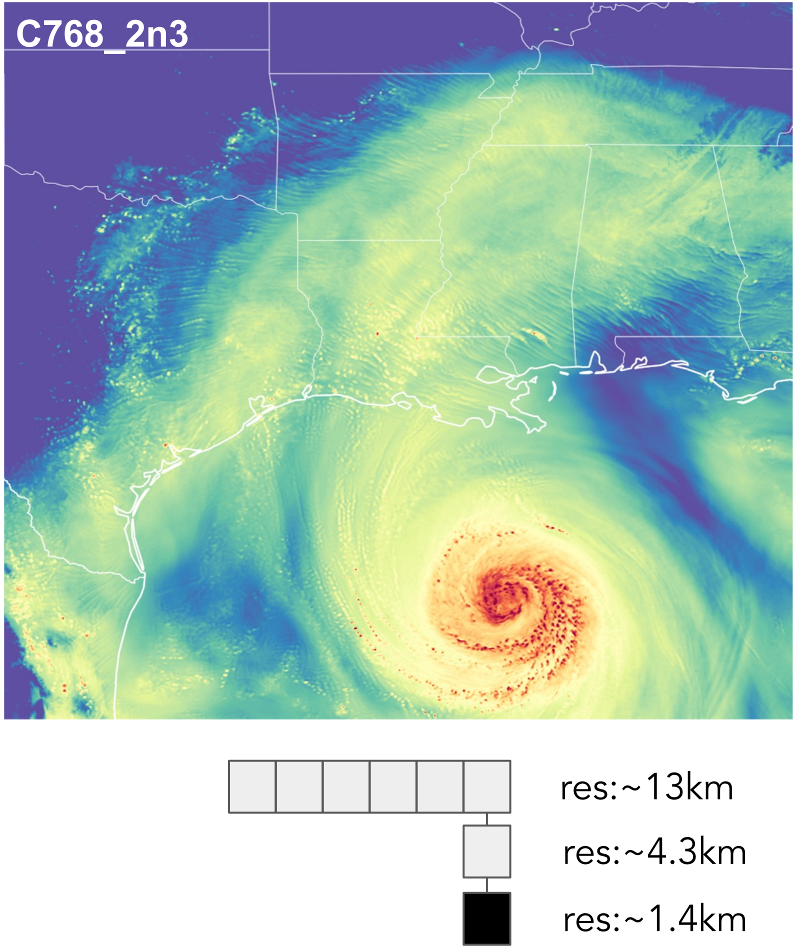





C768
C768_1n3
C768_2n3
C768_3n3
res:~1.4km
res:~0.5km
res:~4.3km
res:~13km

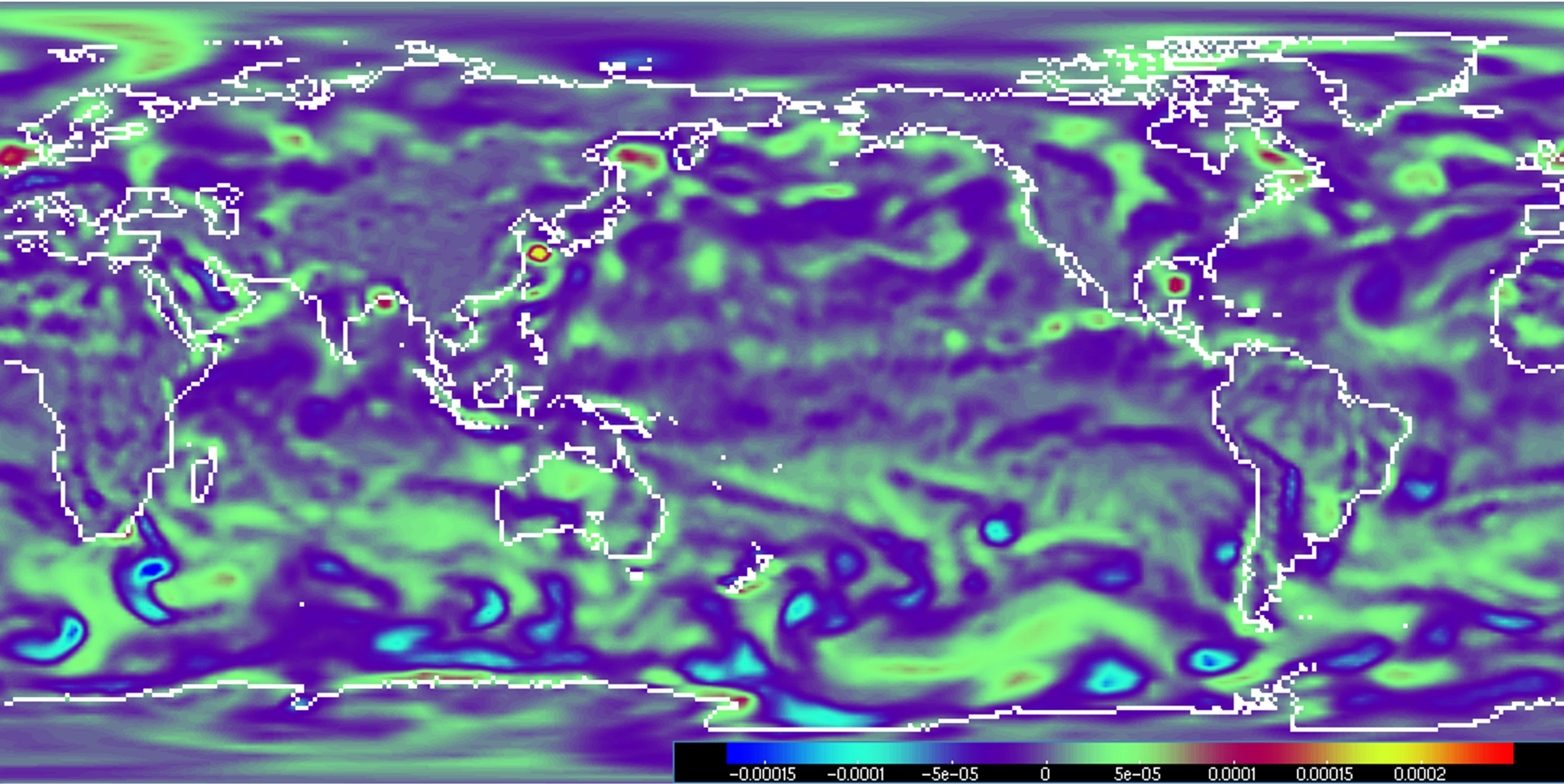




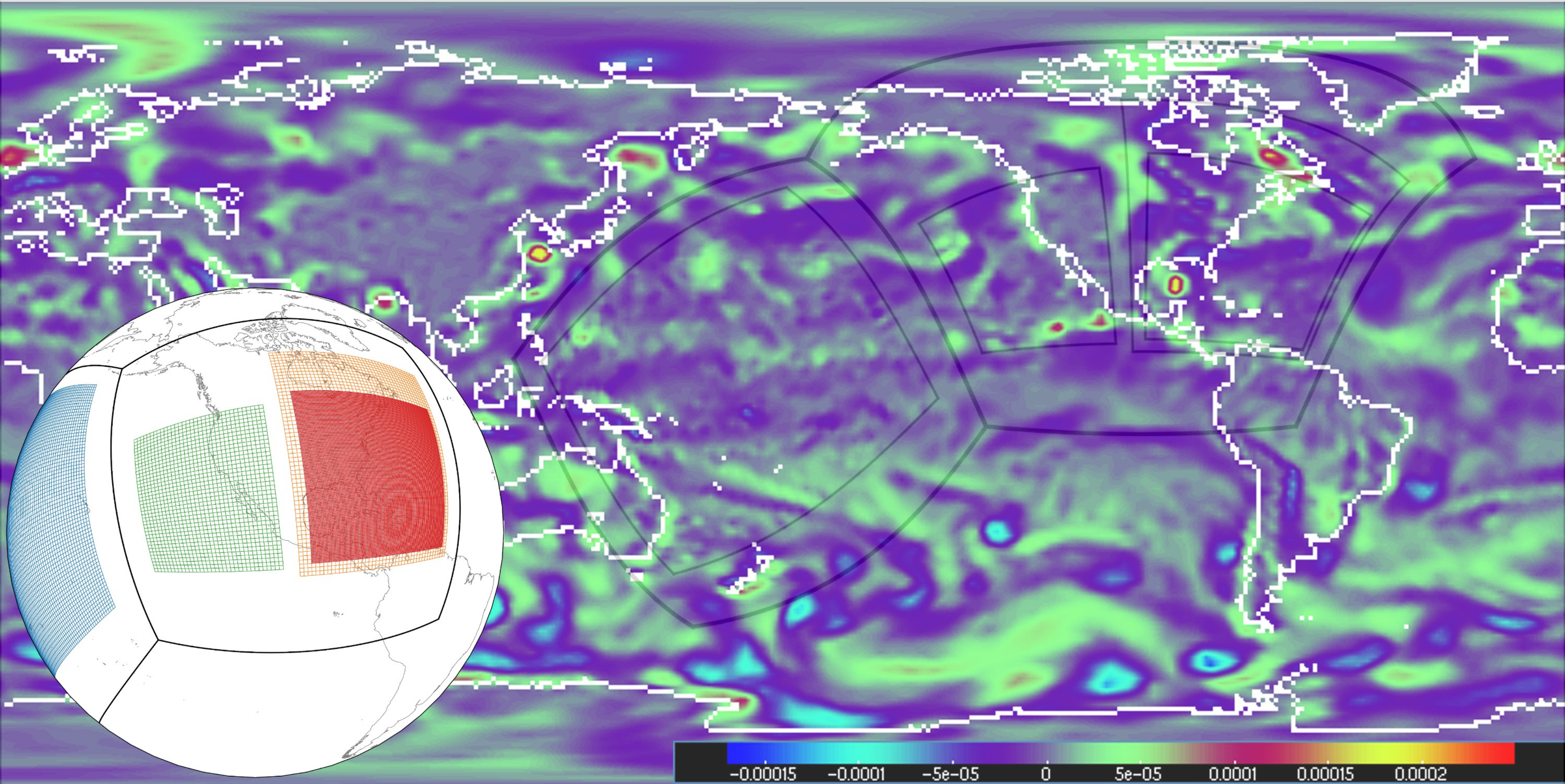


Work in progress!

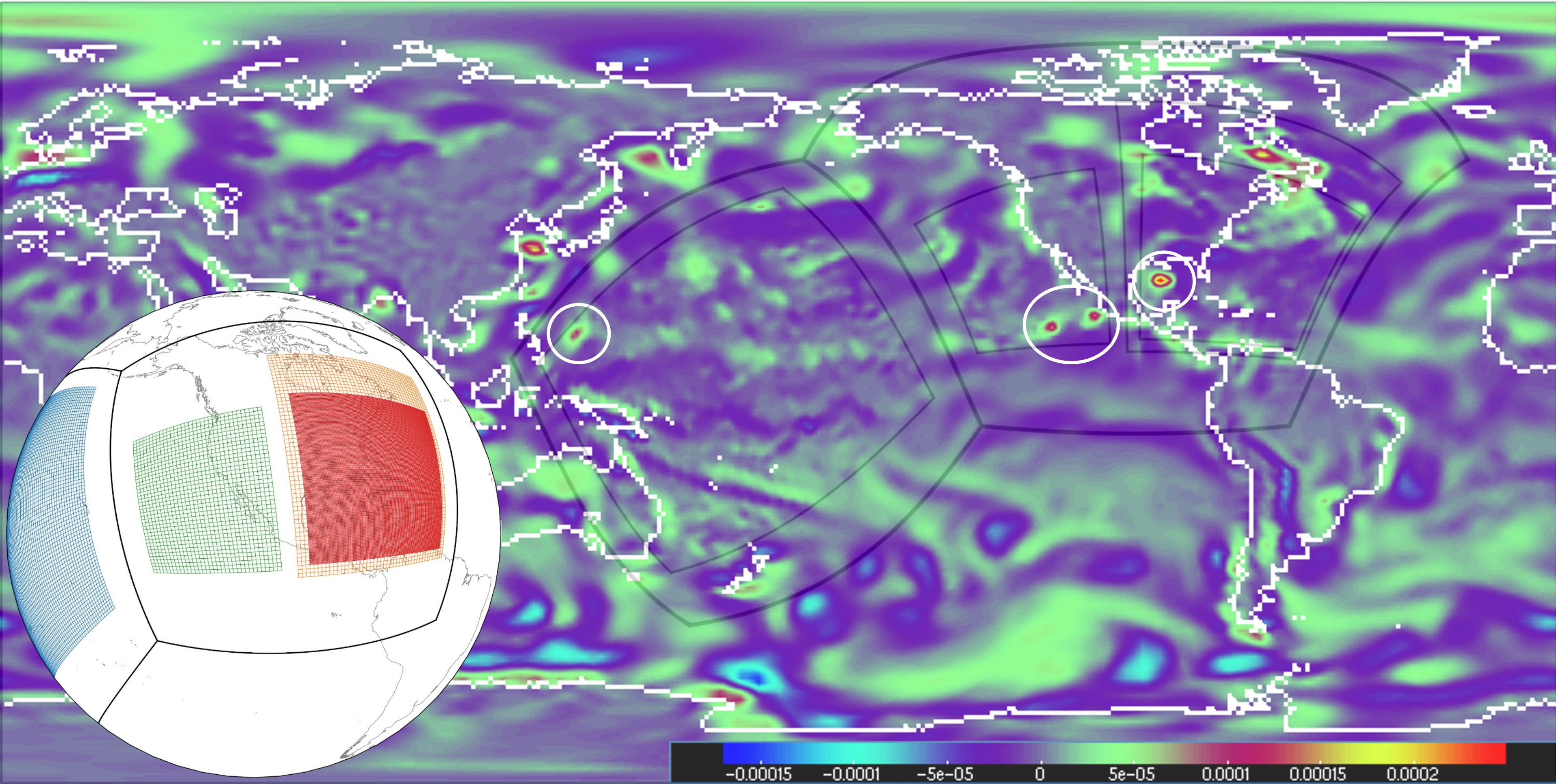
Vort850mb -C48- 3days



Top level Vort850mb -C48_4n4- 3days (Res: 200-50-12)

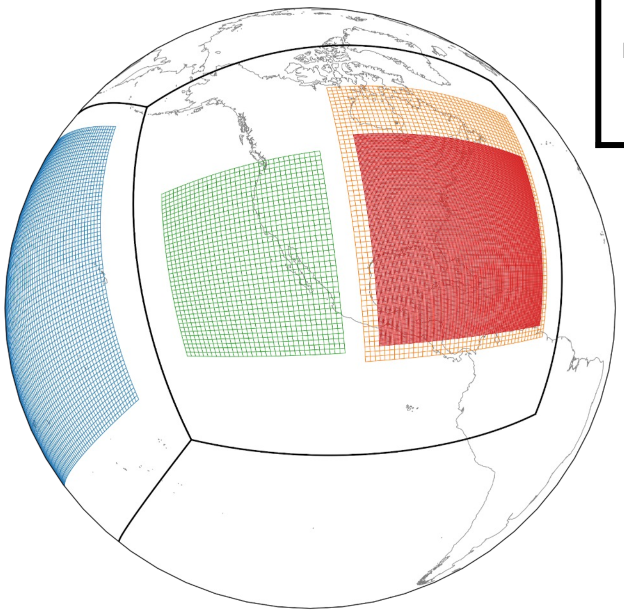


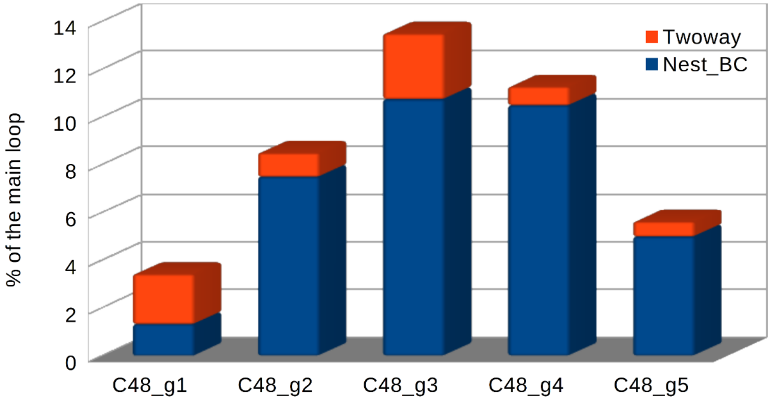
Top level Vort850mb -C48_4n4- 3days (Res: 200-50-12)



Code Timing and Performance

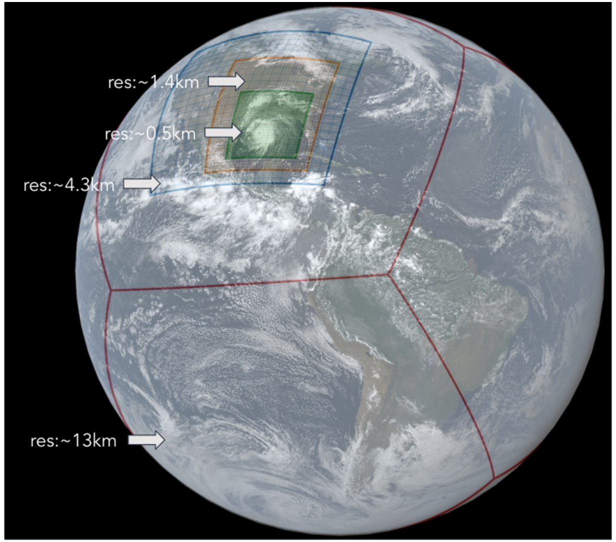
Case	Description	Resolution	# of cores	Memory output files (GB)	Three days simulation time (s)
C48	Global uniform resolution: One grid: six tiles (48x48 per tile)	200km	6x6x6=**216**	0.04	84
C768	Global uniform resolution: One grid: six tiles (768x768 per tile)	13km	30x30x6=**5400**	16	975
C48 + Four Nests	Global uniform resolution: Grid1: six tiles (48x48 per tile) Nests: Grid2: one tile (145x145 Level1) Grid3: one tile (89x117 Level1) Grid4: one tile (69x81 Level1) Grid5: one tile (317x337 Level2)	200km 50km 50km 50km 12.5km	6x6x6=216 12x12=144 8x10=80 6x7=42 30x30=900 Total=**1382**	0.5	415

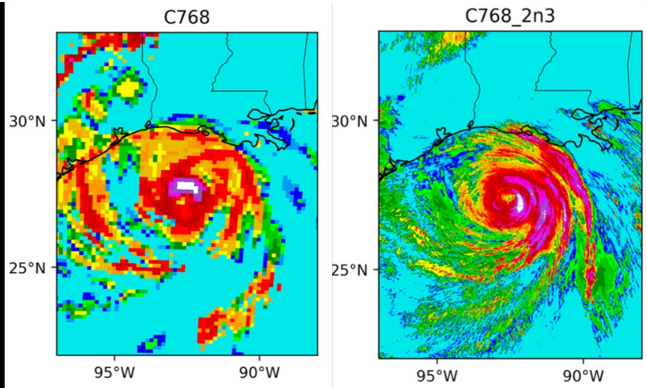




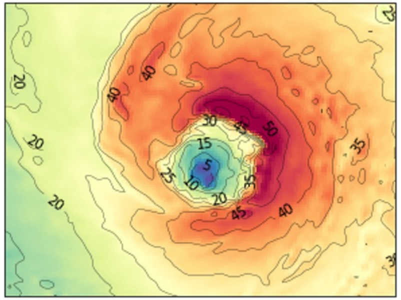
Time spent on the nest BC and fine to coarse twoway updates relative to the main loop time. Basically, this is the overall time spent on grid/grid communication.

Multiple same level and telescoping nests

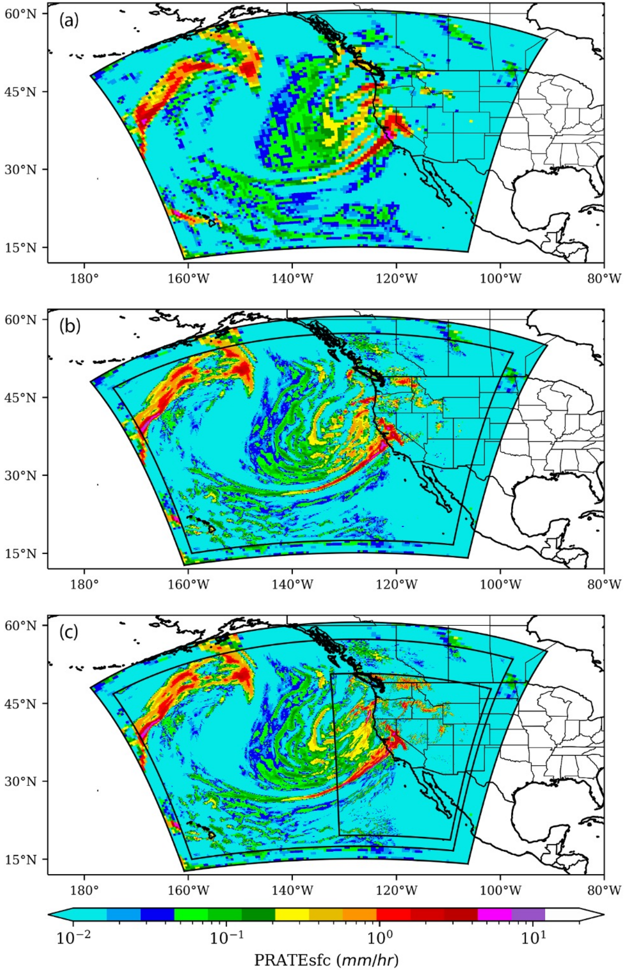




Global-nest rainfall at 13/1km



Laura at 1km taken from 13/4/1km



AR 2021 Regional-nest precipitation 50/17/6km

Multiple same-level and telescoping nesting in GFDL's dynamical core

(Mouallem et al. 2022, GMD)

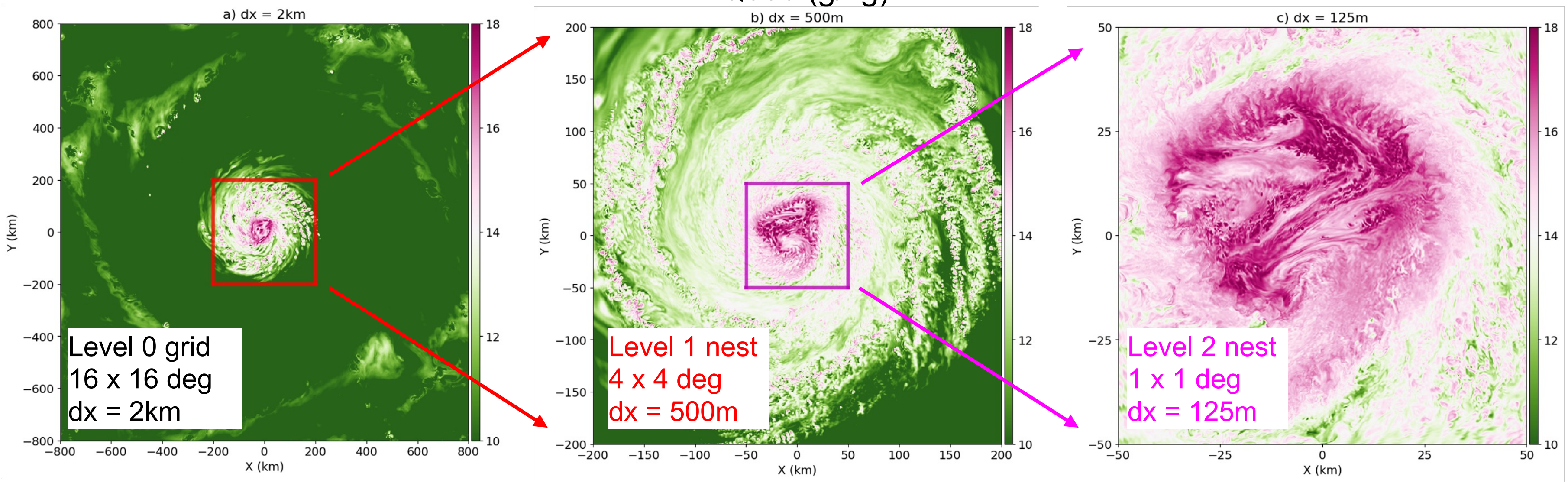
HSUP-funded project transferred to HAFS.

HAFS became operational on June 28, 2023!!

Telescoping nests covering the entire TC

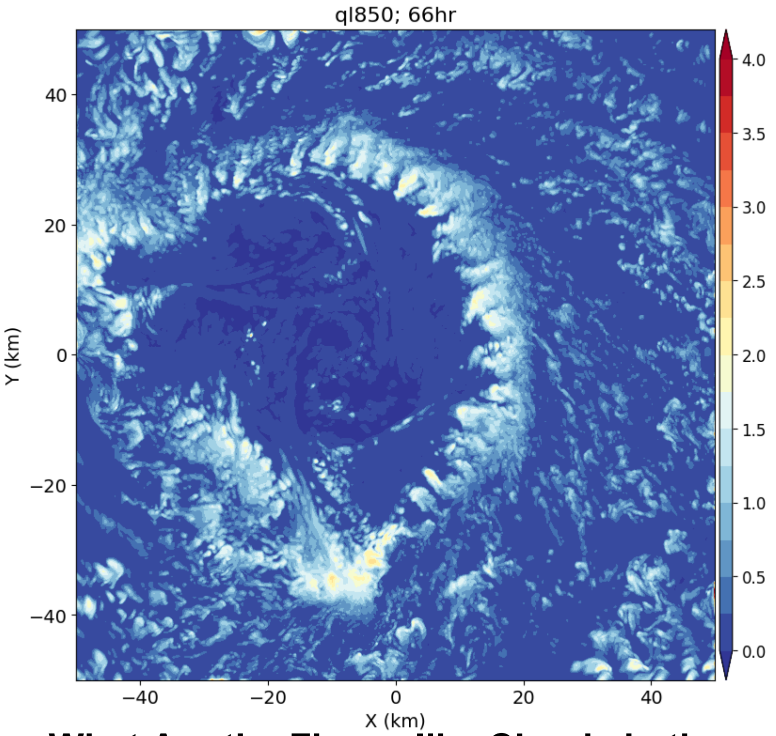
- Two-level nesting: 2km -> 500m -> 125m (same vertical levels)
- Idealized TC that evolves freely on *f*-plane
- 96-hour simulation that covers entire intensification period

Q850 (g/kg)



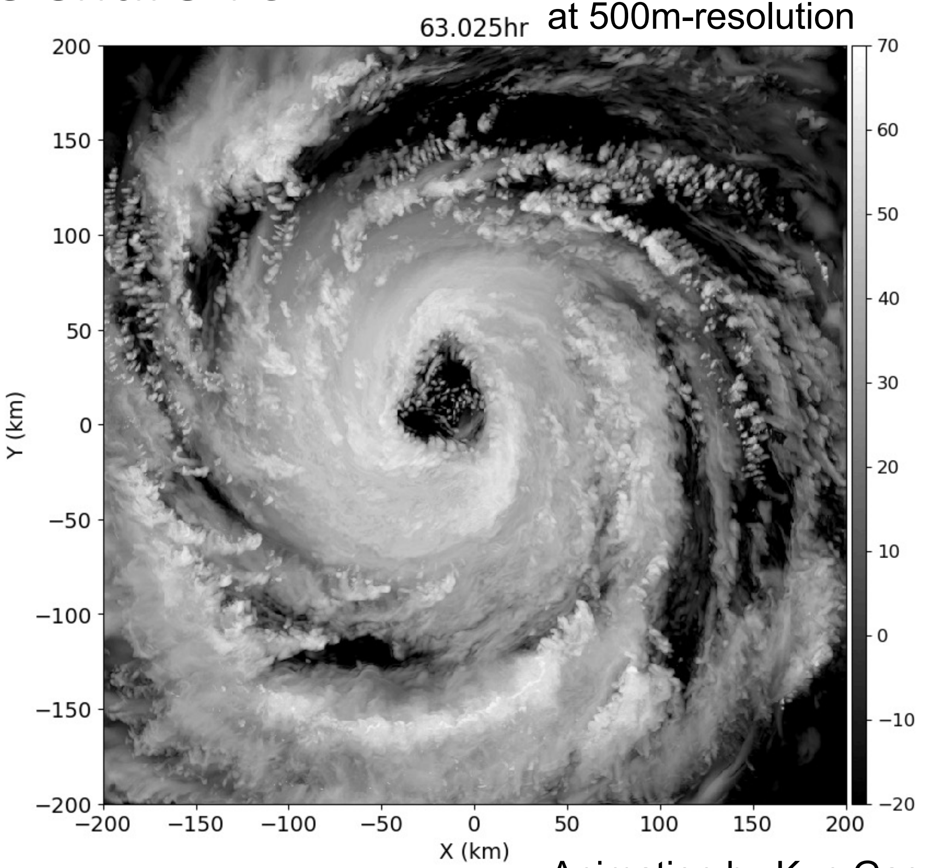
Courtesy Kun Gao

Telescoping nests covering the entire TC



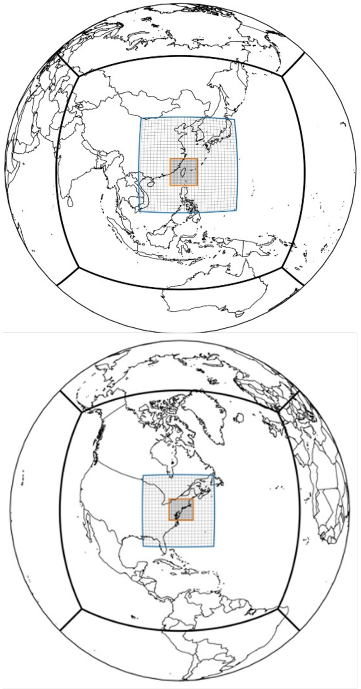
What Are the Finger-like Clouds in the Hurricane Inner-core Region?
Gao et al, 2024 GRL

Simulated reflectivity at 500m-resolution

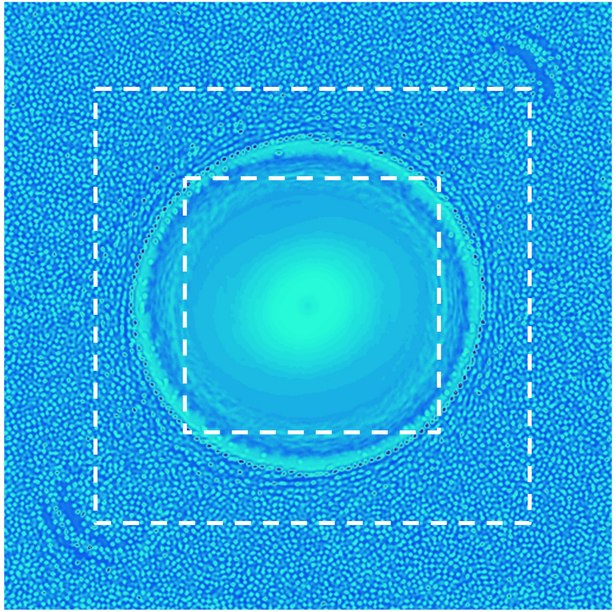


Animation by Kun Gao

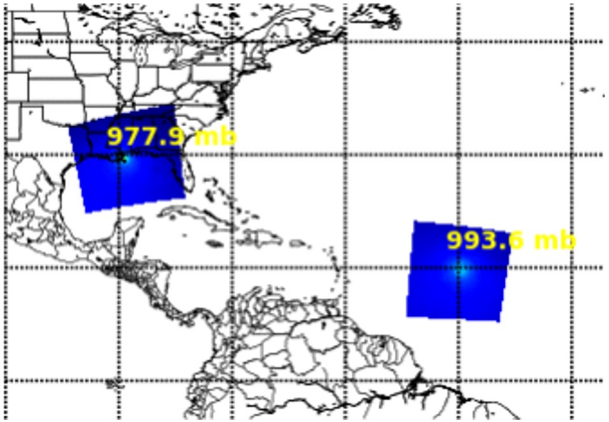
Ongoing Nesting projects



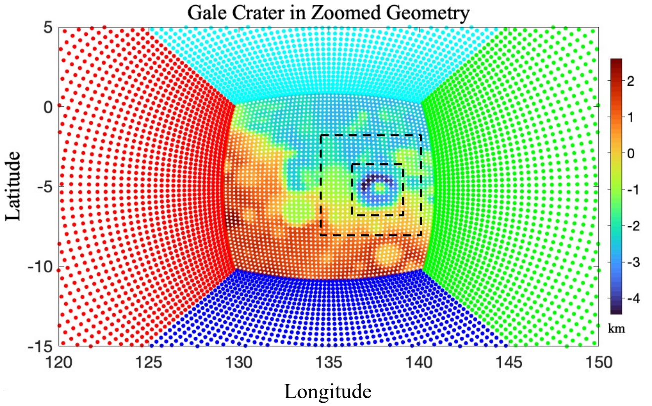
1.4 km Tele-SHiELD for hyper-local impacts
Courtesy Jan-Huey Chen (GFDL)



Telescoping idealized TC in DP domains (down to 100m)
With Kun Gao (GFDL)



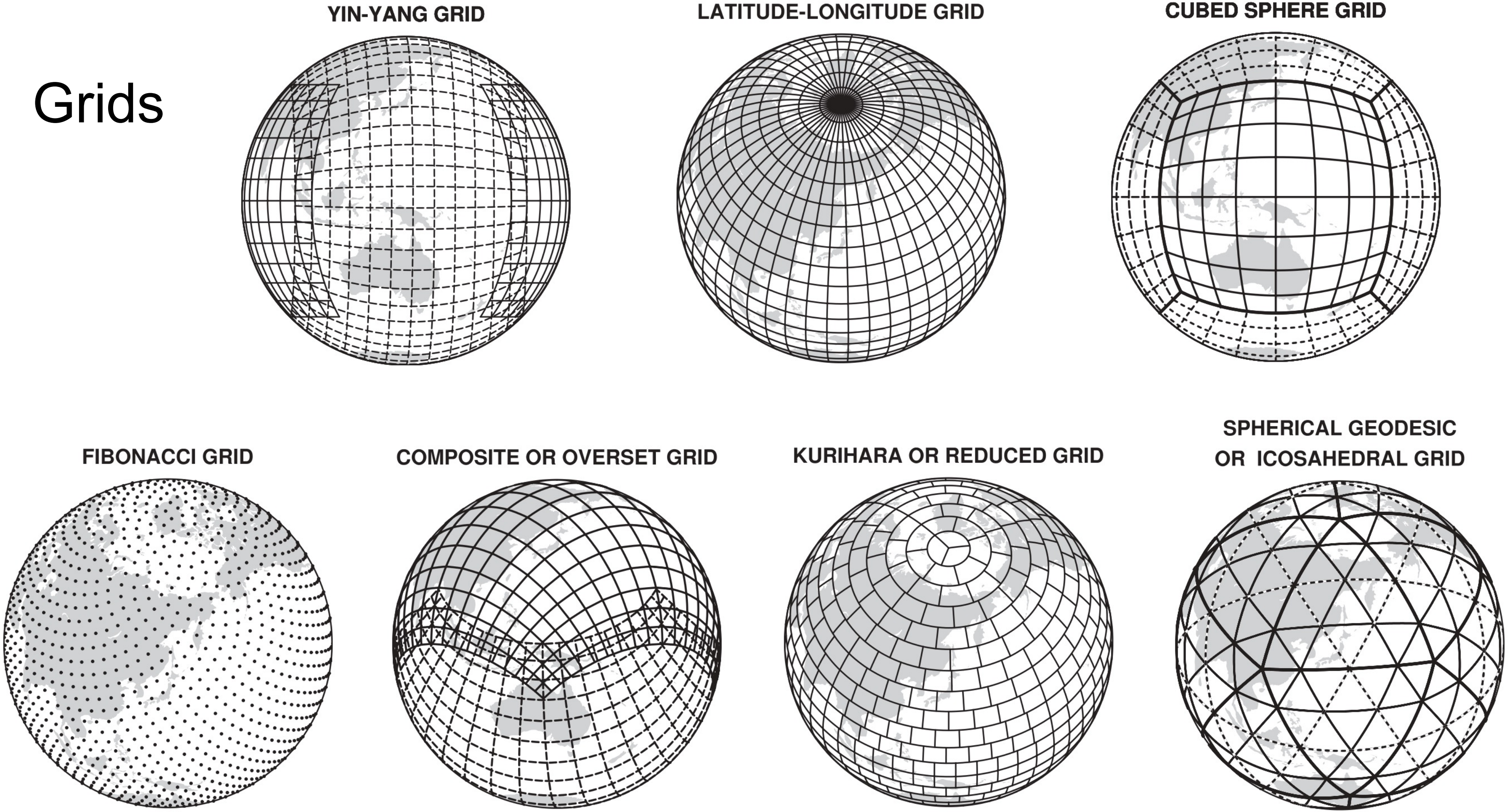
Multiple moving nests
Courtesy Bill Ramstrom (AOML)



Mars craters/landing sites
Courtesy John Wilson (MCMC/NASA)

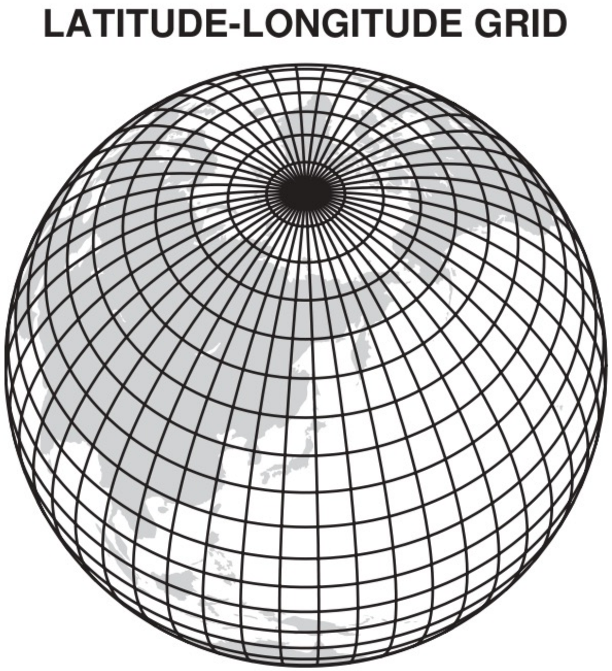
Part 2:
The Duo-Grid

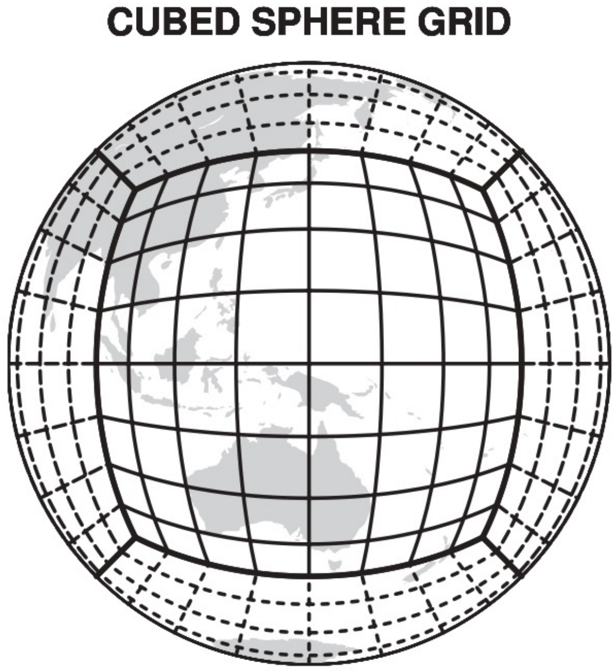
Grids



Williamson 2007

Grids





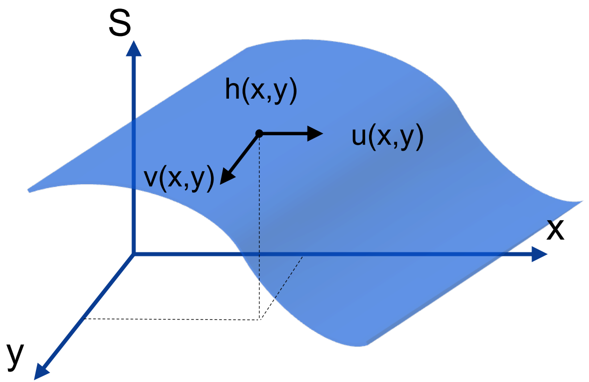
- Extreme grid aspect ratio => restriction on CFL
- Two polar singularities preventing effective 2D decomposition

Not suitable for ultra-high resolution modeling

- Quasi-uniform resolution =>Good aspect ratio
- Scalability friendly
- **8 minor singularities**

Best for ultra-high resolution modeling

Shallow water equations



$$\frac{\partial}{\partial t}h + \nabla.(Vh) = 0$$

$$\frac{\partial}{\partial t}u = \Omega v - \frac{1}{A\cos\theta}\frac{\partial}{\partial\lambda}[\kappa + \Phi]$$

$$\frac{\partial}{\partial t}v = -\Omega u - \frac{1}{A}\frac{\partial}{\partial\theta}[\kappa + \Phi]$$

$\nabla \times$

$$\frac{\partial}{\partial t}h + \nabla.(Vh) = 0$$

$$\frac{\partial}{\partial t}\Omega + \nabla.(V\Omega) = 0$$

Transported in the same manner!

- Better correlation during time integration
- Ensuring that higher order dynamical quantities such as the potential vorticity and divergence are better represented

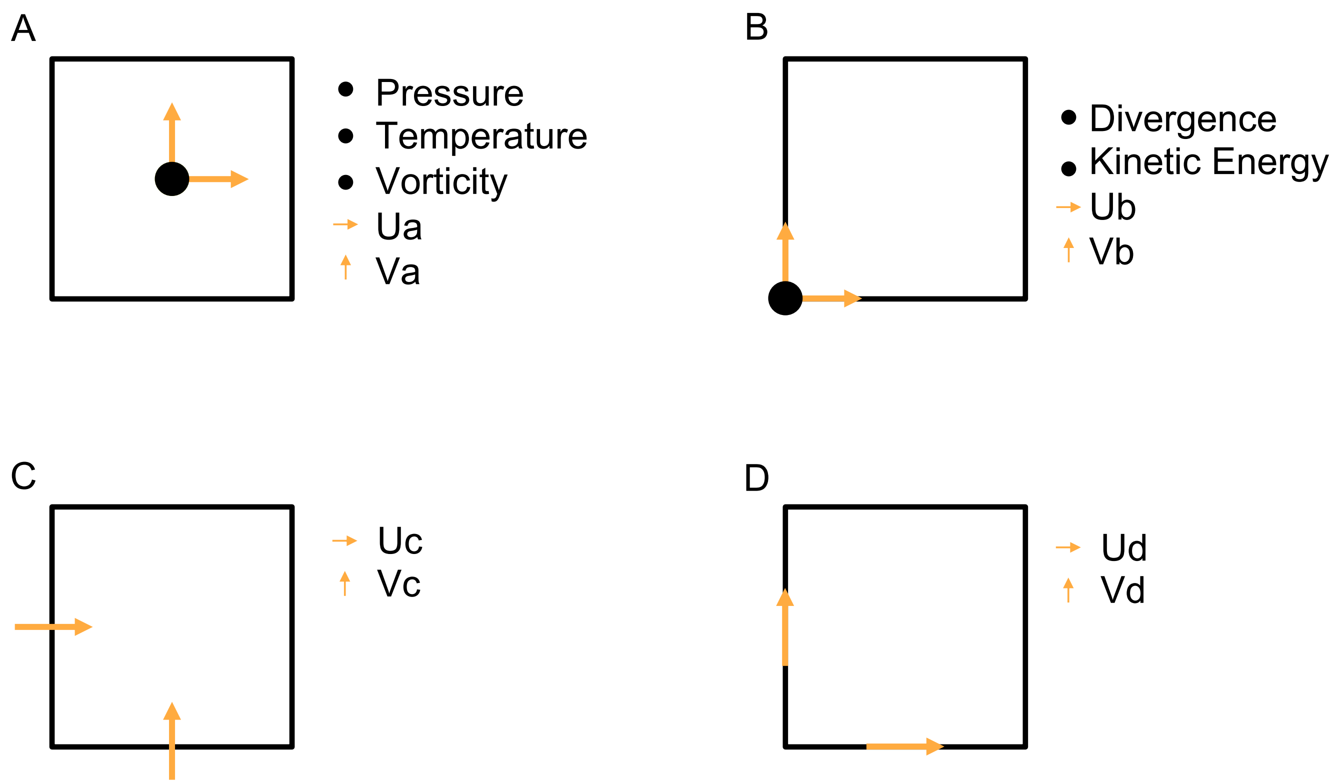
$$\kappa = \frac{1}{2}(u^2 + v^2)$$

$$\Omega = 2\omega\sin\theta + \nabla \times V$$

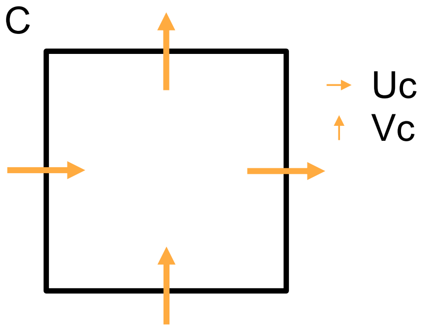
$$\Phi = \Phi_s + gh$$

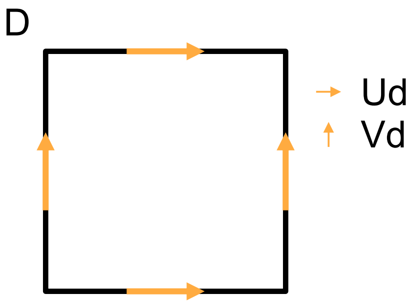
How are these variables placed on a grid?

Variable's locations and grid staggering



The C/D grid system





Pressure gradient (linear):
- C grid requires no averaging (best)
- D grid requires averaging in both directions (worst)

Geostrophic balance (linear):
- C grid requires averaging in both directions (worst)
- D grid requires no averaging (best)

Potential vorticity and helicity (nonlinear):
- C grid is the worst grid for vorticity and helicity
- D grid is the best for vorticity advection and representation of updraft helicity (severe storms)

A combination of C and D is better than a pure C or a pure D!

C&D work together like Yin-Yan!

Adapted from SJ Lin

What is the Duo-Grid?

Answer: It is not the C/D grid system

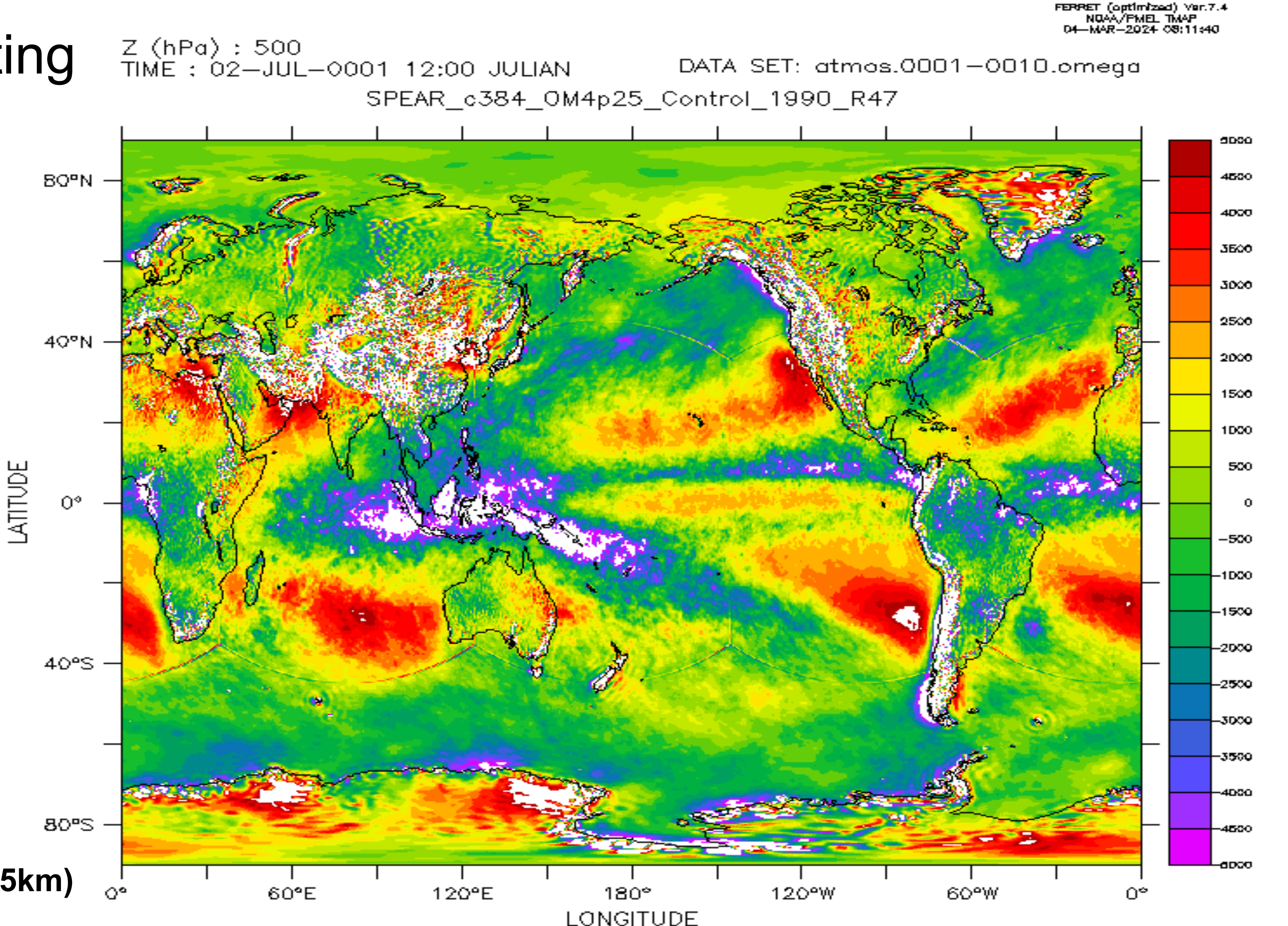
Answer: A solution for grid imprinting

What is grid imprinting?!

Answer: Grid discontinuities generate artifacts and noise in the numerical solution at its corresponding location. This noise will manifest some of the grid shape in the numerical solution thus the name grid imprinting.

GOAL: Find a solution to grid imprinting that is **better** than the current implemented edge handling

Grid imprinting



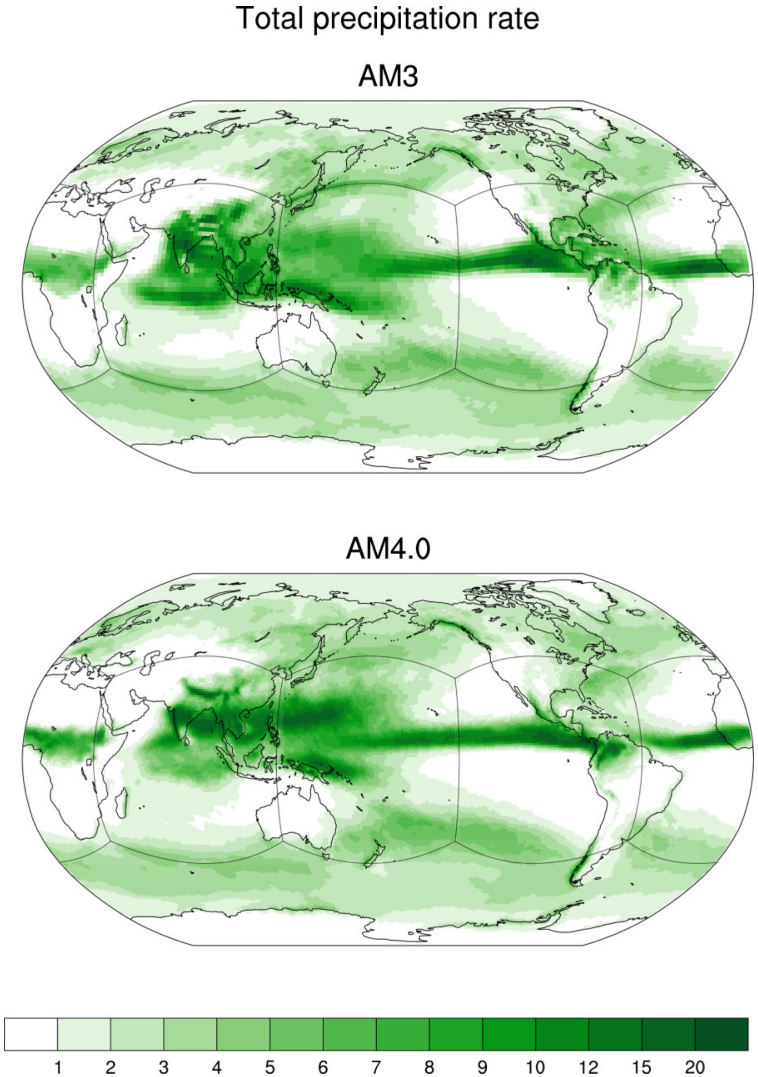
SPEAR (25km)

Courtesy Cooke&Delworth

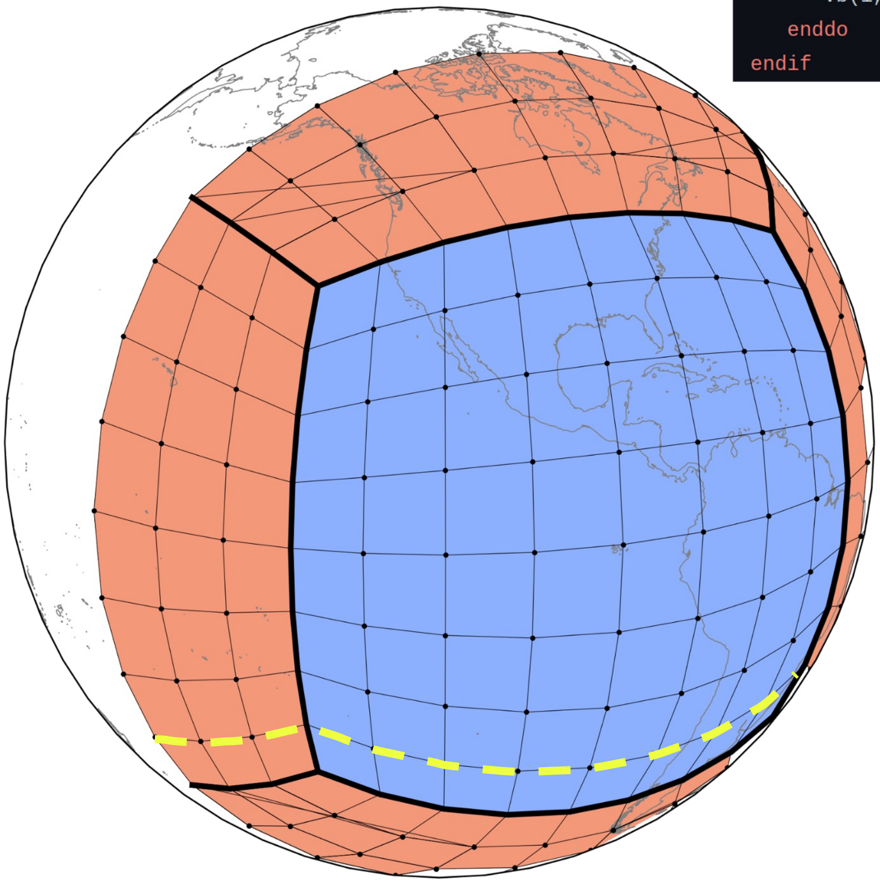
Grid imprinting

"A concern when using the cubed-sphere grid is imprinting of the grid on the model's climatology. Figure 1 shows an example of the grid imprinting in AM3 *in a plot of the climatological mean precipitation distribution in August, the season in which this distortion is most evident. The effect on precipitation of the northern edge of the cube face that passes through the North Pacific is especially visible. It is the large-scale rather than parameterized convective precipitation that suffers from this imprinting.* ***This is the worst case of distortion that we have found in any single month in this field*** *"*

(Zhao et al., 2018)



Grid imprinting



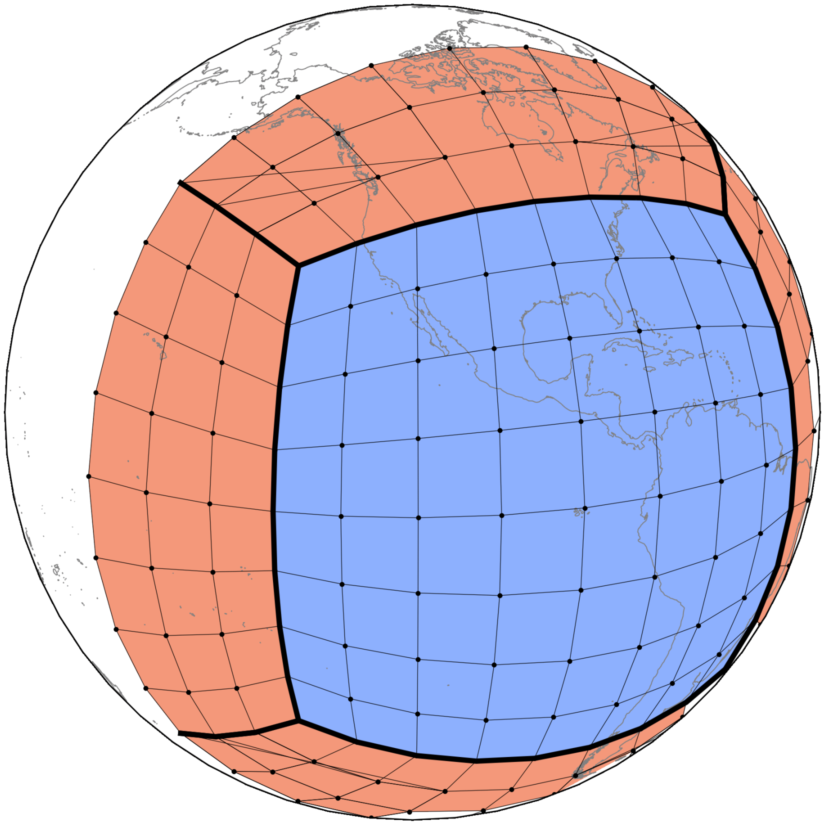
```
if ( (je+1)==npy ) then
   do i=is,ie+1
      vb(i,npy) = dt5*(vt(i-1,npy)+vt(i,npy)) ! corner values are incorrect
   enddo
endif
```

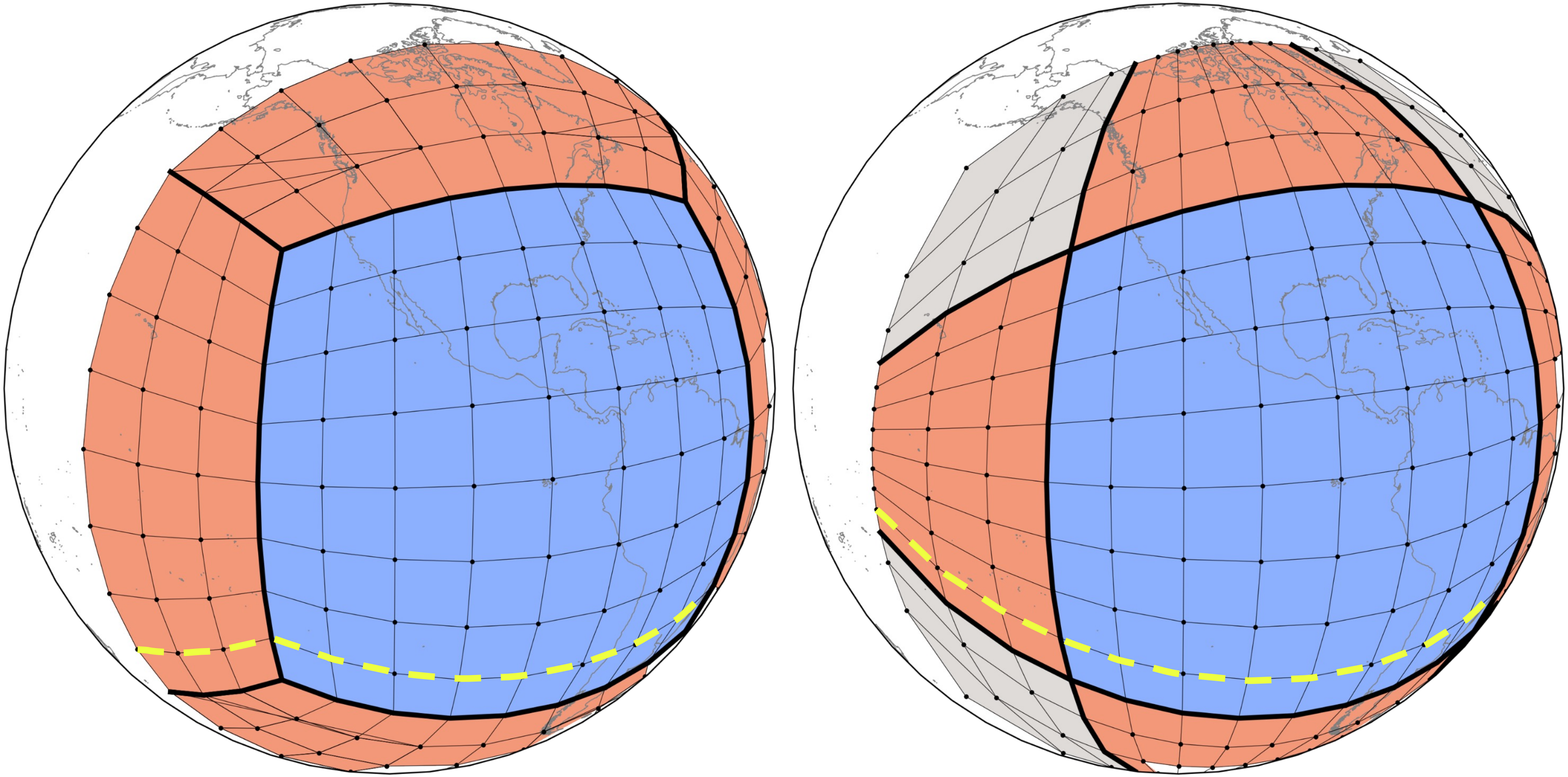
```
! East edge:
      if ( (ie+1)==npx ) then
         do j=jsd,jed
            if ( uc(npx,j)*dt > 0. ) then
               ut(npx,j) = uc(npx,j) / sin_sg(npx-1,j,3)
            else
               ut(npx,j) = uc(npx,j) / sin_sg(npx,j,1)
            endif
         enddo

          do j=max(3,js), min(npy-2,je+1)
             vt(npx-1,j) = vc(npx-1,j) - 0.25*cosa_v(npx-1,j)*   &
                          (ut(npx-1,j-1)+ut(npx,j-1)+ut(npx-1,j)+ut(npx,j))
             vt(npx,j) = vc(npx,j) - 0.25*cosa_v(npx,j)*   &
                        (ut(npx,j-1)+ut(npx+1,j-1)+ut(npx,j)+ut(npx+1,j))
          enddo
      endif
```

```
if ( se_corner ) then
     i = npx
     ke(i,1) = dt6*( (ut(i,1) + ut(i,  0)) * u(i-1,1) + &
                     (vt(i,1) + vt(i-1,1)) * v(i,  1) + &
                     (ut(i,1) - vt(i-1,1)) * u(i,  1) )
endif
```

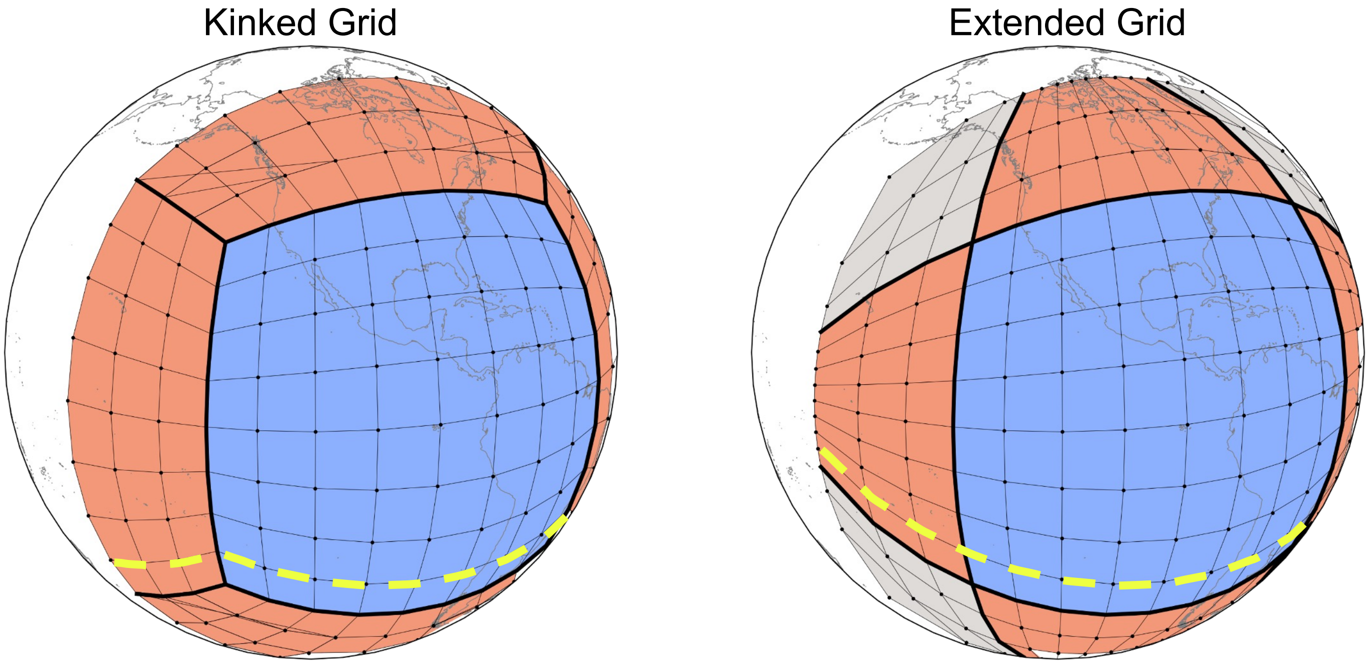
```
if ( fill_c ) call fill_corners(divg_d, npx, npy, FILL=XDir, BGRID=.true.)
```

The Duo-Grid





The Duo-Grid



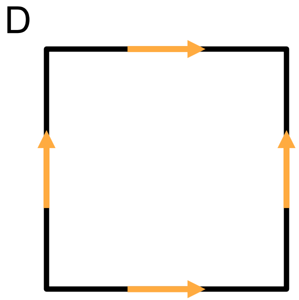
- Continuous integration along great circle lines => No other edge/corner handling code is required!
- The halo remapping algorithm and Duo extension are directly implemented into tiles' halo update message passing calls.
- Minimize data movement on CPU/GPU hybrid systems => Stepping stone for future FV3 developments on GPUs

Mouallem et al., JAMES, 2023

Challenges

- Extend the non-staggered duo grid algorithm to support all staggered and unstaggered FV3 variables (A-B-C-D)
- Implement a corner handling algorithm since the 2D transport scheme reaches the corner region.
- Break down complex and optimized subroutines (such as d_sw) to apply flux averaging on different components used to assemble the time advanced quantities
- Bypass all the edge handling codes (solver, grid generation, etc..)

Duo-grid algorithm for staggered variables

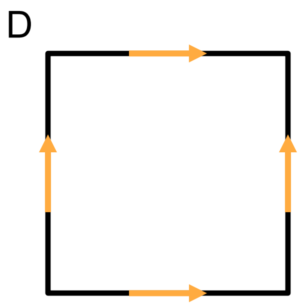


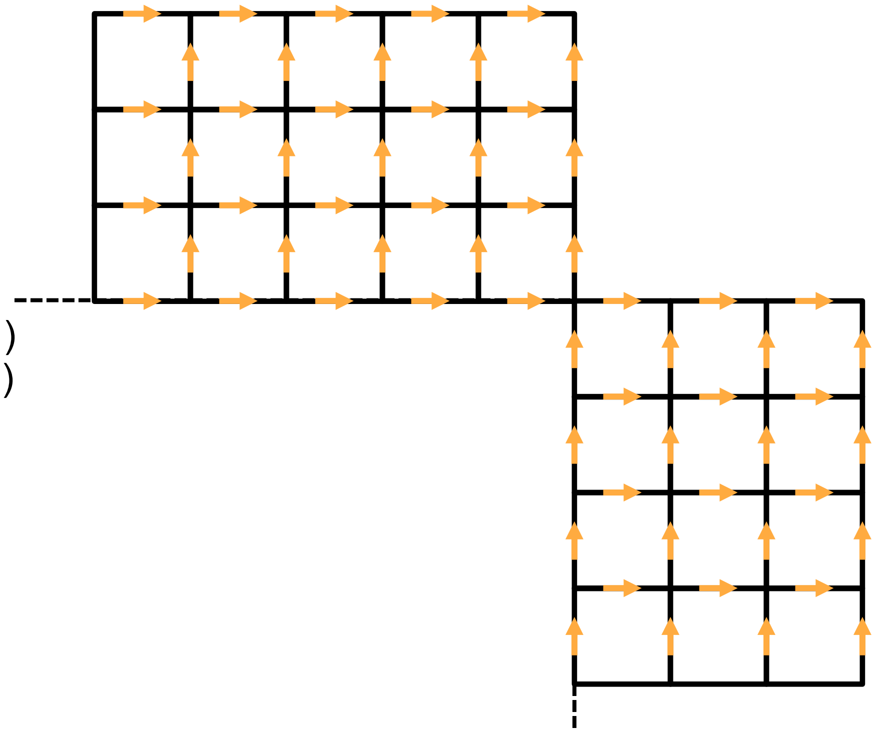
→ Ud (: ied , : jed **+ 1**)

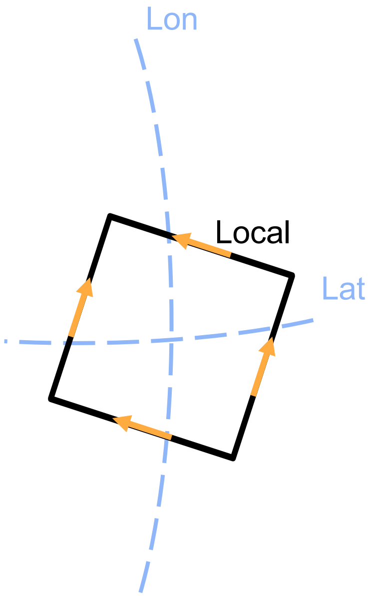
↑ Vd (: ied **+ 1**, : jed)

Duo-Grid algorithm – Staggered variables

1. Project the local velocities U and V to the center or (A-grid) location in local coordinates.



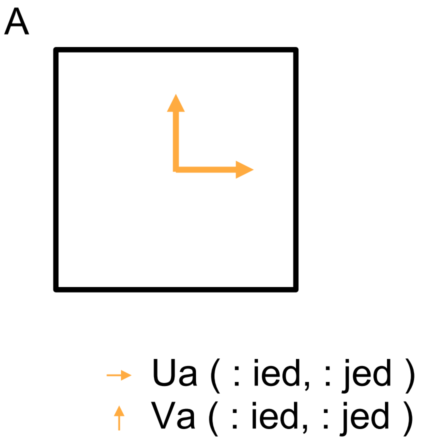


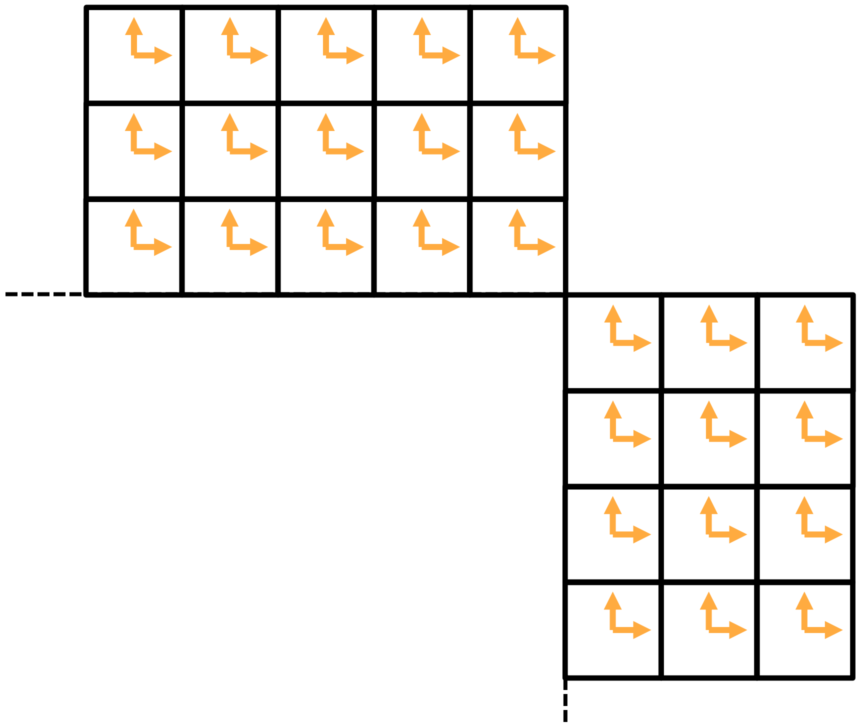


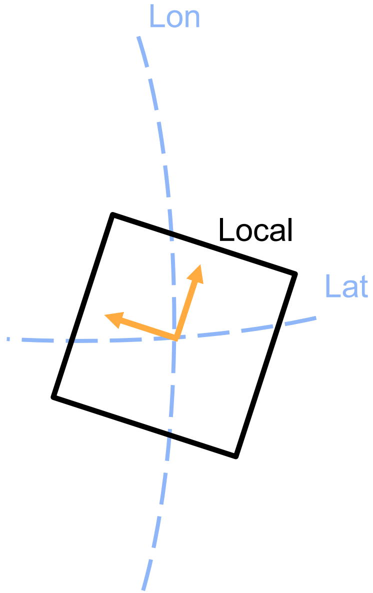
→ Ud (: ied , : jed **+ 1**)
↑ Vd (: ied **+ 1**, : jed)

Duo-Grid algorithm – Staggered variables

1. Project the local velocities U and V to the center or (A-grid) location in local coordinates.

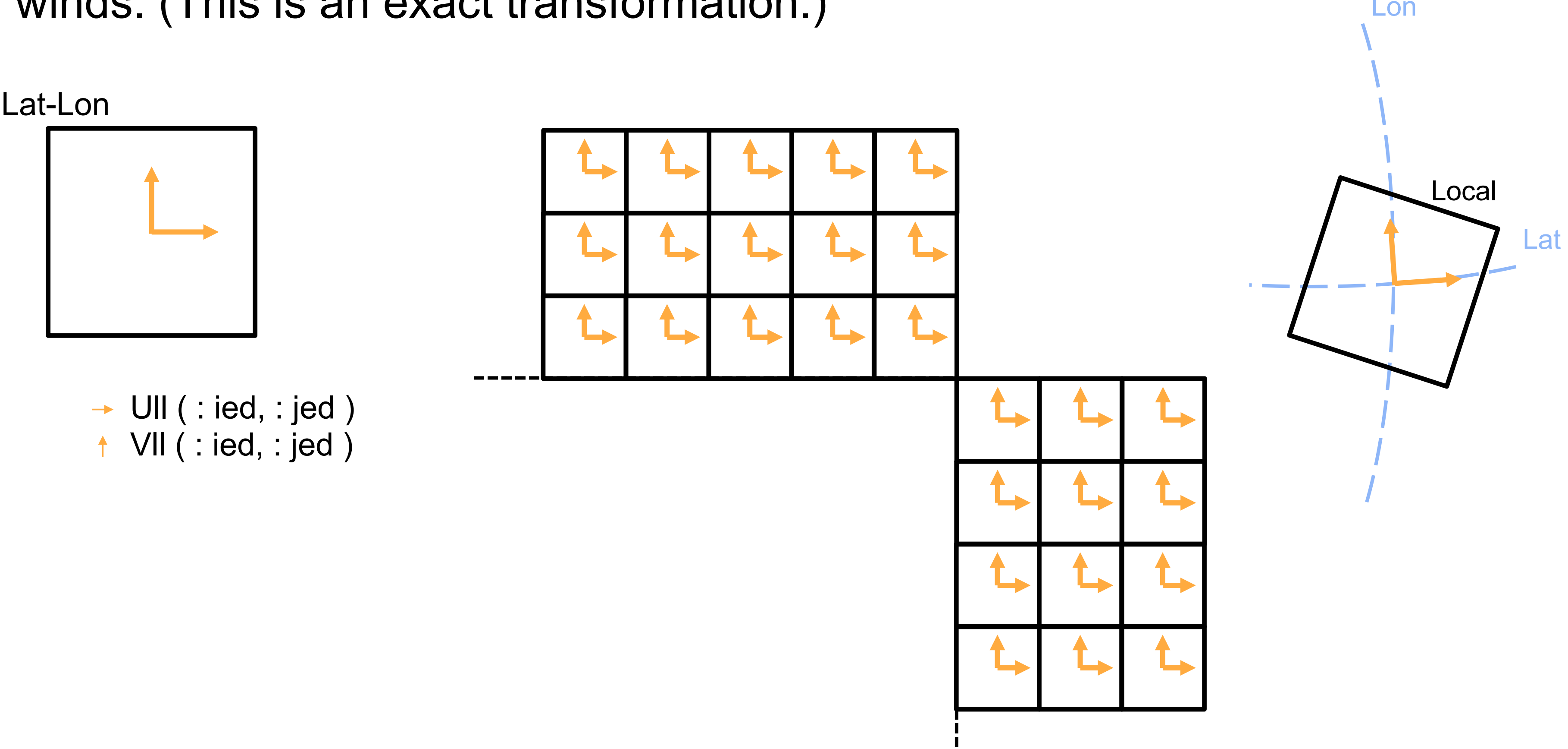






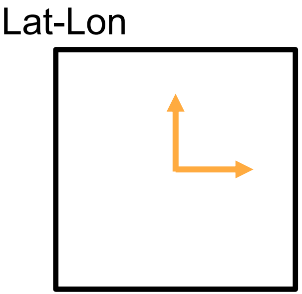
Duo-Grid algorithm – Staggered variables

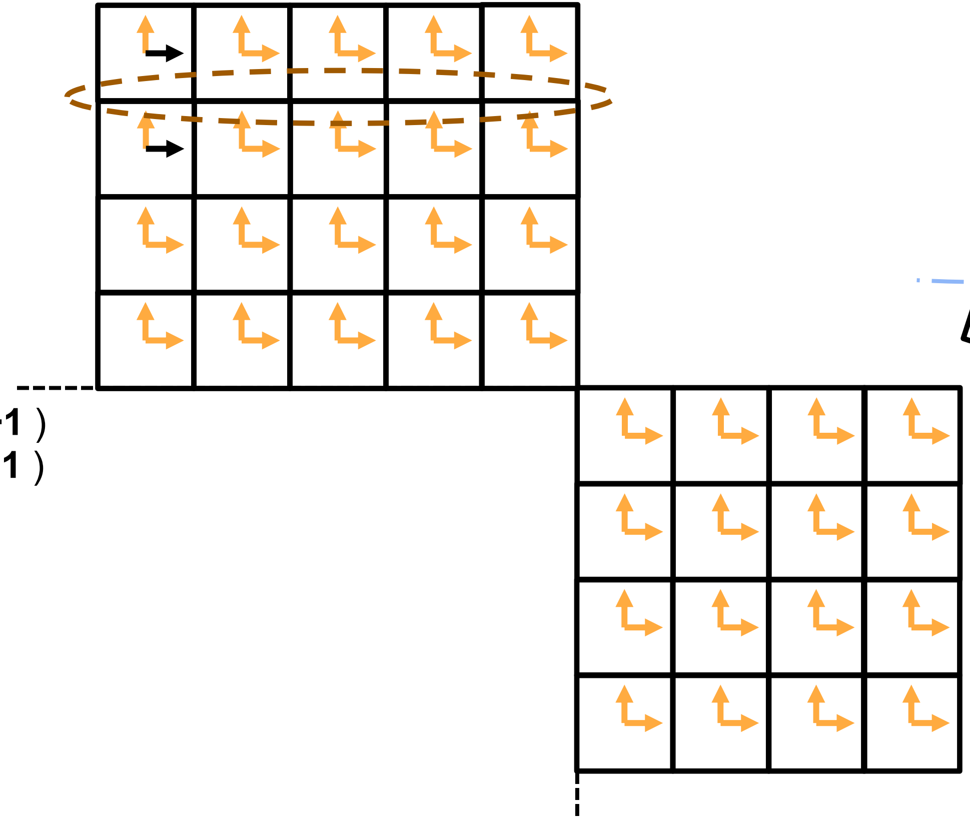
2. Rotate the local velocities into the earth-relative zonal and meridional winds. (This is an exact transformation.)

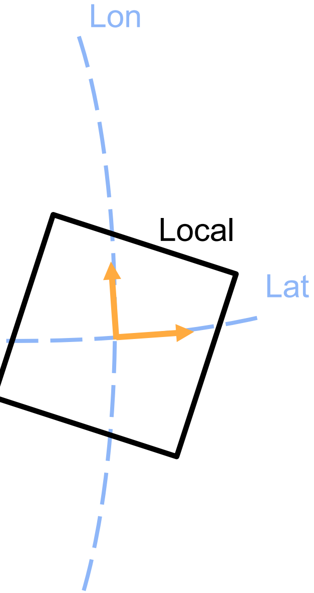


Duo-Grid algorithm – Staggered variables

*Consider extra halo layer

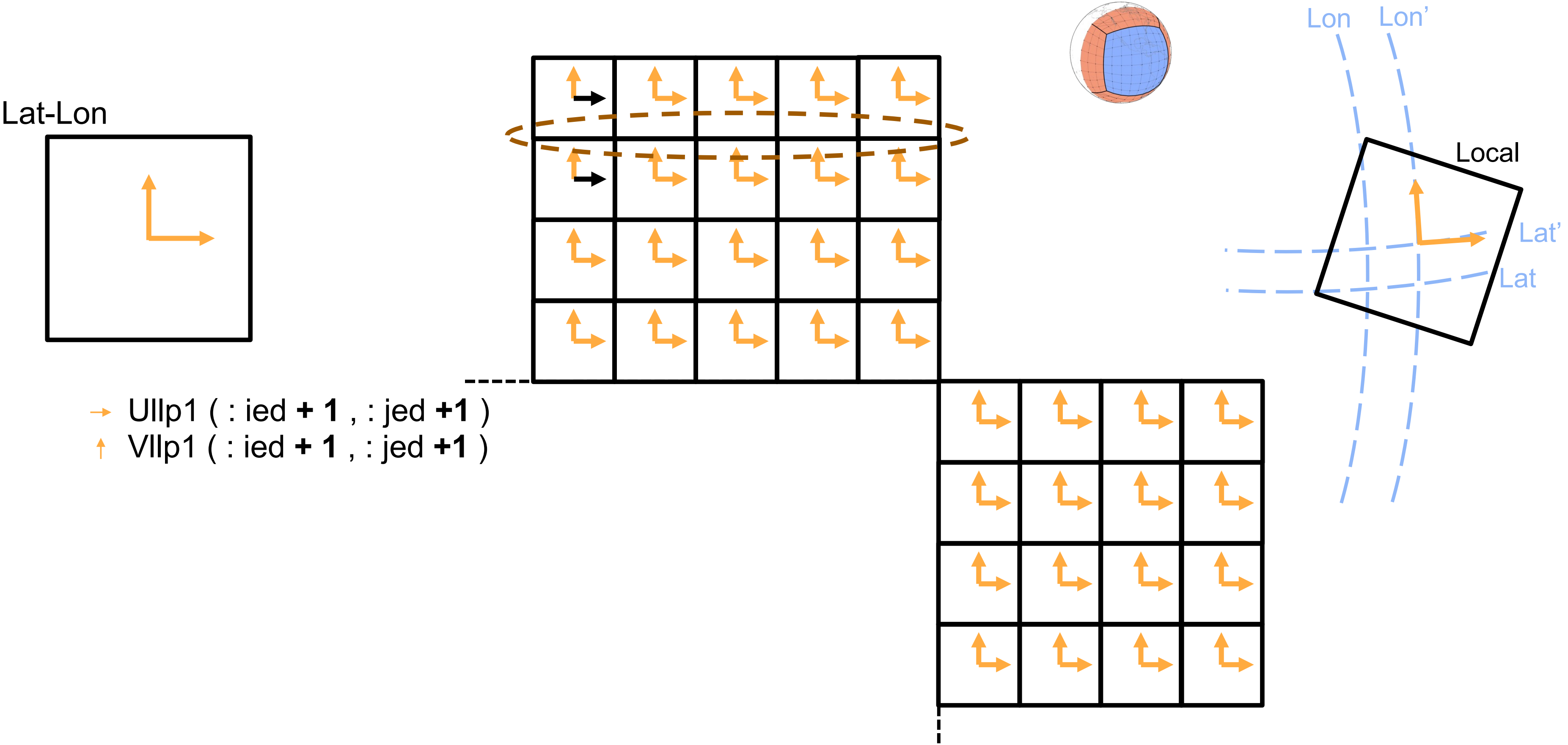






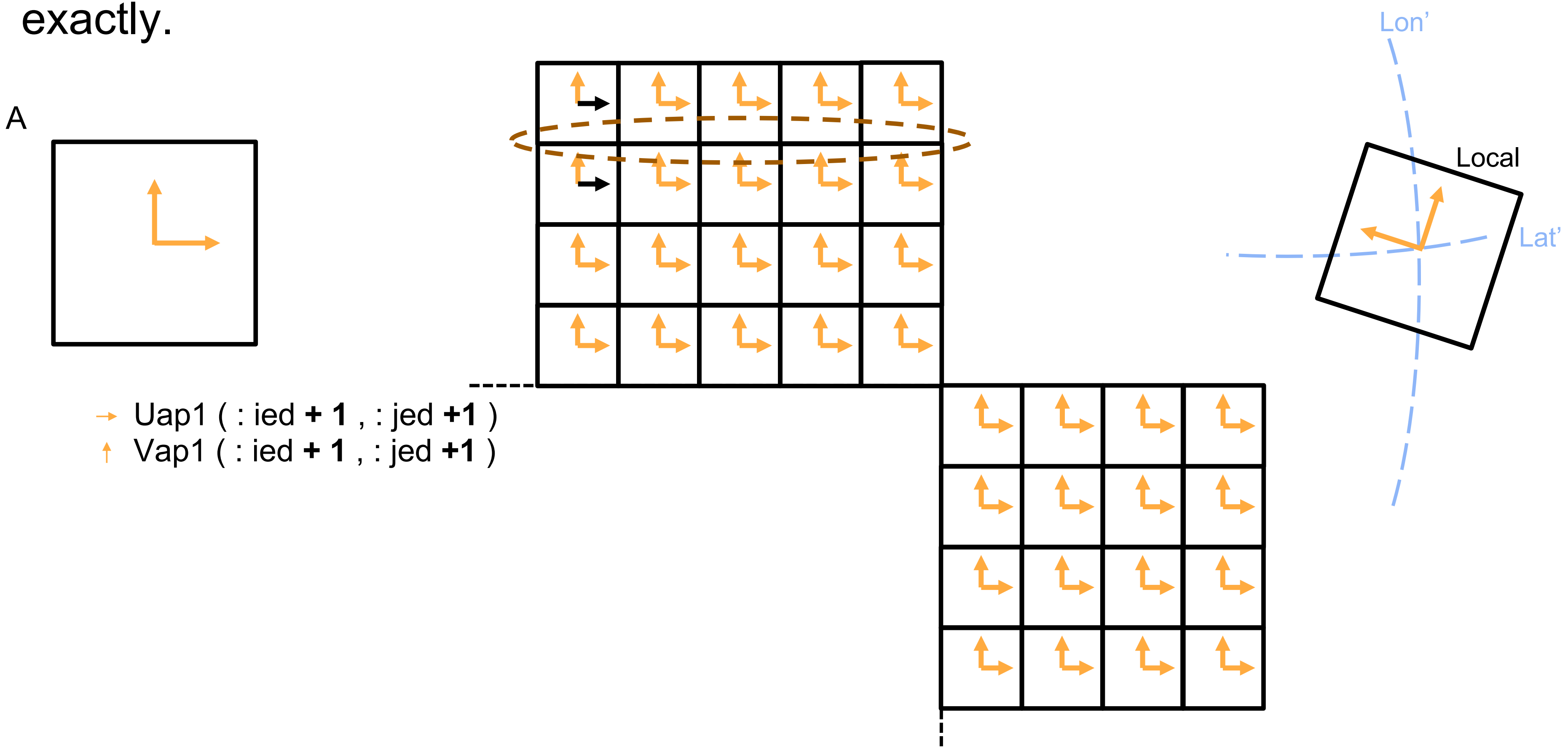
Duo-Grid algorithm – Staggered variables

3. Remap the earth-relative winds from kinked to extended grid locations.



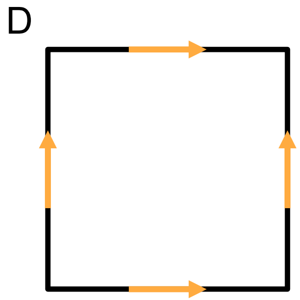
Duo-Grid algorithm – Staggered variables

4. Rotate the earth-relative winds back to the local velocities, again done exactly.

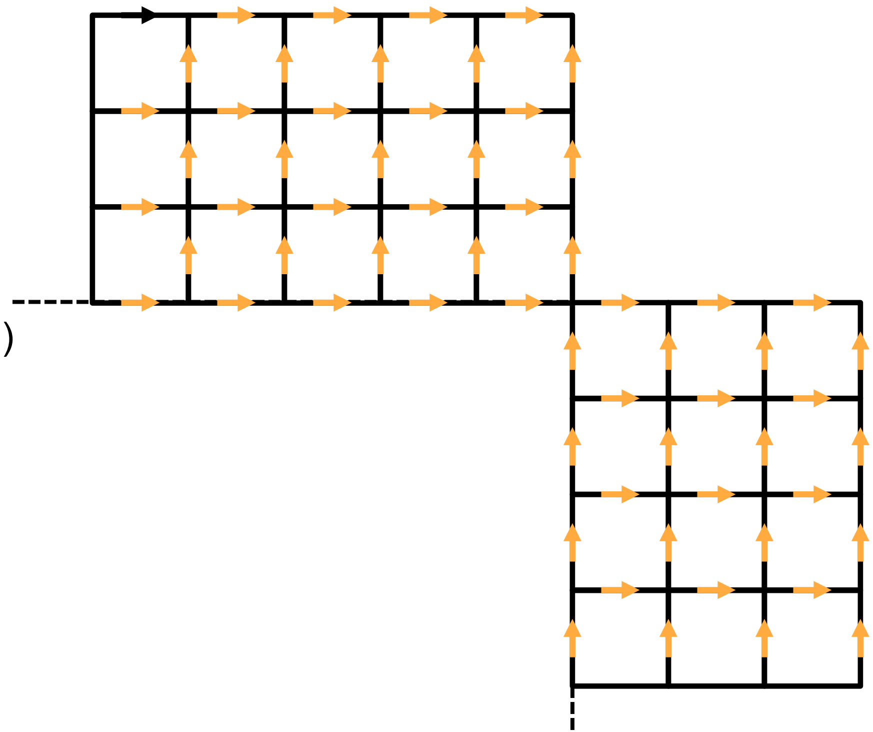


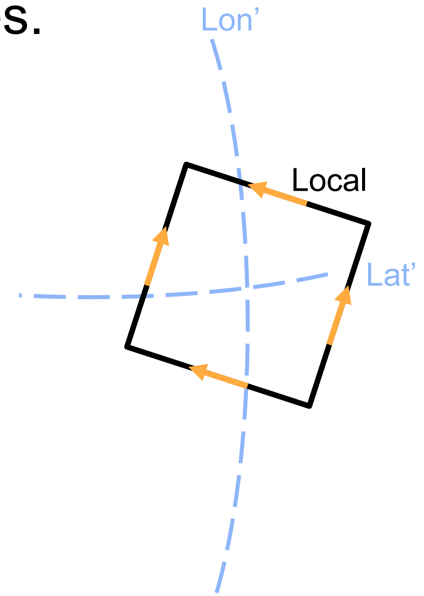
Duo-Grid algorithm – Staggered variables

5. Project the local velocities back to the appropriate grid cell faces to obtain the remapped fields in local cubed-sphere coordinates.



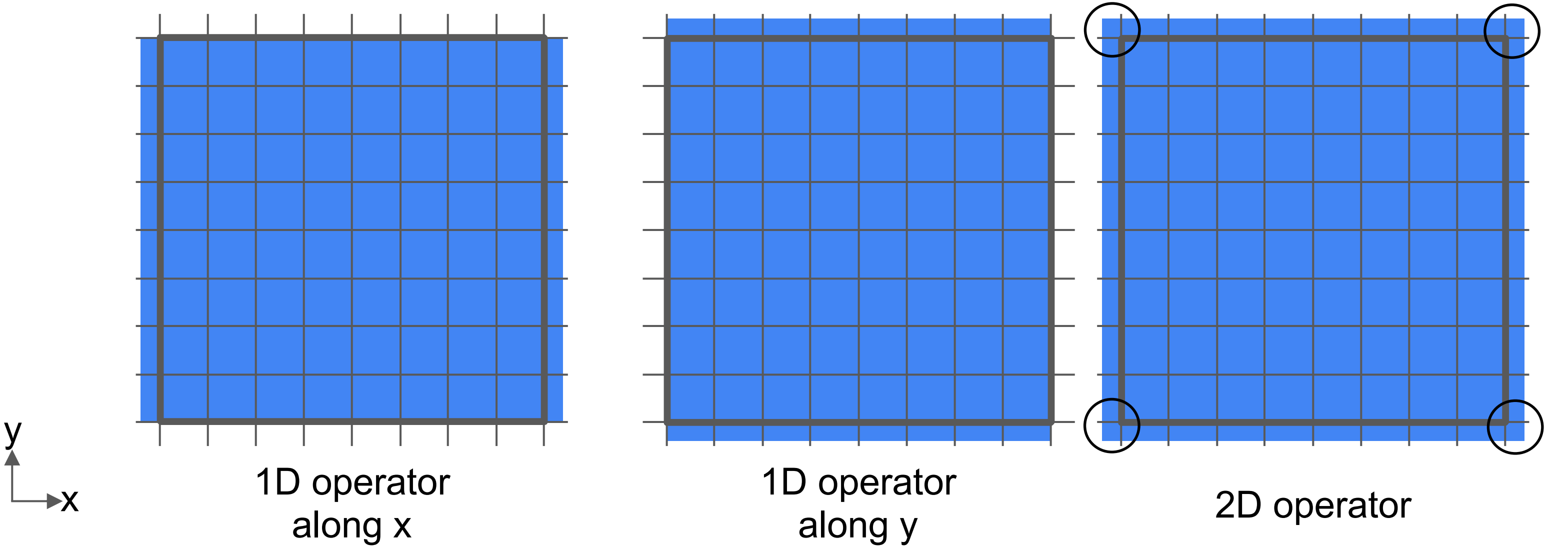
→ Ud (: ied , : jed **+ 1**)
↑ Vd (: ied **+ 1**, : jed)





Challenges

- Implement a corner handling algorithm since the 2D transport scheme reaches the corner region (check fv_tp_2d).
  The current FV3 corner handling uses copy_corner and fill_c to fill data from neighboring points in the x or y directions depending on the direction of operation within the algorithm



Reaches the four corner ghost regions due the nested x/y 1D operators and cross-derivative like terms (FG/GF)

Challenges:

- New Corner handling algorithm

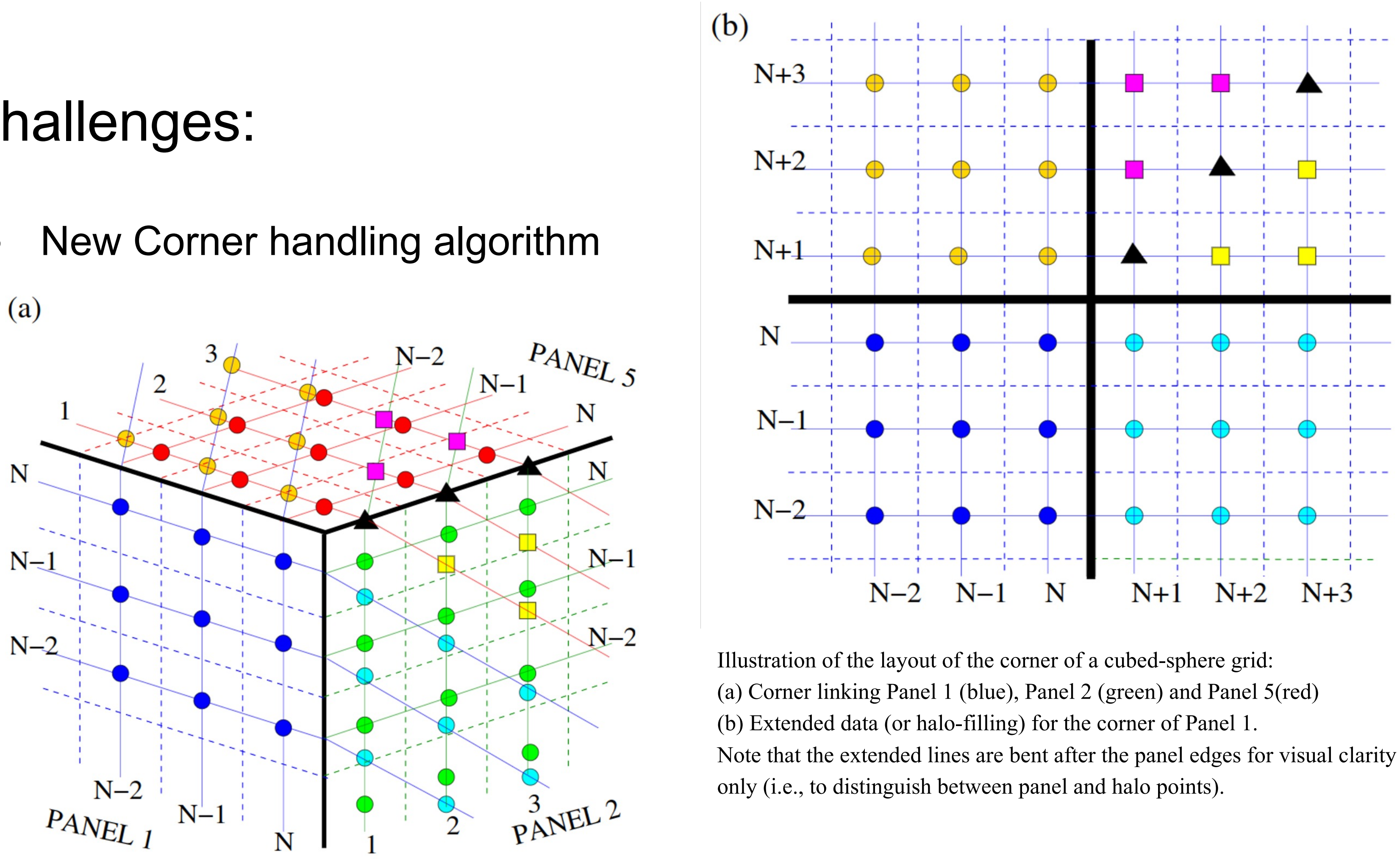


Illustration of the layout of the corner of a cubed-sphere grid:
(a) Corner linking Panel 1 (blue), Panel 2 (green) and Panel 5(red)
(b) Extended data (or halo-filling) for the corner of Panel 1.
Note that the extended lines are bent after the panel edges for visual clarity only (i.e., to distinguish between panel and halo points).

Zerroukat, M. & Allen, T.(2022)

Challenges:

- Break down complex and optimized subroutines (such as d_sw), separate flux computation and flux application, then in between, apply flux averaging on different components used to assemble the time advanced quantities
- Bypass all the edge handling codes (solver, grid generation, etc..)

```
if ( (je+1)==npy ) then
   do i=is,ie+1
      vb(i,npy) = dt5*(vt(i-1,npy)+vt(i,npy)) ! corner values are incorrect
   enddo
endif
```

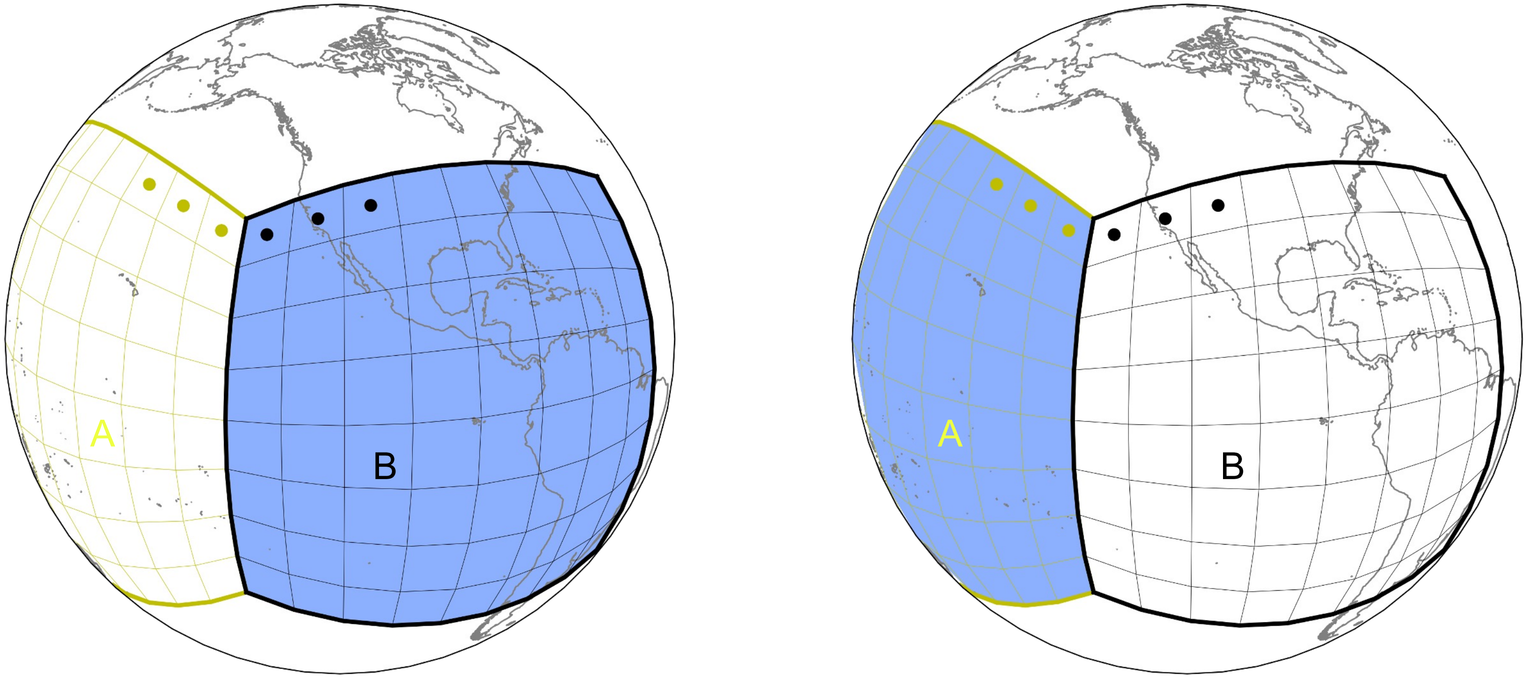
```
      call d_sw(vt(isd,jsd,k), delp(isd,jsd,k), ptc(isd,jsd,k),  pt(isd,jsd,k),      &
                u(isd,jsd,k),    v(isd,jsd,k),   w(isd:,jsd:,k),  uc(isd,jsd,k),      &
                vc(isd,jsd,k),   ua(isd,jsd,k),  va(isd,jsd,k), divgd(isd,jsd,k),   &
                mfx(is, js, k),  mfy(is, js, k),  cx(is, jsd,k),  cy(isd,js, k),    &
                crx(is, jsd,k),  cry(isd,js, k), xfx(is, jsd,k), yfx(isd,js, k),    &
#ifdef USE_COND
                q_con(isd:,jsd:,k),  z_rat(isd,jsd),  &
#else
                q_con(isd:,jsd:,1),  z_rat(isd,jsd),  &
#endif
                kgb, heat_s, diss_e, zvir, sphum, nq,  q,  k,  npz, flagstruct%inline_q,  dt,  &
                flagstruct%hord_tr, hord_m, hord_v, hord_t, hord_p,    &
                nord_k, nord_v(k), nord_w, nord_t, flagstruct%dddmp, d2_divg, flagstruct%d4_bg,  &
                damp_vt(k), damp_w, damp_t, d_con_k, &
                hydrostatic, gridstruct, flagstruct, bd)
```

Called from model/dyn_core.F90, routine in model/sw_core.F90

```
if ( se_corner ) then
     i = npx
     ke(i,1) = dt6*( (ut(i,1) + ut(i,  0)) * u(i-1,1) + &
                     (vt(i,1) + vt(i-1,1)) * v(i,  1) + &
                     (ut(i,1) - vt(i-1,1)) * u(i,  1) )
endif
```

```
if ( se_corner ) then
     i = npx
     ke(i,1) = dt6*( (ut(i,1) + ut(i,  0)) * u(i-1,1) + &
                     (vt(i,1) + vt(i-1,1)) * v(i,  1) + &
                     (ut(i,1) - vt(i-1,1)) * u(i,  1) )
endif
```

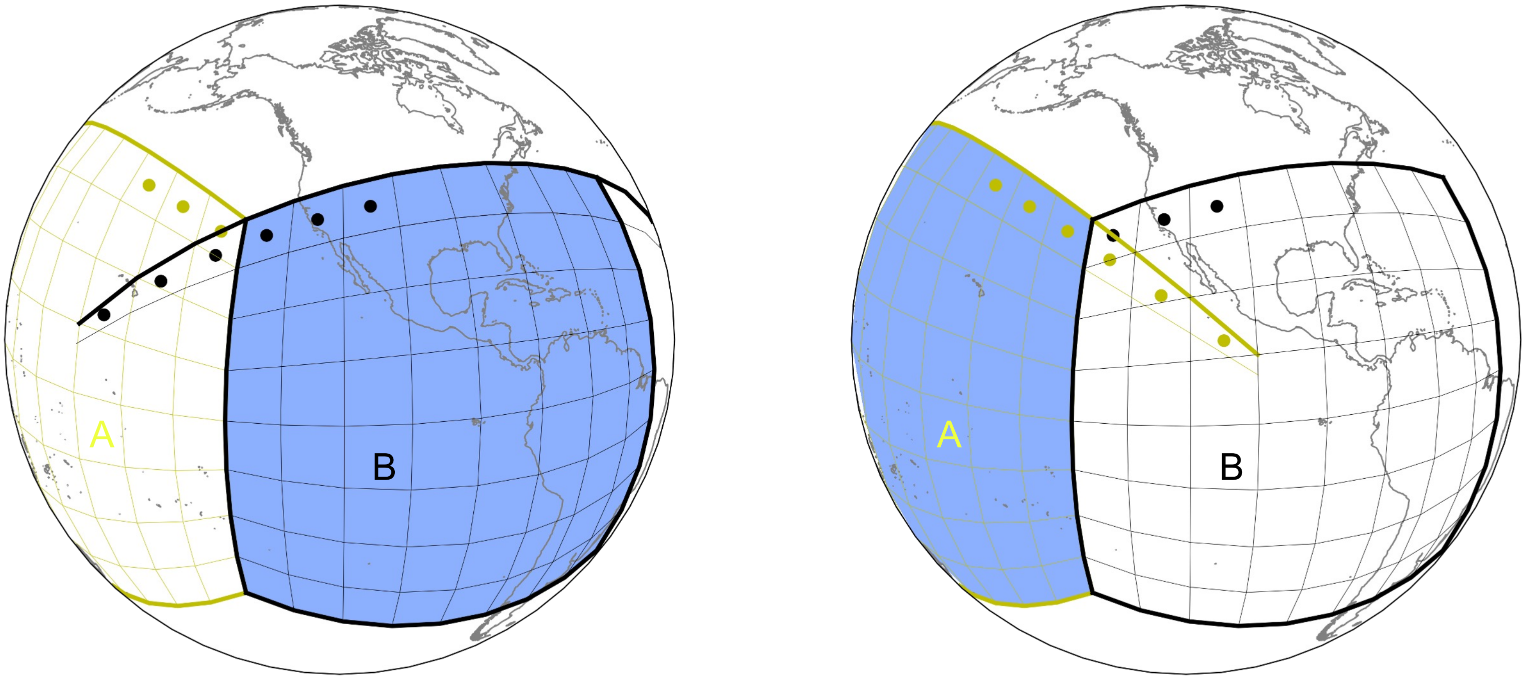
Flux averaging across tiles' edges



$$\text{Flux}_{B\to A} = \text{Flux}_{A\to B}$$

- No need for flux adjustment on a kinked grid, since the 2D SL scheme is free of directional bias. Same operation is performed on both sides of the edge.

Flux averaging across tiles' edges



$Flux_{B \to A}$ ≠ $Flux_{A \to B}$

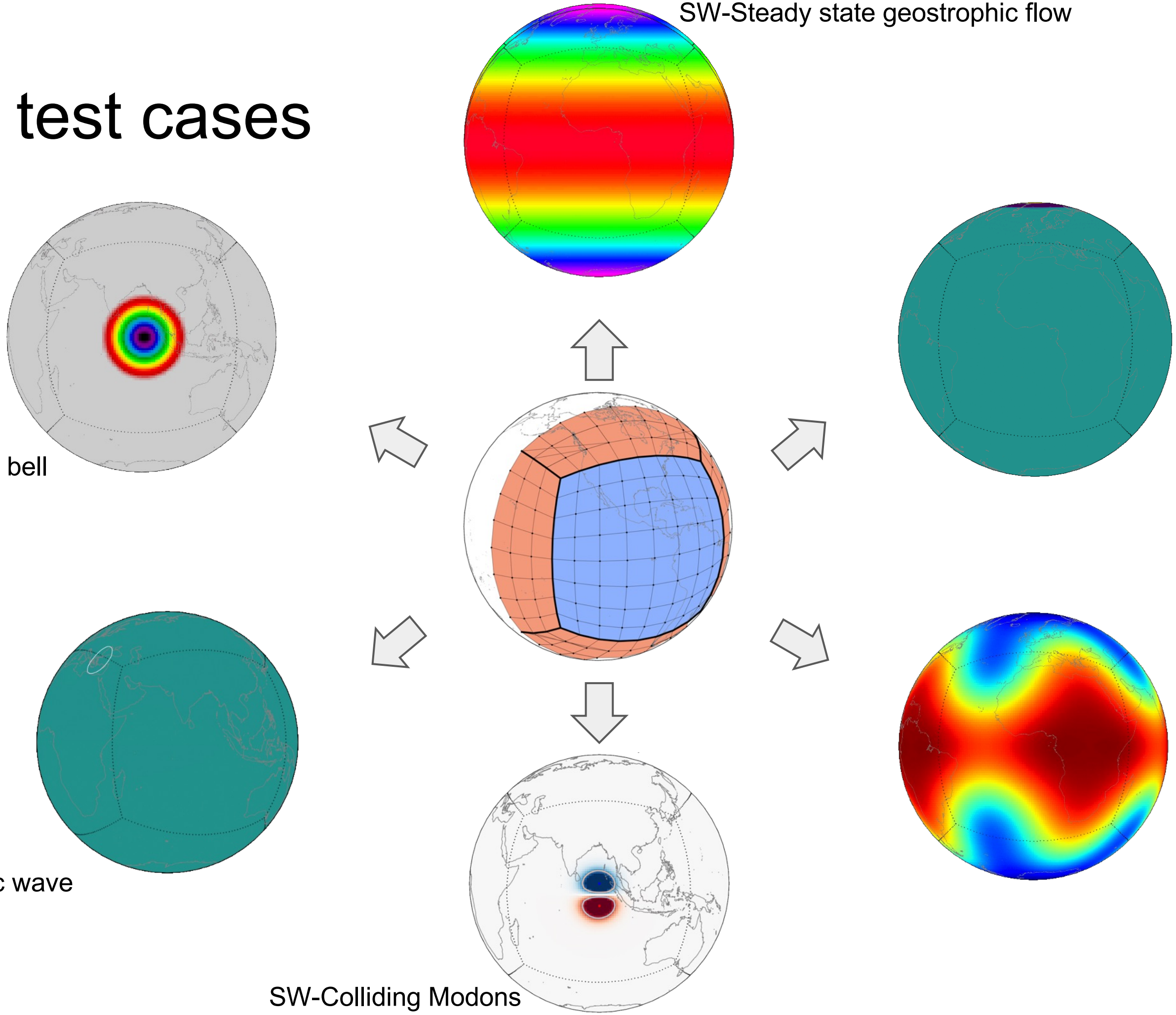
- Due the remapping algorithm on the Duo-Grid, flux adjustment is needed to ensure conservation properties. Fluxes at the same interface are shared then averaged between two neighboring tiles: $Flux = 0.5 * (Flux_{A \to B} + Flux_{B \to A})$

Idealized test cases

$$l_1(h) = \frac{I\left(|h - h_T|\right)}{I\left(|h_T|\right)}$$

$$l_2(h) = \sqrt{\frac{I\left((h - h_T)^2\right)}{I\left(h_T^2\right)}}$$

$$l_\infty(h) = \frac{max|h - h_T|}{max|h_T|}$$



Mouallem et al., JAMES, 2023