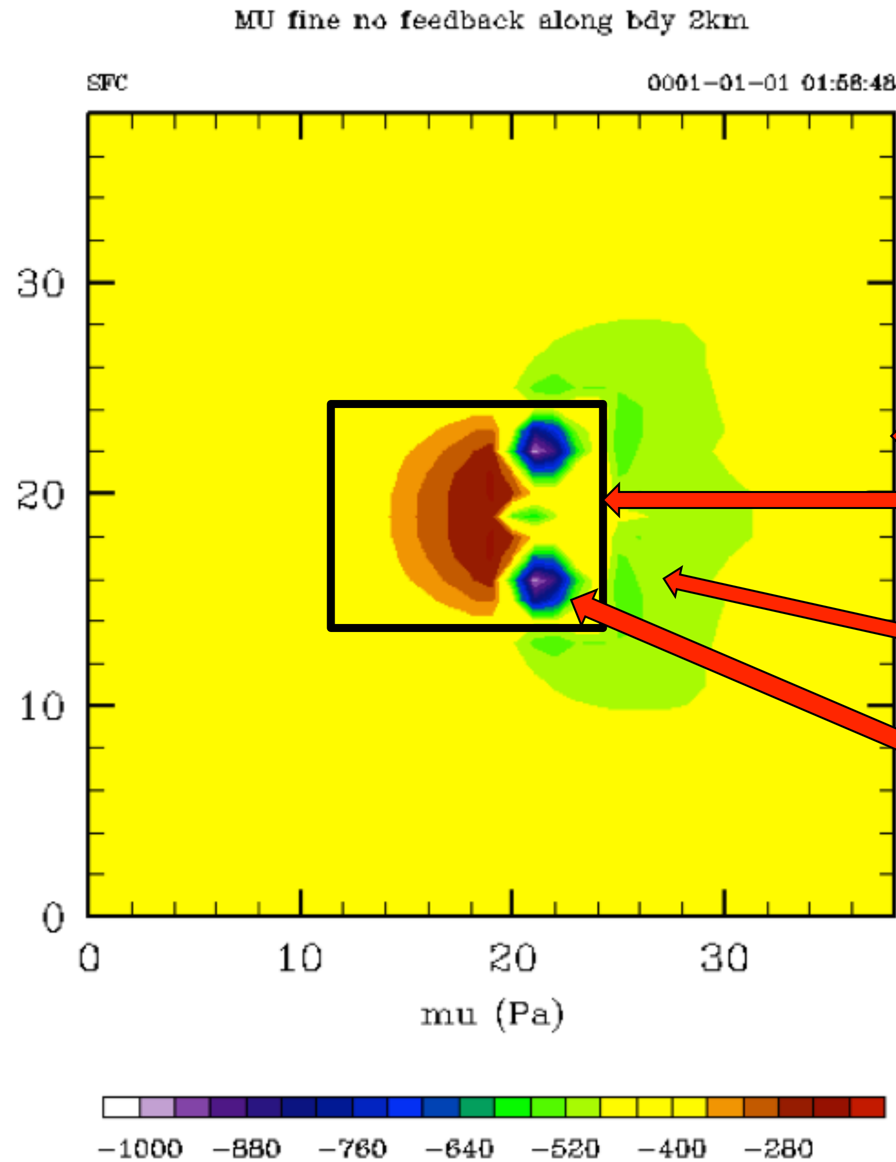
Nesting in WRF

Dave Gill



Super Cell QSS

2-domain run

CG

FG

Feedback: on

Split storm

Overview

- Nesting: Journalism 101: Who, what, why, when, where
- Domains
  - OK *vs* semi-OK *vs* not OK at all
  - Variable staggering CG to FG
  - Lateral forcing
  - Feedback
  - Masked interpolation
  - Time stepping for multi-domain
- Concurrent *vs* Offline Nesting
- Registry
  - U D F S
  - i2
- Some suggestions
  - Performance
  - Location, location, location
  - Inside out, start with inner domain
  - Go big or go home
  - Map factors, stability, time step

Nesting Basics - What is a nest

- A nest is a ***finer-resolution*** model run. It may be ***embedded*** simultaneously within a coarser-resolution (parent) model run, or ***run independently*** as a separate model forecast.
- The nest ***covers a portion*** of the parent domain, and is driven along its ***lateral boundaries*** by the parent domain.
- Nesting enables running at finer resolution without the following problems:
  - Uniformly high resolution over a large domain - prohibitively expensive
  - High resolution for a very small domain with mismatched time and spatial lateral boundary conditions

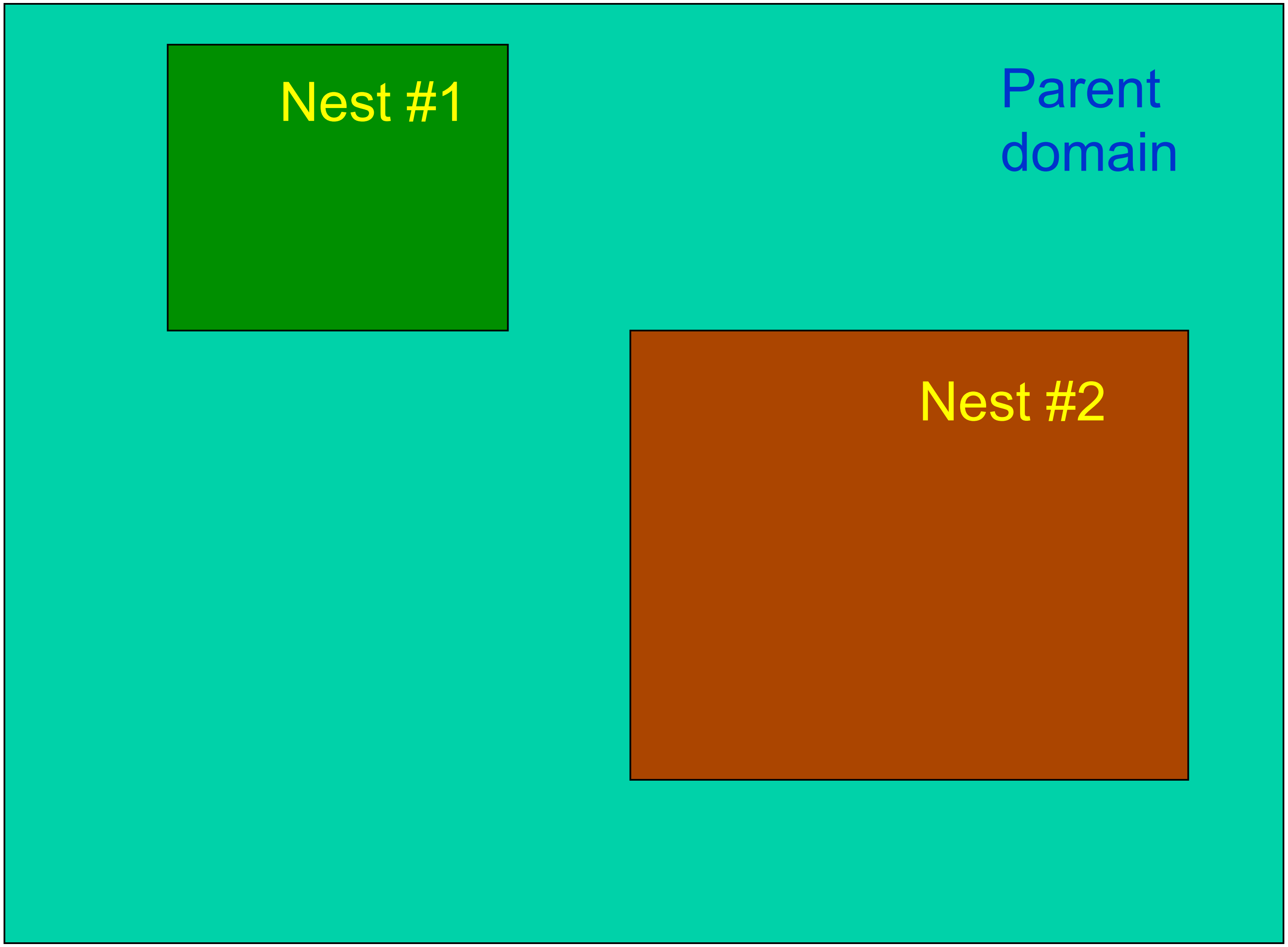
Nesting Basics

- One-way nesting via multiple model forecasts
- One-way nesting with a single model forecast, without feedback

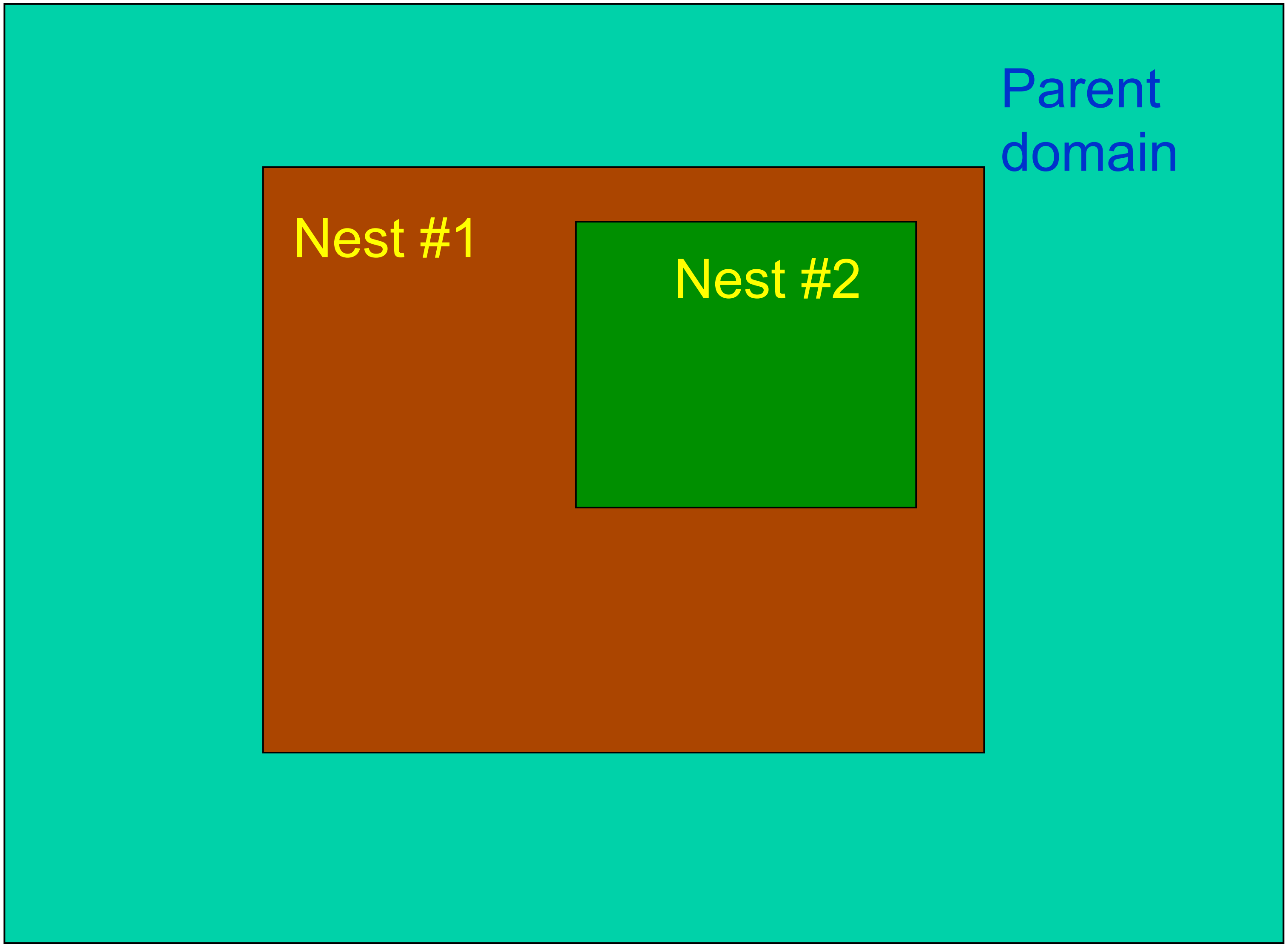
- One-way/two-way nesting with a single input file, all fields interpolated from the coarse grid
- One-way/two-way nesting with multiple input files, each domain with a full input data file
- One-way/two-way nesting with the coarse grid data including all meteorological fields, and the fine-grid domains including only the static files

- One-way/two-way nesting with a specified move for each nest
- One-way/two-way nesting with an automatic move on the nest determined through (usually) 700 hPa low tracking

Two nests on the same “level”, with a common parent domain

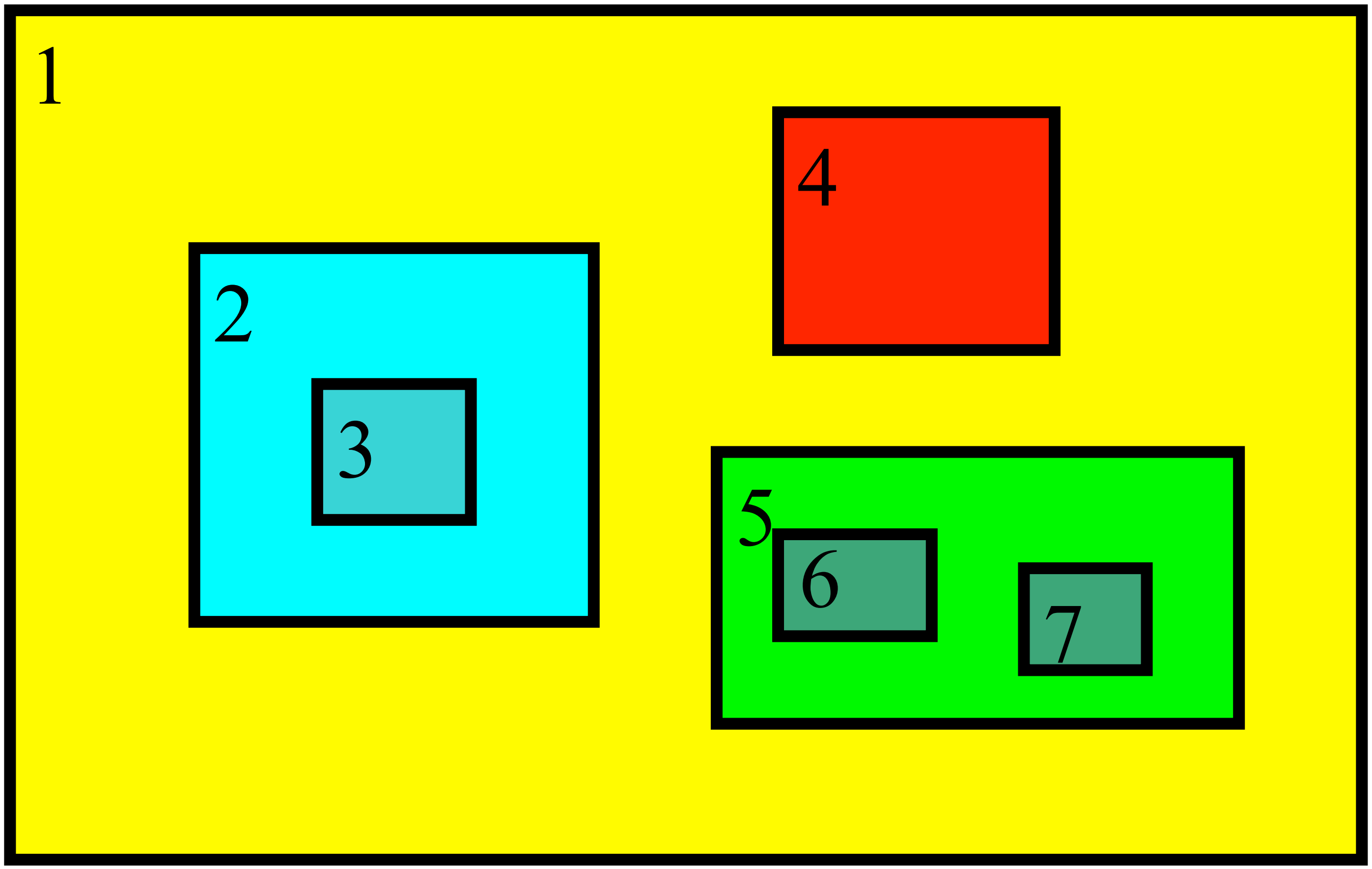


Two levels of nests, with nest #1 acting as the parent for nest #2



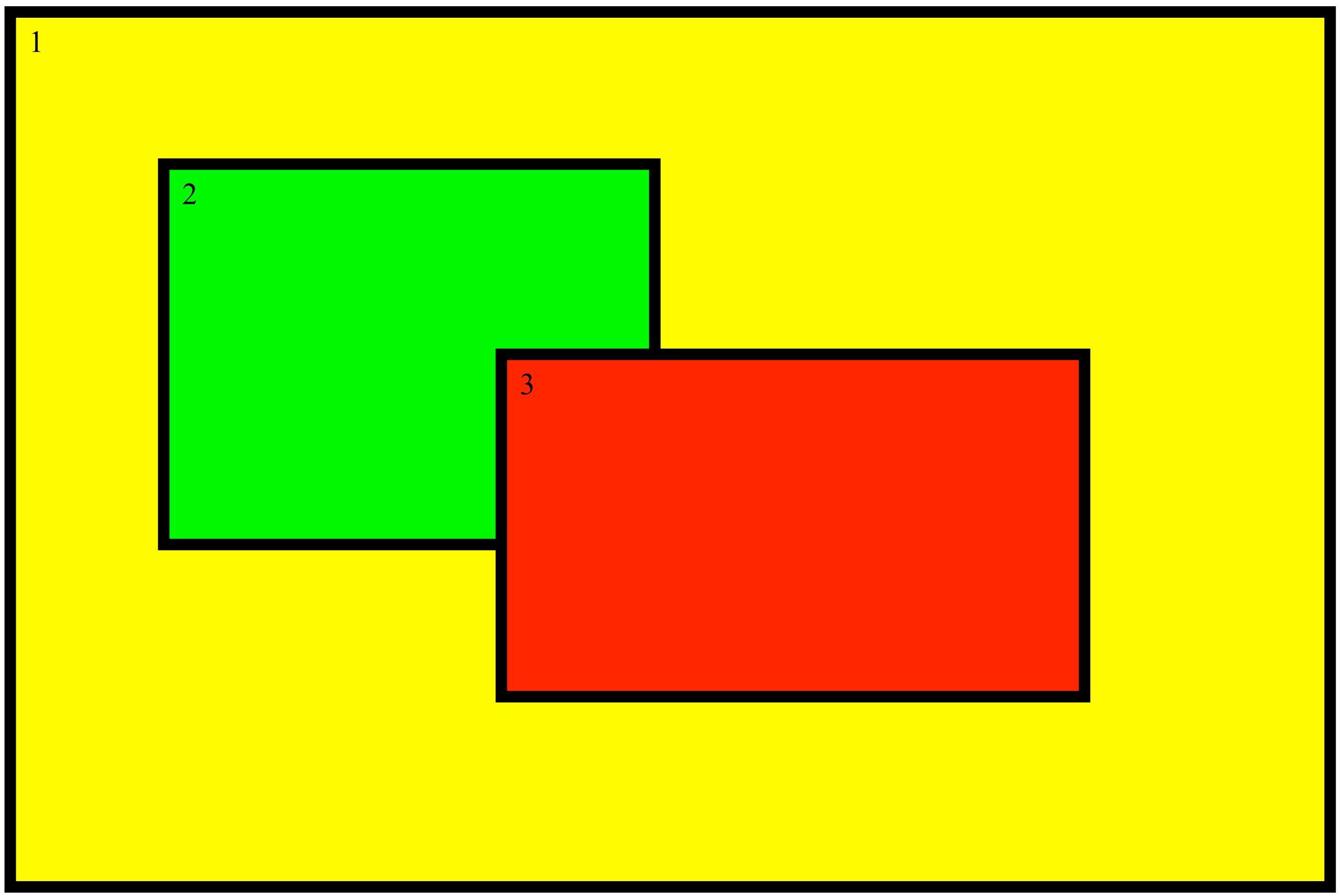
These are all OK

Telescoped to any depth
Any number of siblings

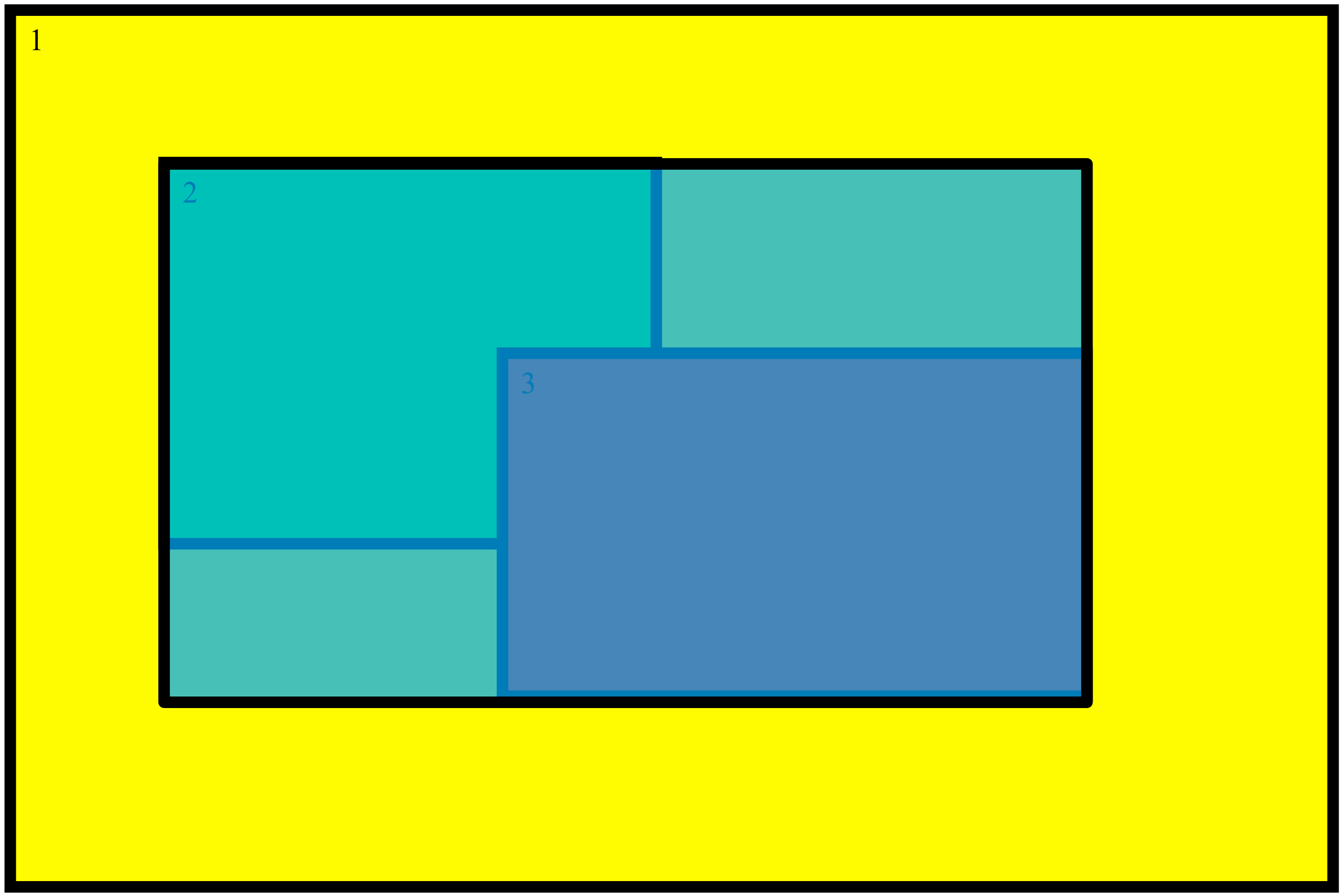


Not OK for 2-way

Child domains *may not* have overlapping points in the parent domain (1-way nesting excluded).

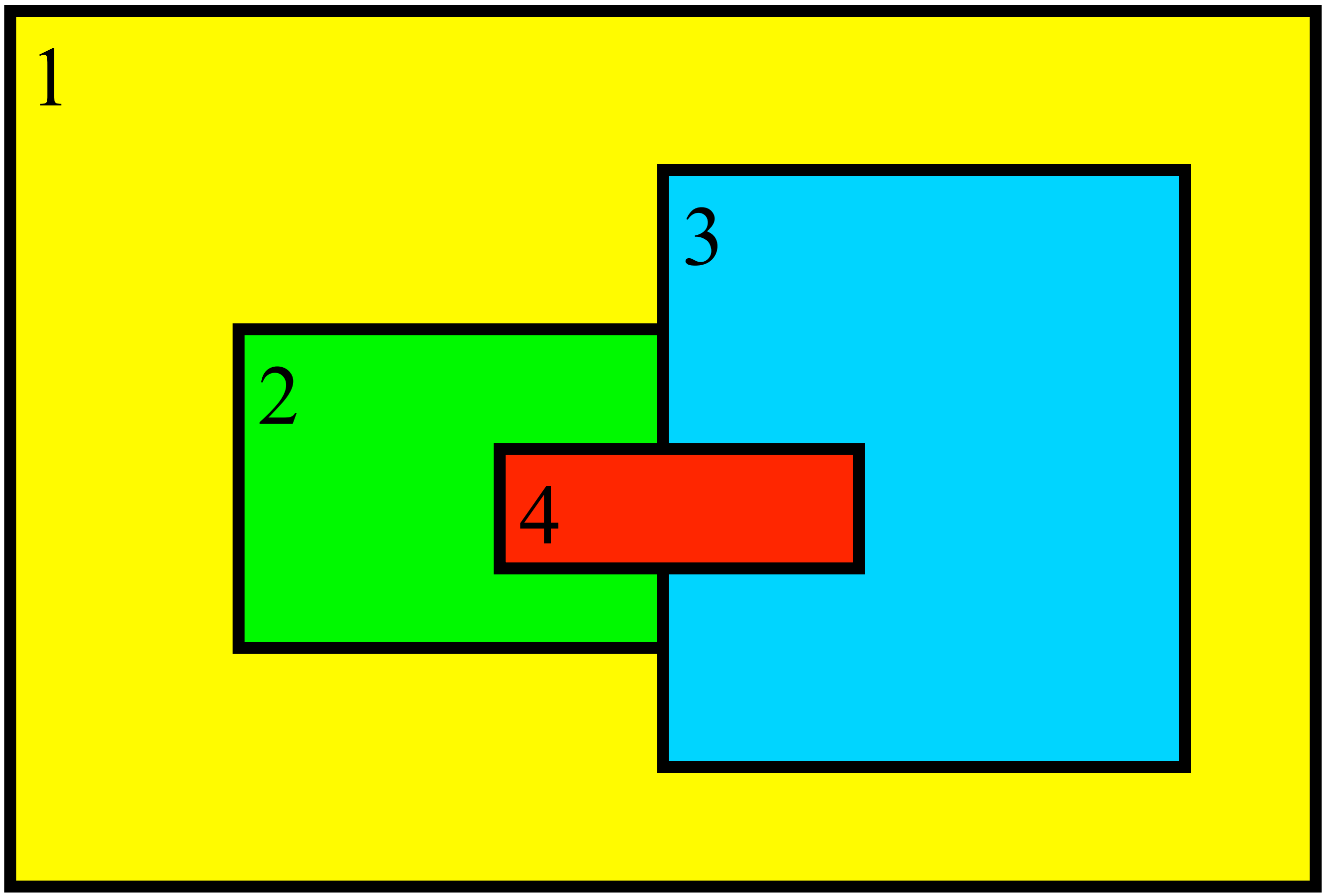


Preferred – One big Nest



Not OK either

Domains have one, and only one, parent -
(domain 4 is NOT acceptable even with 1-way nesting)

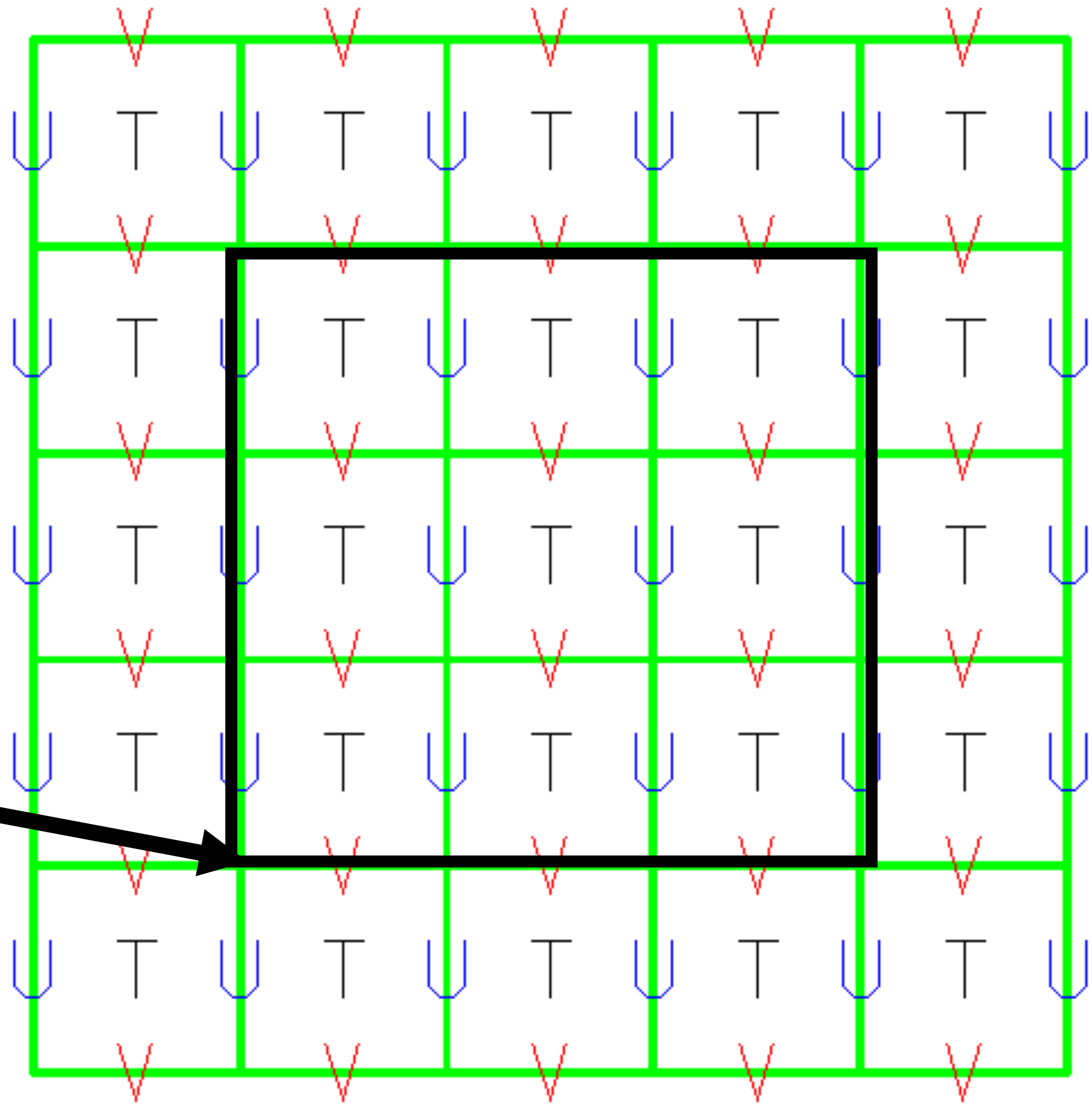


WRF Coarse-Fine Overlap

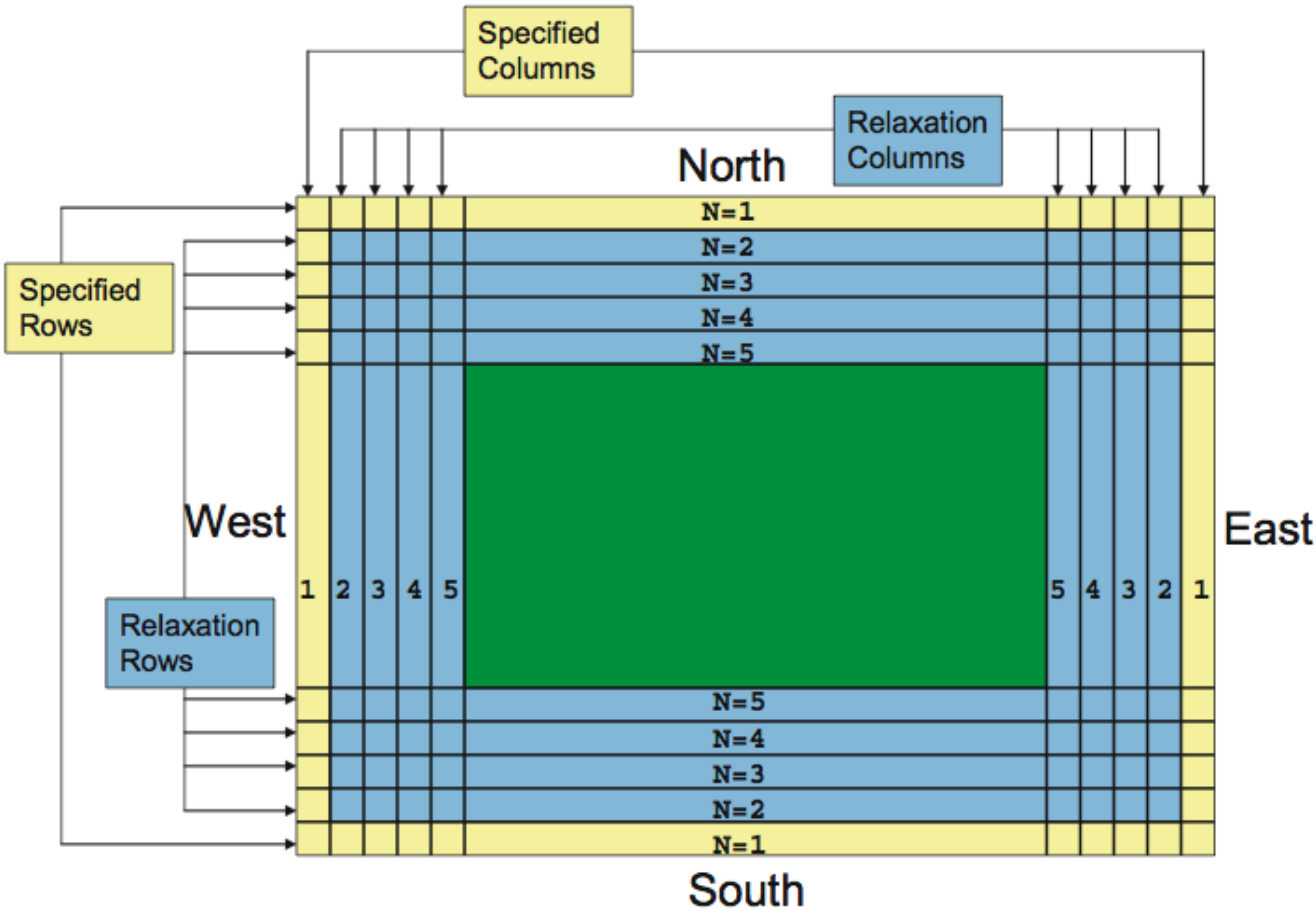
- The rectangular fine grid is coincident with a portion of the high-resolution grid that covers the entire coarse grid cell
- The nested domain can be placed anywhere within the parent domain and the nested grid cells will exactly overlap the parent cells at the coincident cell boundaries.
- Coincident parent/nest grid points eliminate the need for complex, generalized remapping calculations, and enhances model performance and portability.

Coarse Grid Staggering

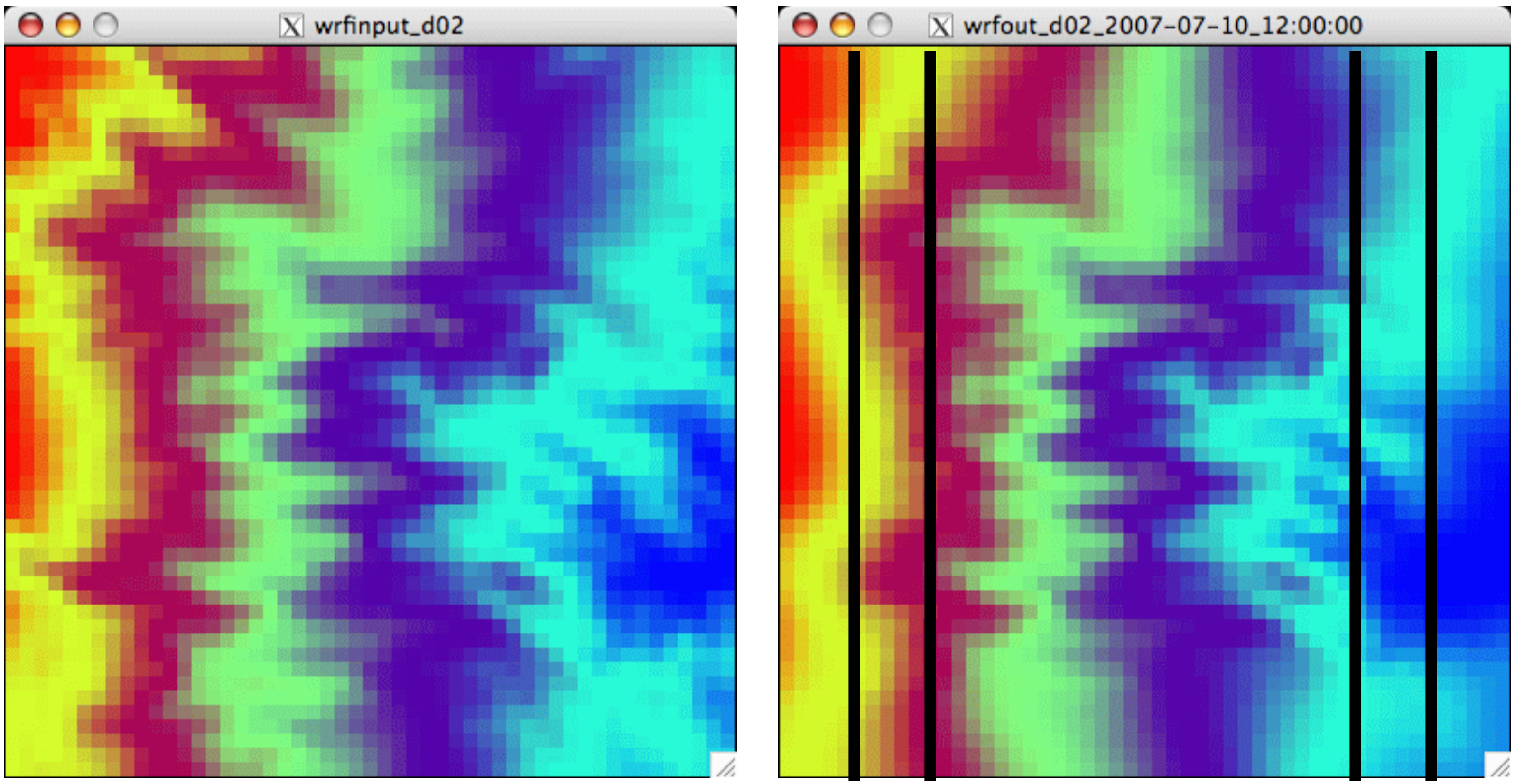
i_parent_start
j_parent_start



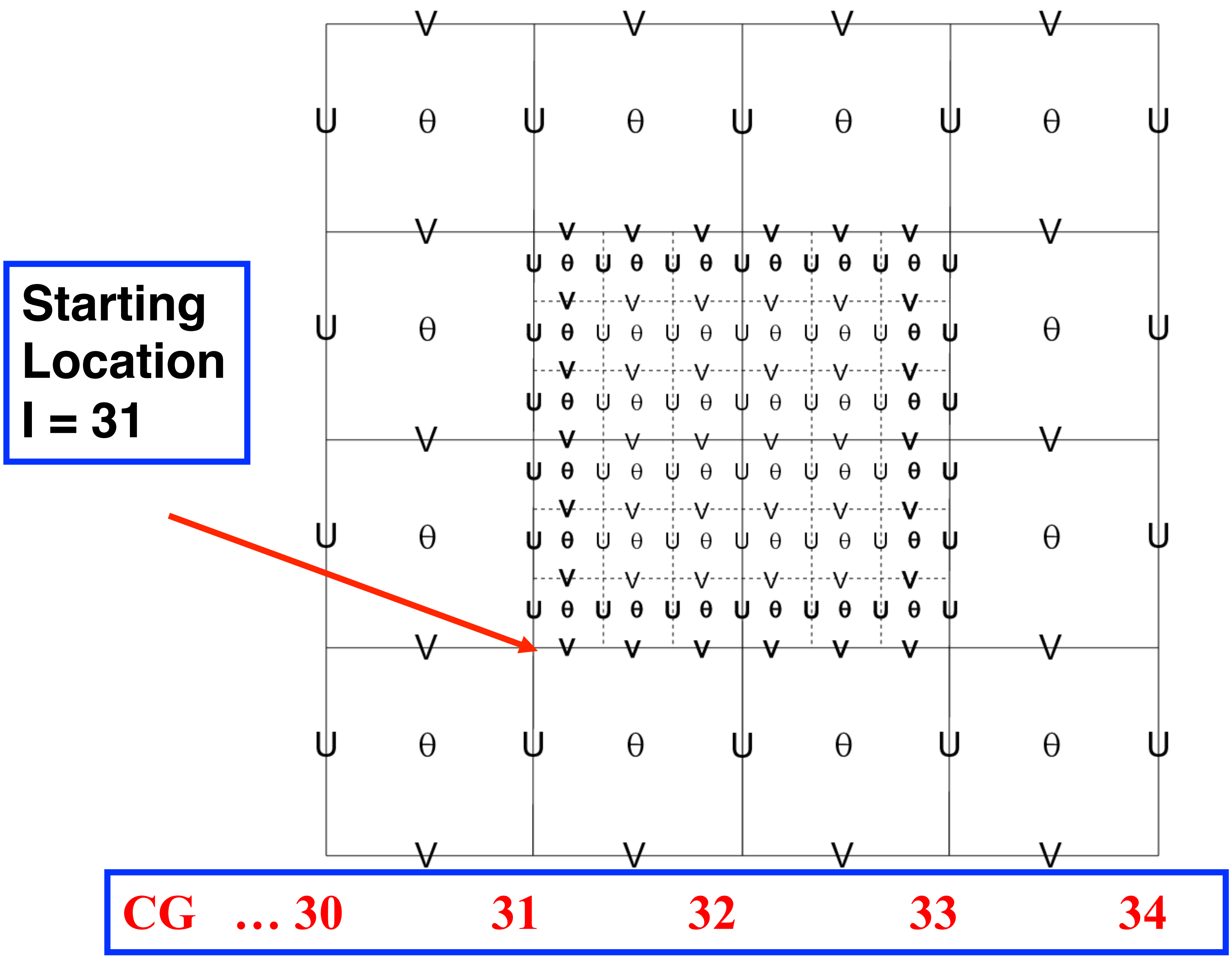
Real-Data Lateral Boundary Condition: Location of Specified and Relaxation Zones



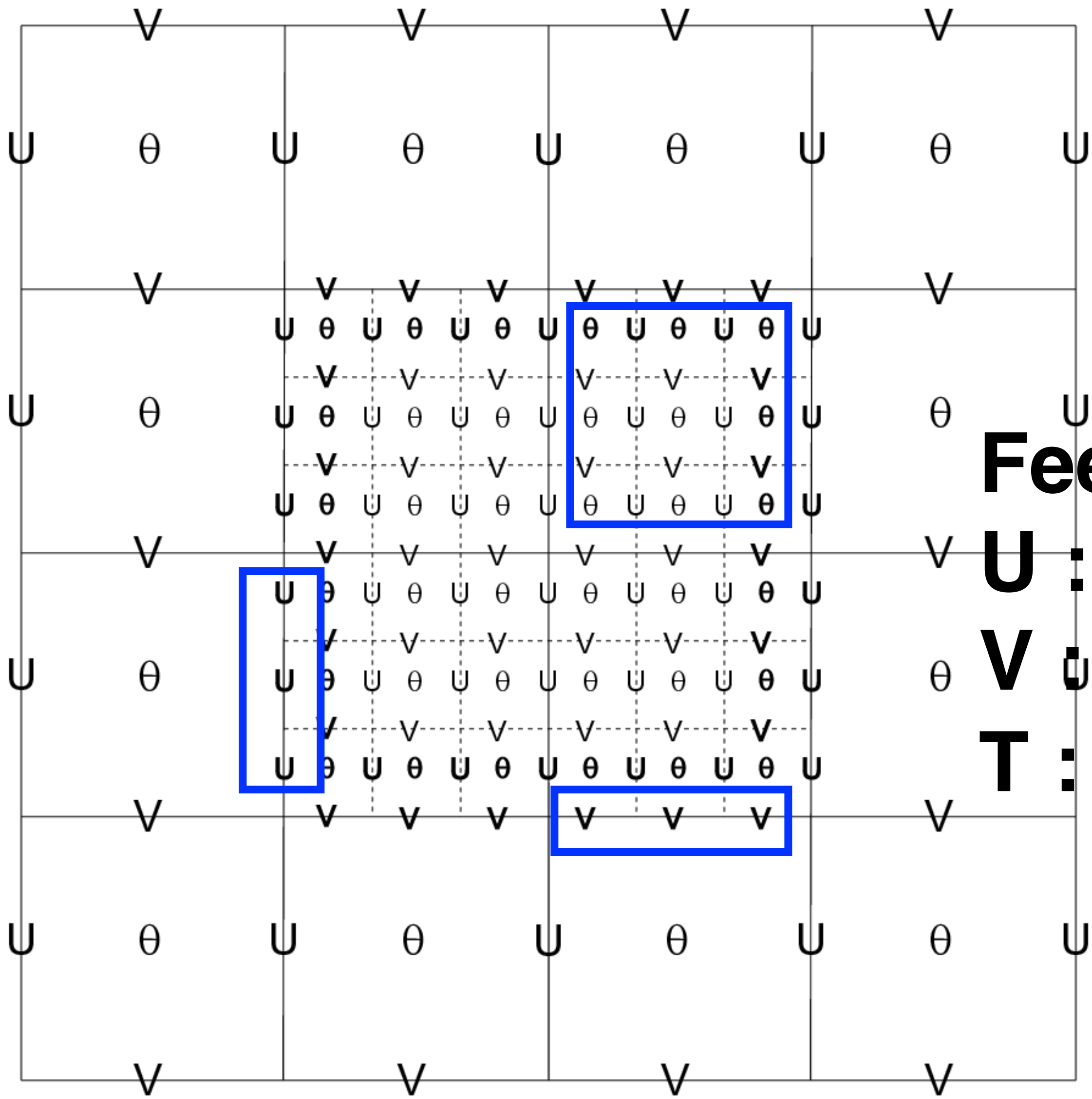
Lateral Smoothing



Coarse Grid Staggering 3:1 Ratio



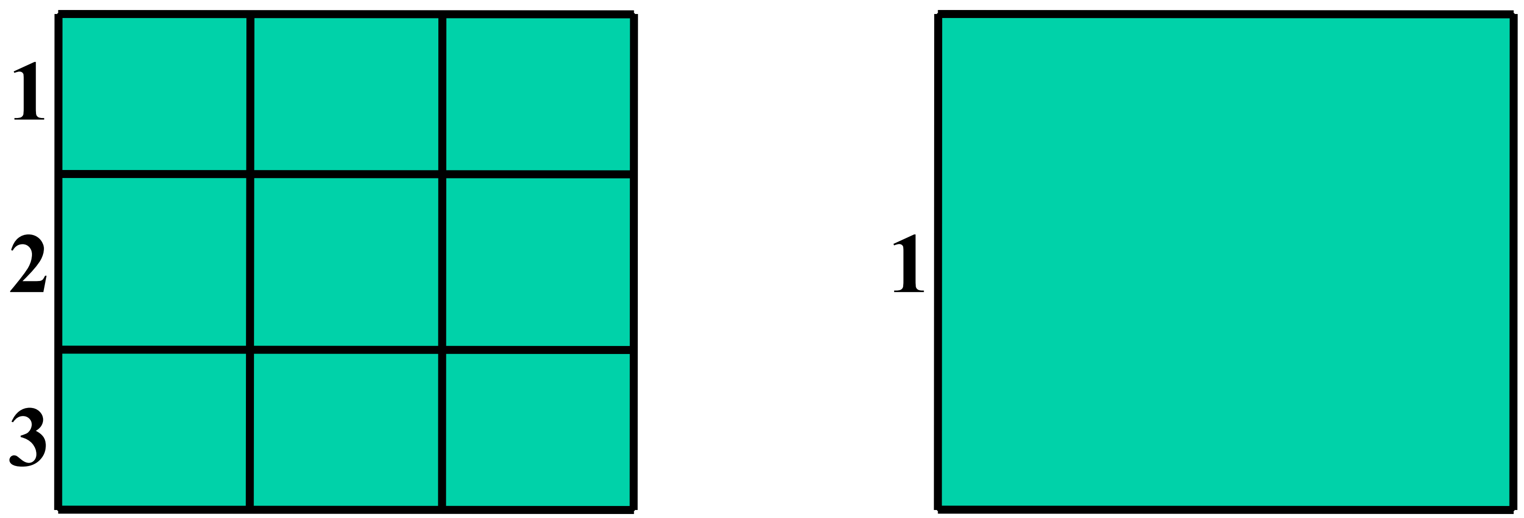
Coarse Grid Staggering 3:1 Ratio



Feedback:
U : column
V : row
T : cell

Coarse Grid Staggering 3:1 Ratio

Feedback:
U : column
V : row
T : cell



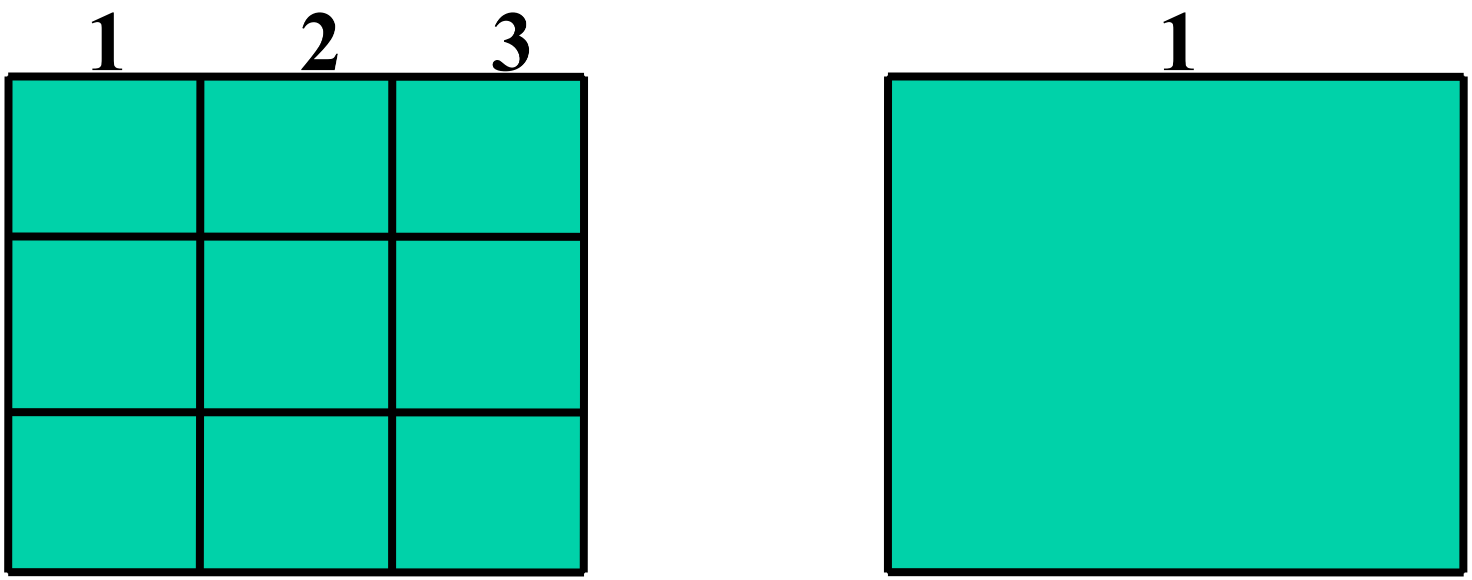
Coarse Grid Staggering 3:1 Ratio

Feedback:

U : column

V : row

T : cell



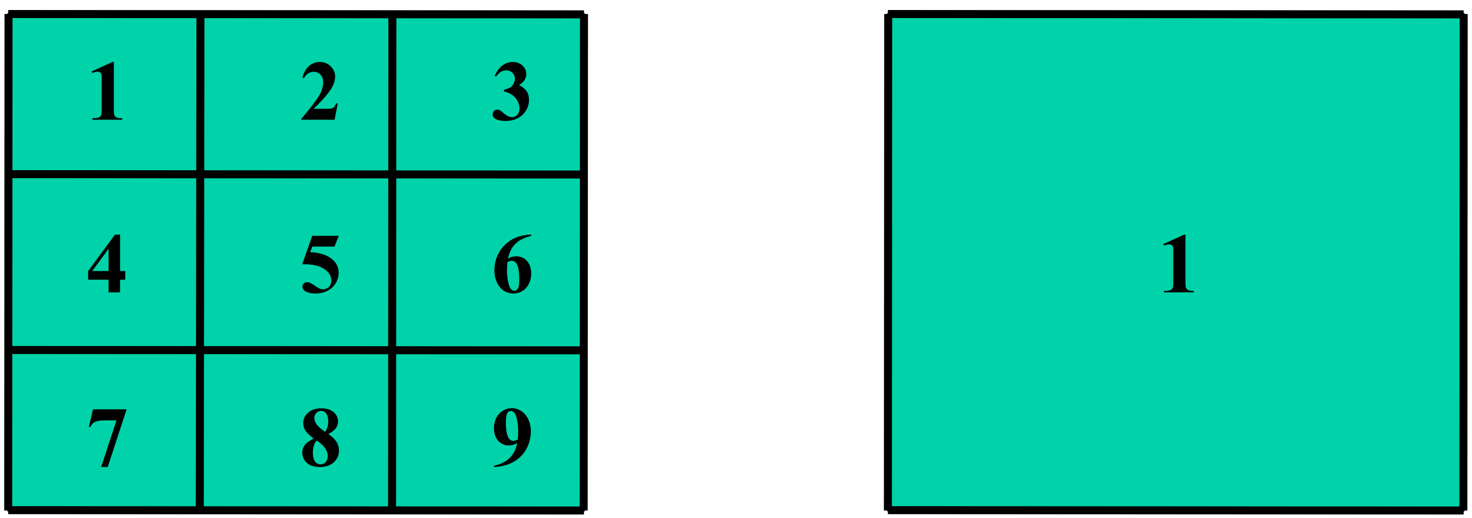
Coarse Grid Staggering 3:1 Ratio

Feedback:

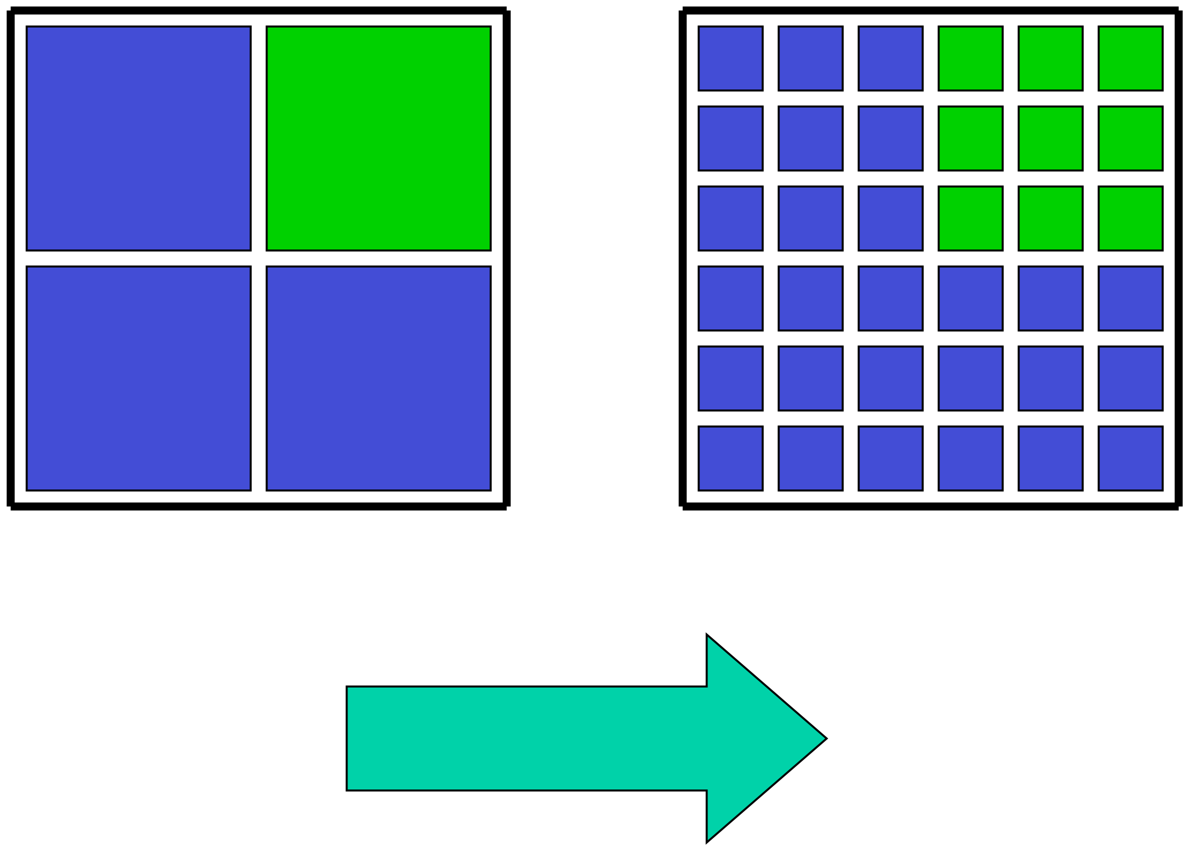
U : column

V : row

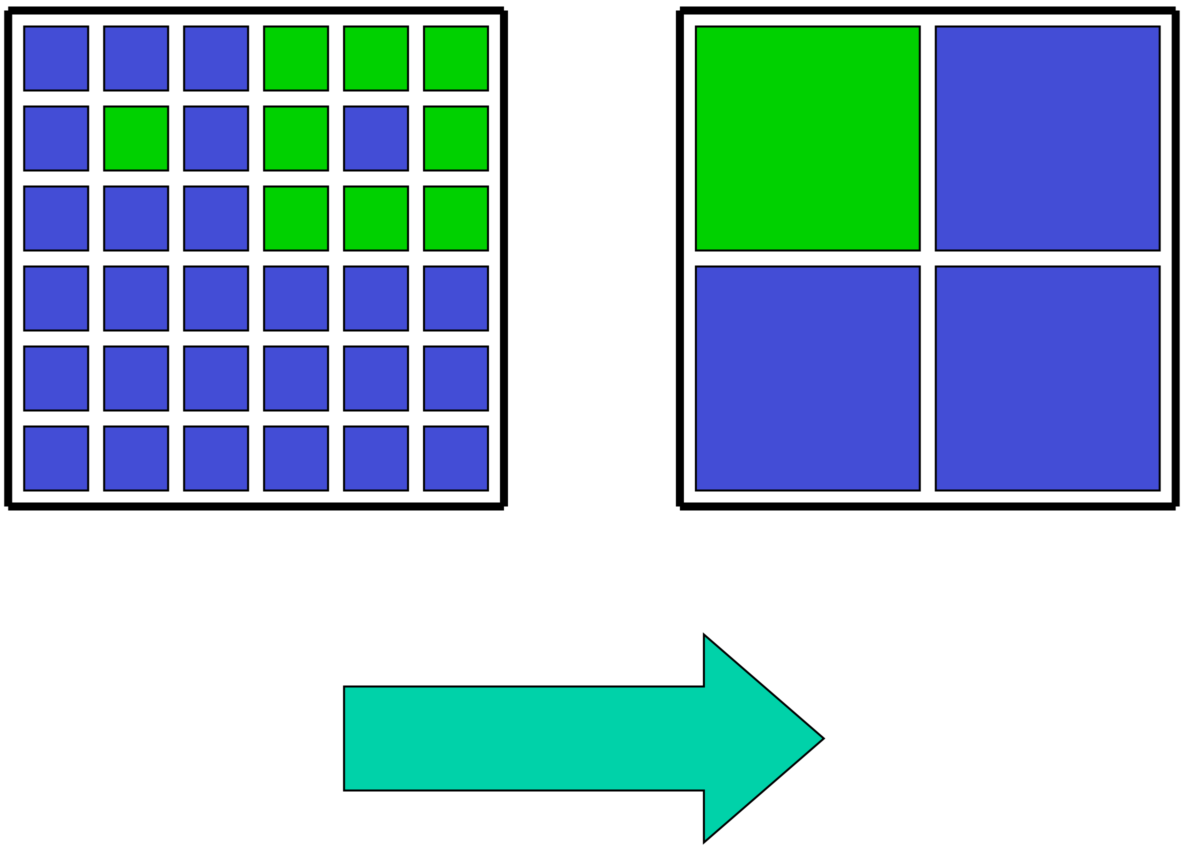
T : cell



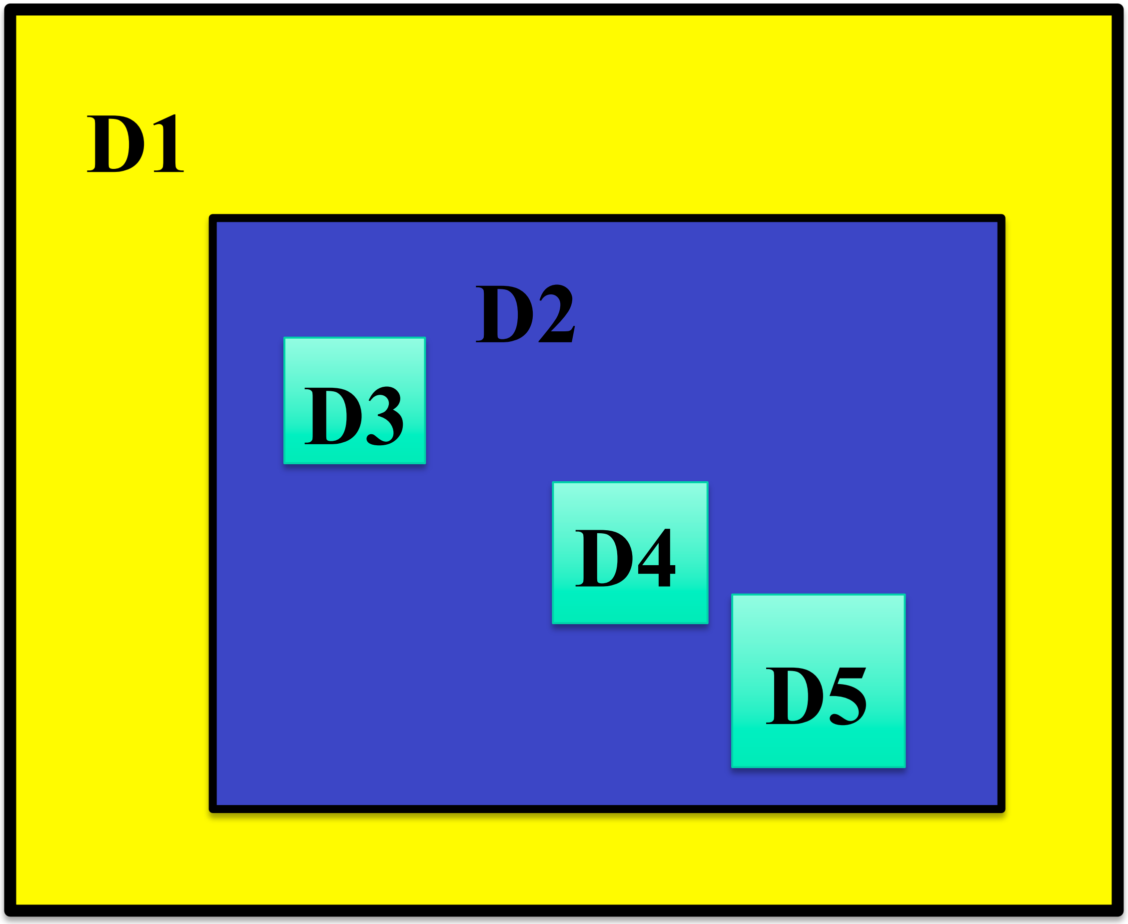
Masked Interpolation



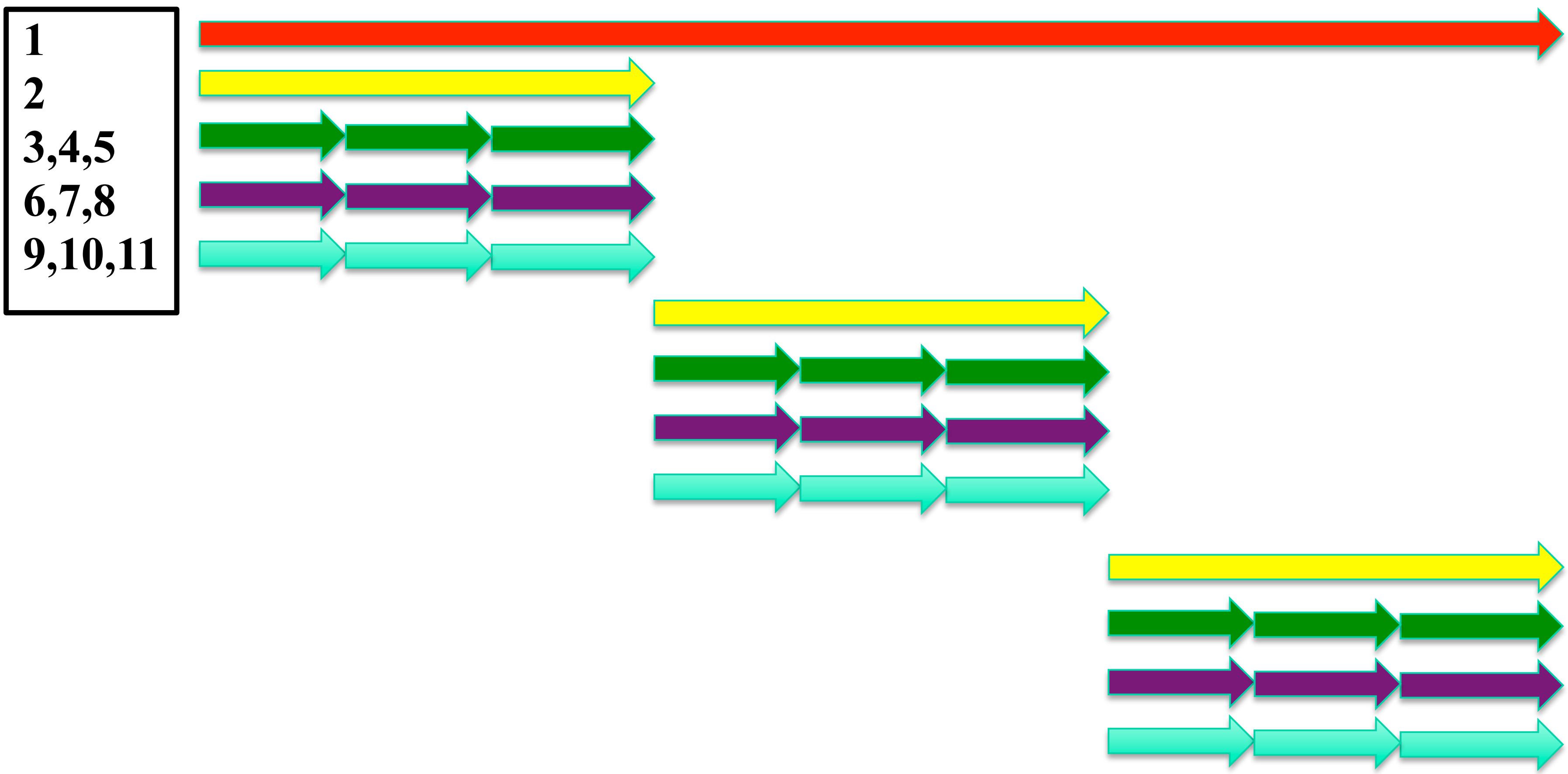
Masked Feedback



WRF 5-domain run: Domain 1 (a single 3 min dt), then Domain 2 (a single 1 min dt). Then Domain 3, in 20 s pieces up to 1 min. Then Domain 4, in 20 s pieces up to 1 min, and same with Domain 5.



WRF 5-domain run: Domain 1 (a single 3 min dt), then Domain 2 (a single 1 min dt). Then Domain 3, in 20 s pieces up to 1 min. Then Domain 4, in 20 s pieces up to 1 min, and same with Domain 5.



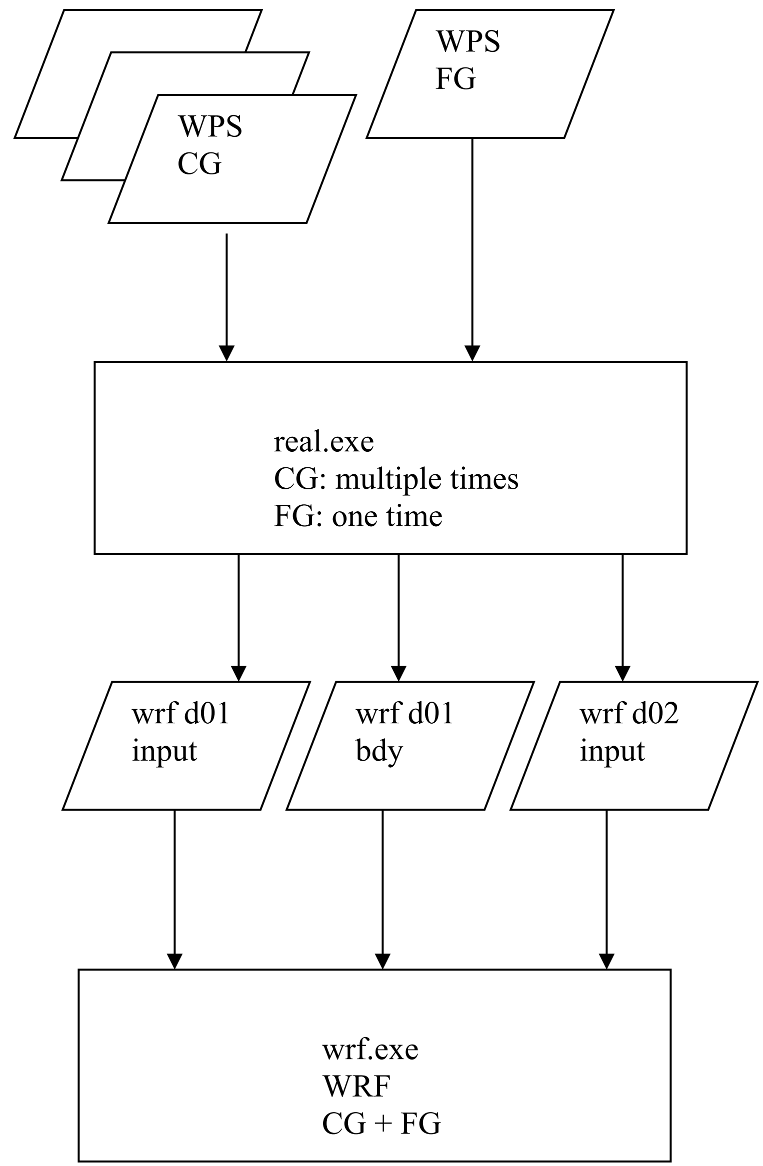
Concurrent Nesting with *n* Inputs

Coarse and fine grid domains must start at the same time, fine domain may end at any time

Feedback may be shut off to produce a 1-way nest (cell face and cell average)

Any integer ratio for coarse to fine is permitted, odd is usually chosen for real-data cases

Options are available to ingest only the static fields from the fine grid, with the coarse grid data horizontally interpolated to the nest



Concurrent Nesting with *n* Inputs

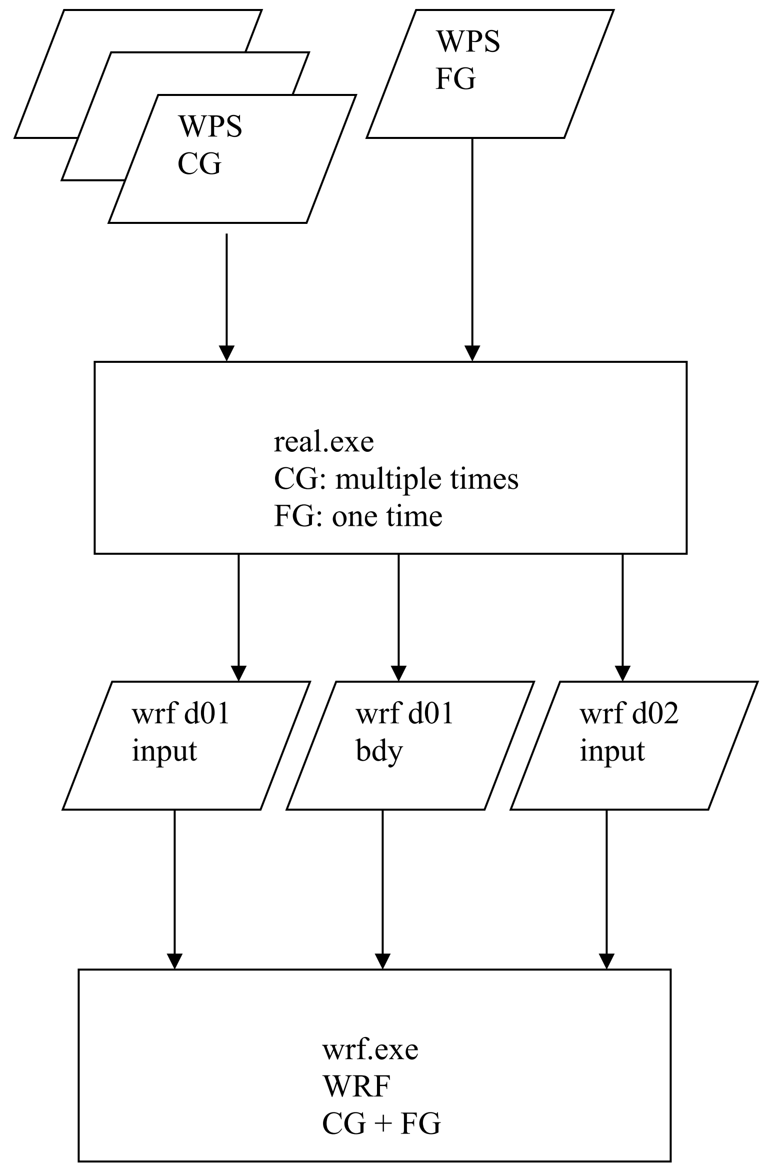
Restricted vertical nesting

Usually the same physics are run on all of the domains (excepting cumulus)

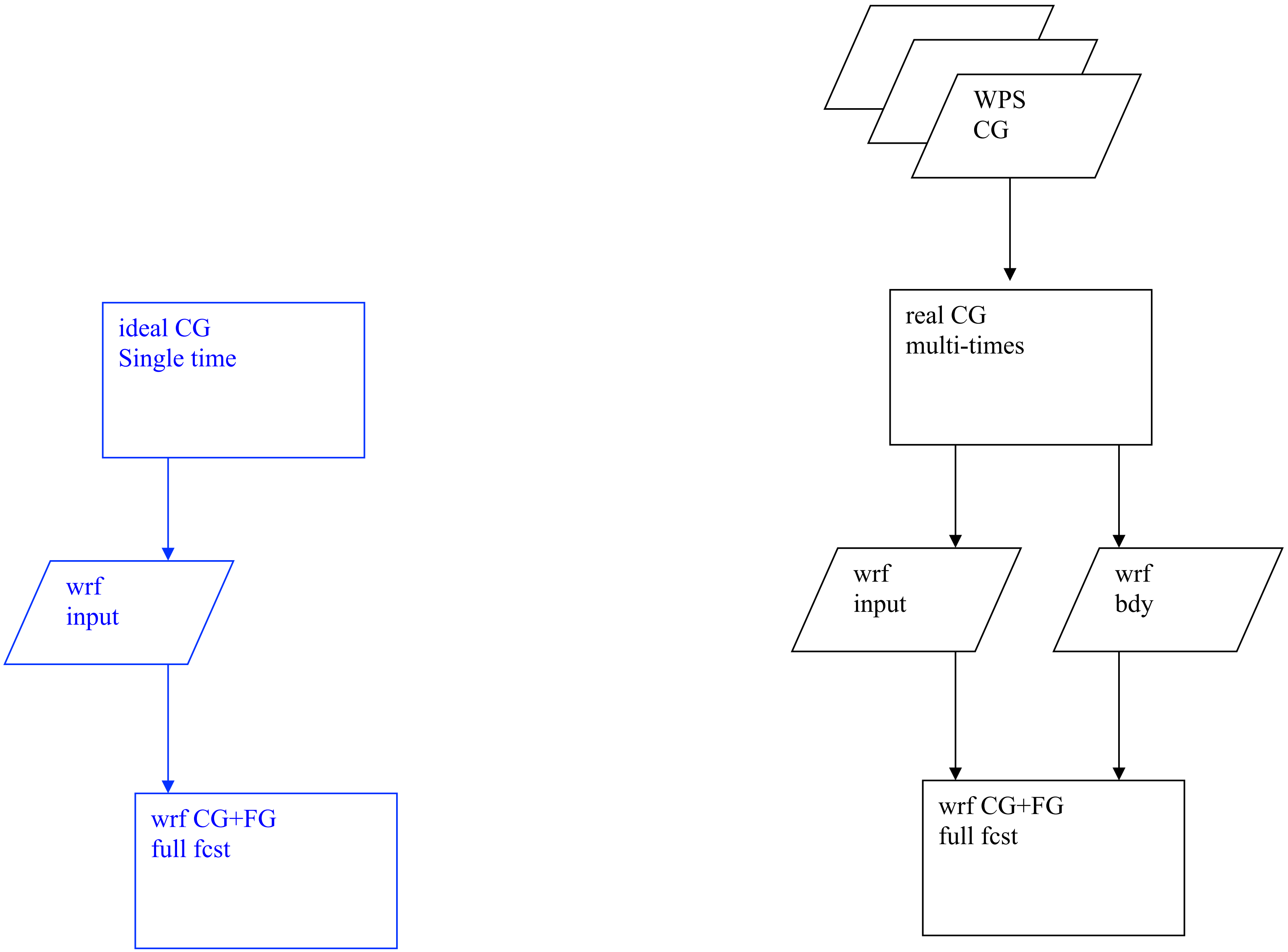
The grid distance ratio is not strictly tied to the time step ratio

Topography smoothly ramps from coarse grid to the fine grid along the interface along the nest boundary

All fine grids must use the nested lateral boundary condition



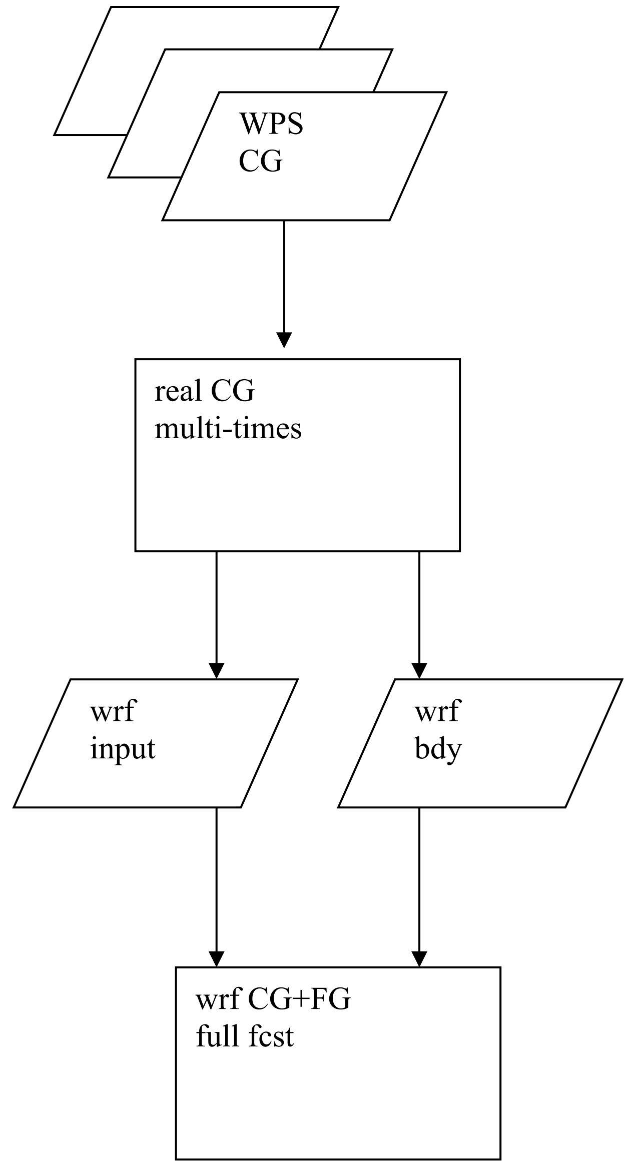
Concurrent Nesting with 1 Input



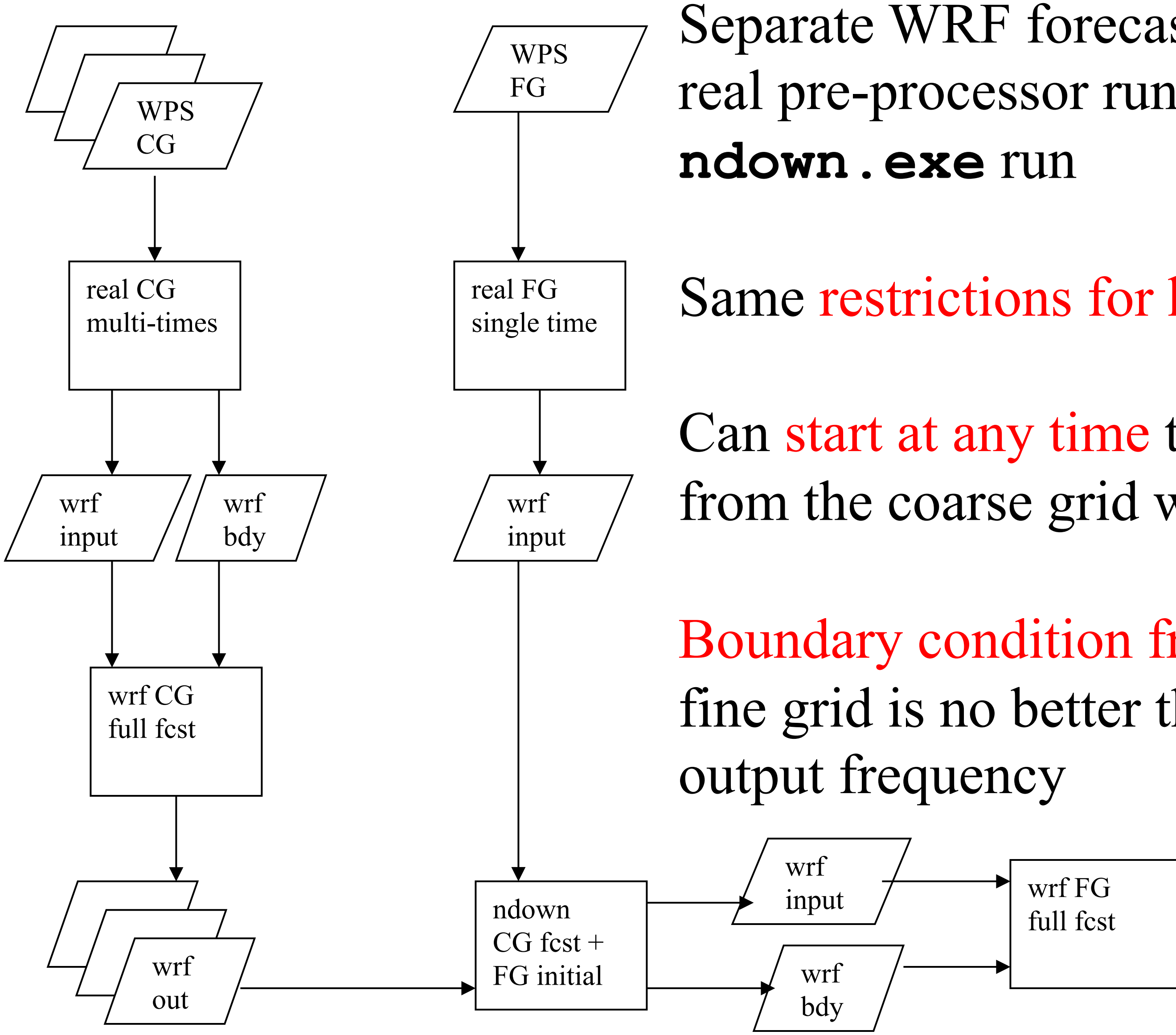
Concurrent Nesting with 1 Input

A single namelist column entry is tied to each domain

The horizontal interpolation method, feedback, and smoothing are largely controlled through the Registry file



ndown: Offline Nesting



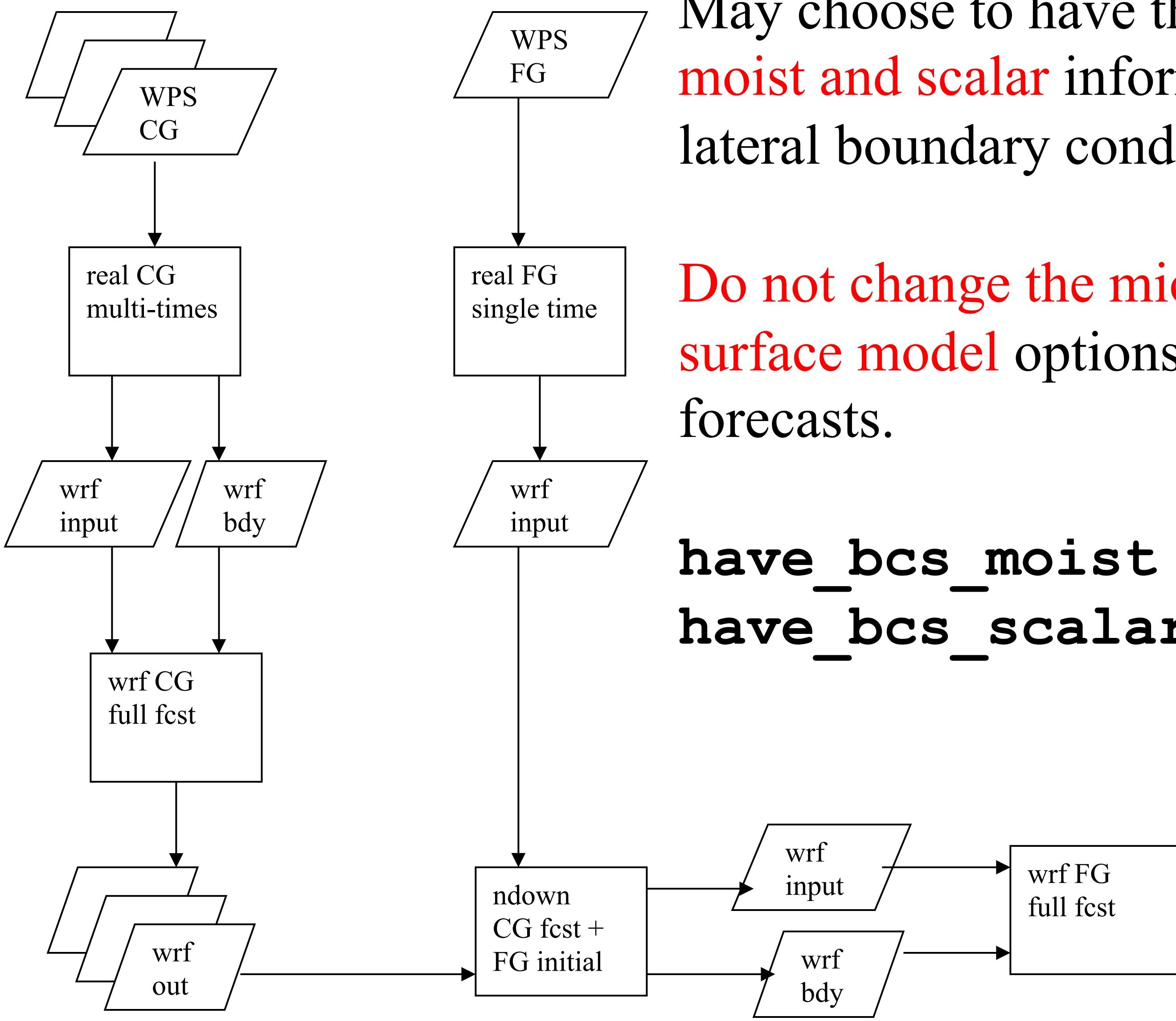
Separate WRF forecast runs, separate real pre-processor runs, intervening **`ndown.exe`** run

Same restrictions for horizontal nest ratios

Can start at any time that an output time from the coarse grid was created

Boundary condition frequency for the fine grid is no better than the coarse grid output frequency

ndown: Offline Nesting



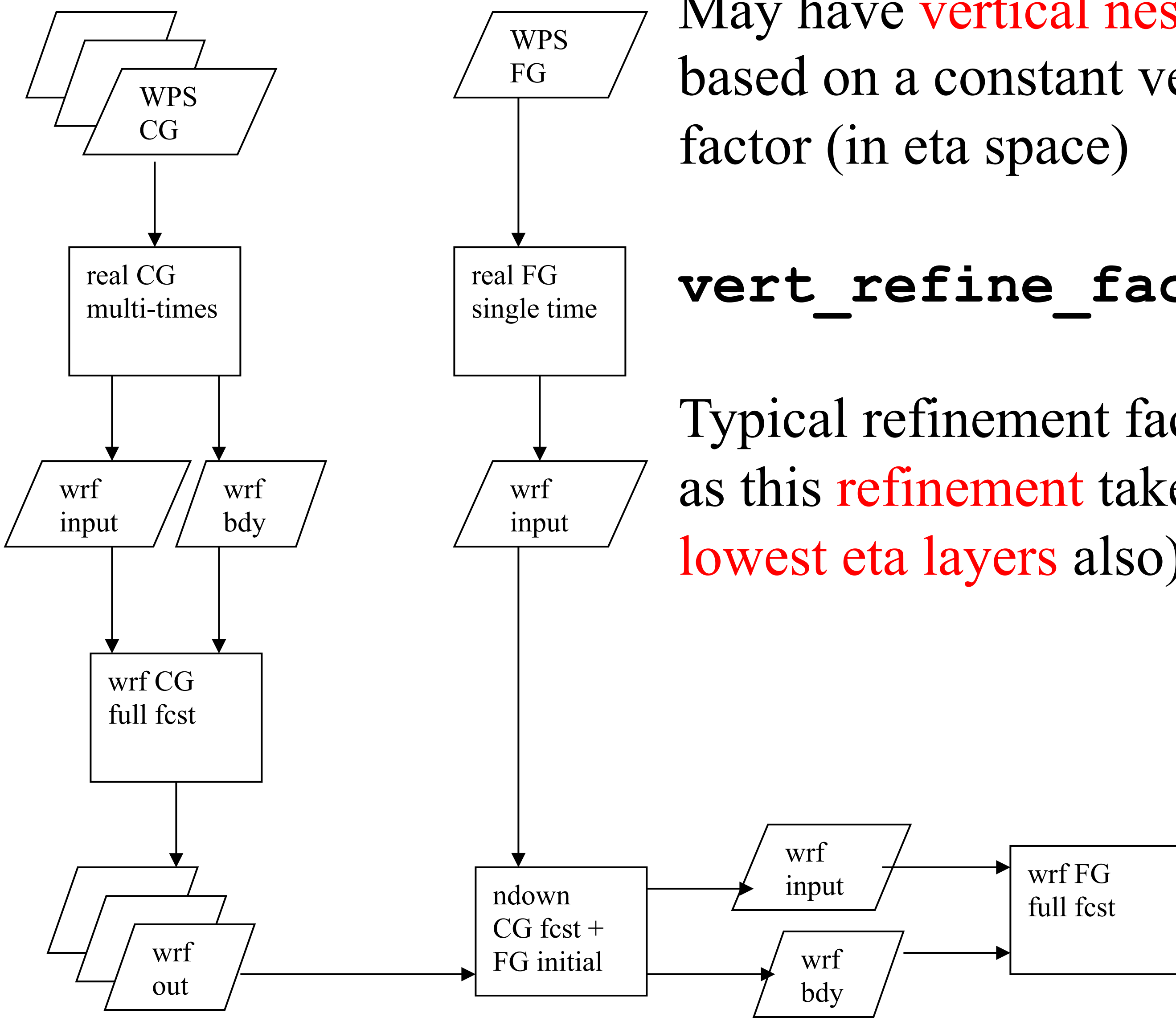
May choose to have the parent WRF model moist and scalar information used for the lateral boundary conditions.

Do not change the microphysics or land surface model options between the WRF forecasts.

have_bcs_moist

have_bcs_scalar

ndown: Offline Nesting



May have vertical nesting on the fine grid based on a constant vertical refinement factor (in eta space)

`vert_refine_fact`

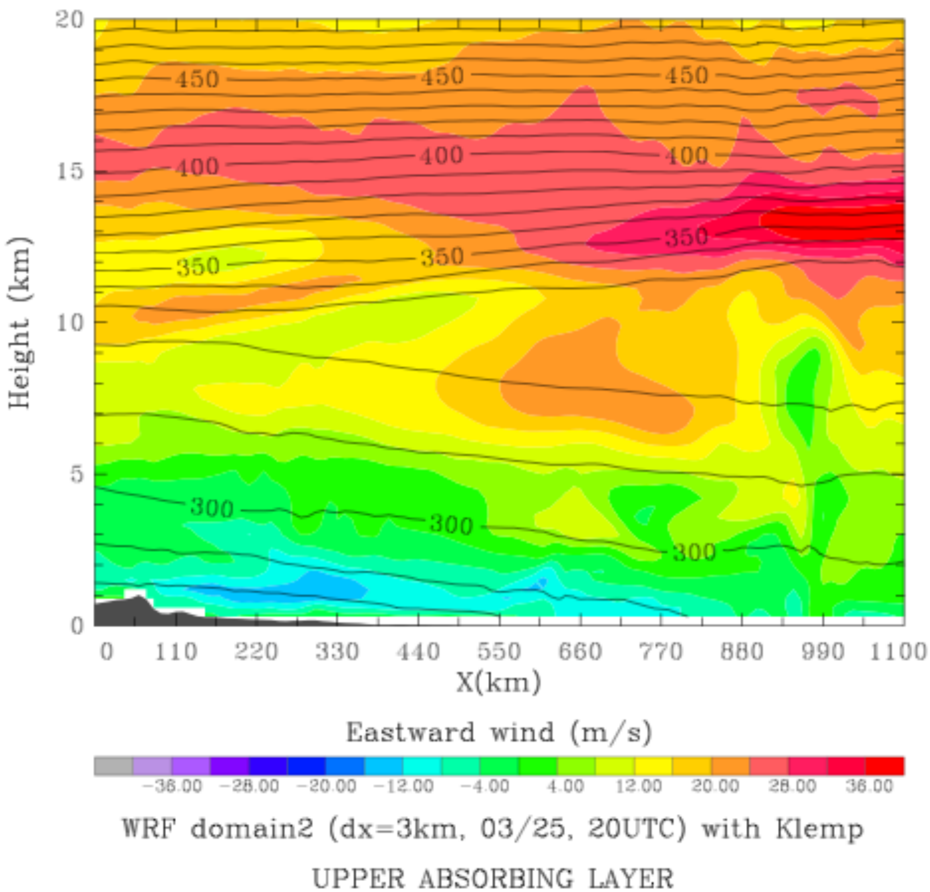
Typical refinement factors 2-5 (be careful, as this refinement takes place in the lowest eta layers also)

West East Cross section

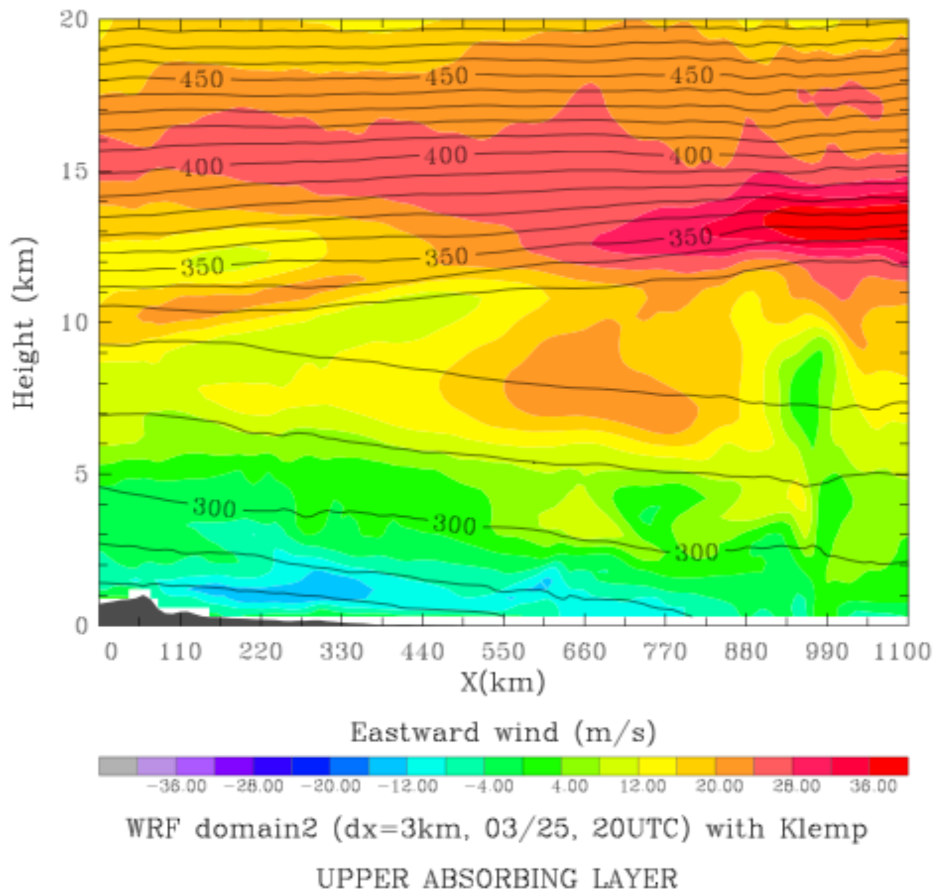
Shaded: v; Contour: theta

6-h Forecast, from Mohamed Moustaoui

Standard Levels



3x Refinement



Vertical Nesting

Starting with V3.7, the model allows different numbers of vertical levels on different domains. Usually, this is to include more levels on the inner domains.

This is a new feature and needs to be handled cautiously.

Restrictions:

- Only RRTM and RRTMG radiation schemes are available
- Real data cases only
- Static nest locations only

Vertical Nesting

```
&domains
max_dom                = 2,
e_vert                 = 35,      45,
vert_refine_method = 0,      2,
```

Vertical Nesting

```
eta_levels( 1:35) = 1., 0.993, 0.983, 0.97, 0.954, 0.934,
                    0.909, 0.88, 0.8406663, 0.8013327,
                    0.761999, 0.7226653, 0.6525755,
                    0.5877361, 0.5278192, 0.472514,
                    0.4215262, 0.3745775, 0.3314044,
                    0.2917579, 0.2554026, 0.2221162,
                    0.1916888, 0.1639222, 0.1386297,
                    0.1156351, 0.09525016, 0.07733481,
                    0.06158983, 0.04775231, 0.03559115,
                    0.02490328, 0.0155102, 0.007255059, 0.
```

Vertical Nesting

```
eta_levels(36:81) = 1.0000, 0.9946, 0.9875, 0.9789, 0.9685,
                    0.9562, 0.9413, 0.9238, 0.9037, 0.8813,
                    0.8514, 0.8210, 0.7906, 0.7602, 0.7298,
                    0.6812, 0.6290, 0.5796, 0.5333, 0.4901,
                    0.4493, 0.4109, 0.3746, 0.3412, 0.3098,
                    0.2802, 0.2524, 0.2267, 0.2028, 0.1803,
                    0.1593, 0.1398, 0.1219, 0.1054, 0.0904,
                    0.0766, 0.0645, 0.0534, 0.0433, 0.0341,
                    0.0259, 0.0185, 0.0118, 0.0056, 0.
```

What are those "usdf" Options

```
state real u ikjb dyn_em 2 X \
  i01rhusdf=(bdy_interp:dt) \
  "U" "x-wind component" "m s-1"
```

"f" defines what lateral boundary forcing routine (found in **`share/interp_fcn.F`**) is utilized, colon separates the additional fields that are required (fields must be previously defined in the Registry)

Called at beginning of each set of child time steps, has parent and child information available – could be used with SST.

What are those “usdf” Options

```
state real landmask ij misc 1 -  \
  i012rhd=(interp_fcnm)u=(copy_fcnm)\
  "LANDMASK" "LAND MASK (1=LAND, 0=WATER)"
```

“u” and “d” define which feedback (up-scale) and horizontal interpolation (down-scale) routines (found in **`share/interp_fcn.F`**) are utilized

Default values (i.e. not a subroutine name listed in the parentheses) assume non-masked fields

What are those “usdf” Options

```
state real ht ij  misc  1  - i012rhdus  "HGT" \
  "Terrain Height"   "m"
```

“s” if the run-time option for smoothing is activated, this field is to be smoothed - only used for the parent of a nest domain, smoothing is in the area of the nest, excluding the outer row and column of the nest coverage

Whether or not smoothing is enabled is a run-time option from the namelist – **smoothing can always be turned off without introducing any problems**

Special IO Stream #2 Fields

```
state real msft ij  misc  1 -  \
  i012rhdu=(copy_fcnm)  "MAPFAC_M"   \
  "Map scale factor on mass grid" ""

state real msfu ij  misc  1 X  \
  i012rhdu=(copy_fcnm)  "MAPFAC_U"   \
  "Map scale factor on u-grid" ""

state real msfv ij  misc  1 Y  \
  i012rhdu=(copy_fcnm)  "MAPFAC_V"   \
  "Map scale factor on v-grid" ""
```

Nesting Suggestions

- The size of the nested domain may need to be chosen with computing performance in mind.
- Assuming a 3:1 ratio and the same number of grid cells in the parent and nest domains, the fine grid will require 3x as many time steps to keep pace with the coarse domain.
- A simple nested domain forecast is approximately 4x the cost of just the coarse domain.
- Don't be *cheap* on the coarse grid, doubling the CG points results in only a 25% nested forecast time increase.

Nesting Suggestions

- Example: assume 3:1 nest ratio

If the nest has the same number of grid cells, then the **amount of CPU** to do a single time step for a coarse grid (CG) and a fine grid step (FG) is **approximately the same**.

Since the fine grid (3:1 ratio) has 1/3 the grid distance, it requires 1/3 the model time step. Therefore, the FG requires 3x the CPU to catch up with the CG domain.

Nesting Suggestions

- Example: assume 3:1 nest ratio

If you try to cover the SAME area with a FG domain as a CG domain, you need **(ratio)^2 grid points**.

With the associated FG time step ratio, you require a **(ratio)^3**.

With a 3:1 ratio, a FG domain covering the same area as a CG domain **requires 27x CPU**.

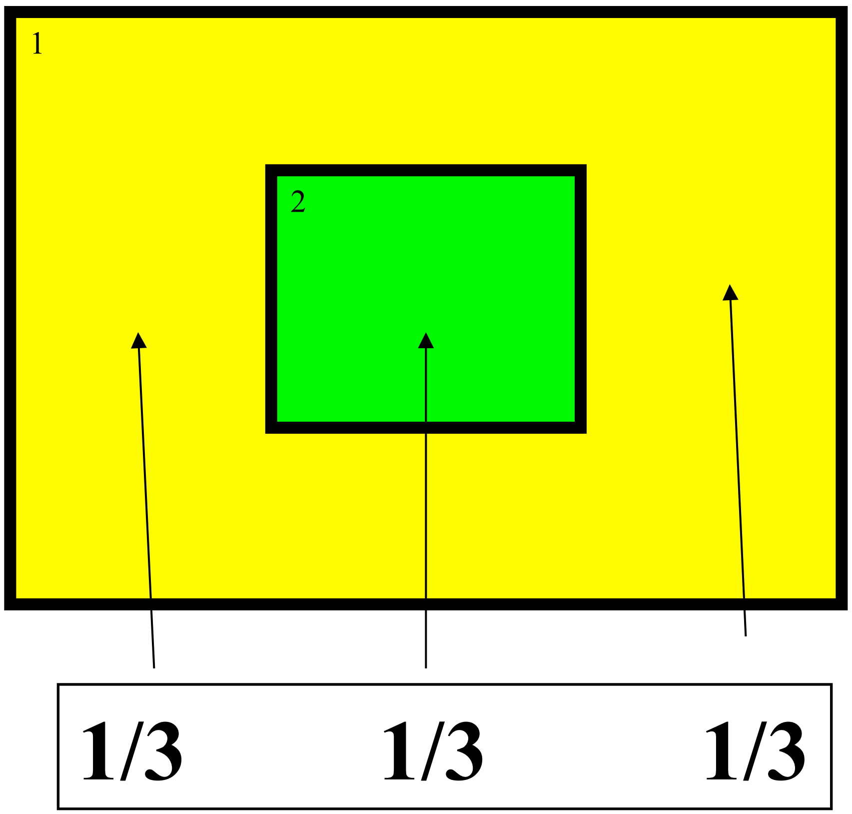
Nesting Suggestions

- Example: assume **10:1 nest ratio**

To change your test case from 50-km resolution to a finer 5-km resolution would be **1000x more** expensive.

Nesting Suggestions

- The minimum distance between the nest boundary and the parent boundary is FOUR grid cells
- You should have a **MUCH** larger buffer zone
- It is not unreasonable to have approximately 1/3 of your coarse-grid domain surrounding each side of your nest domain



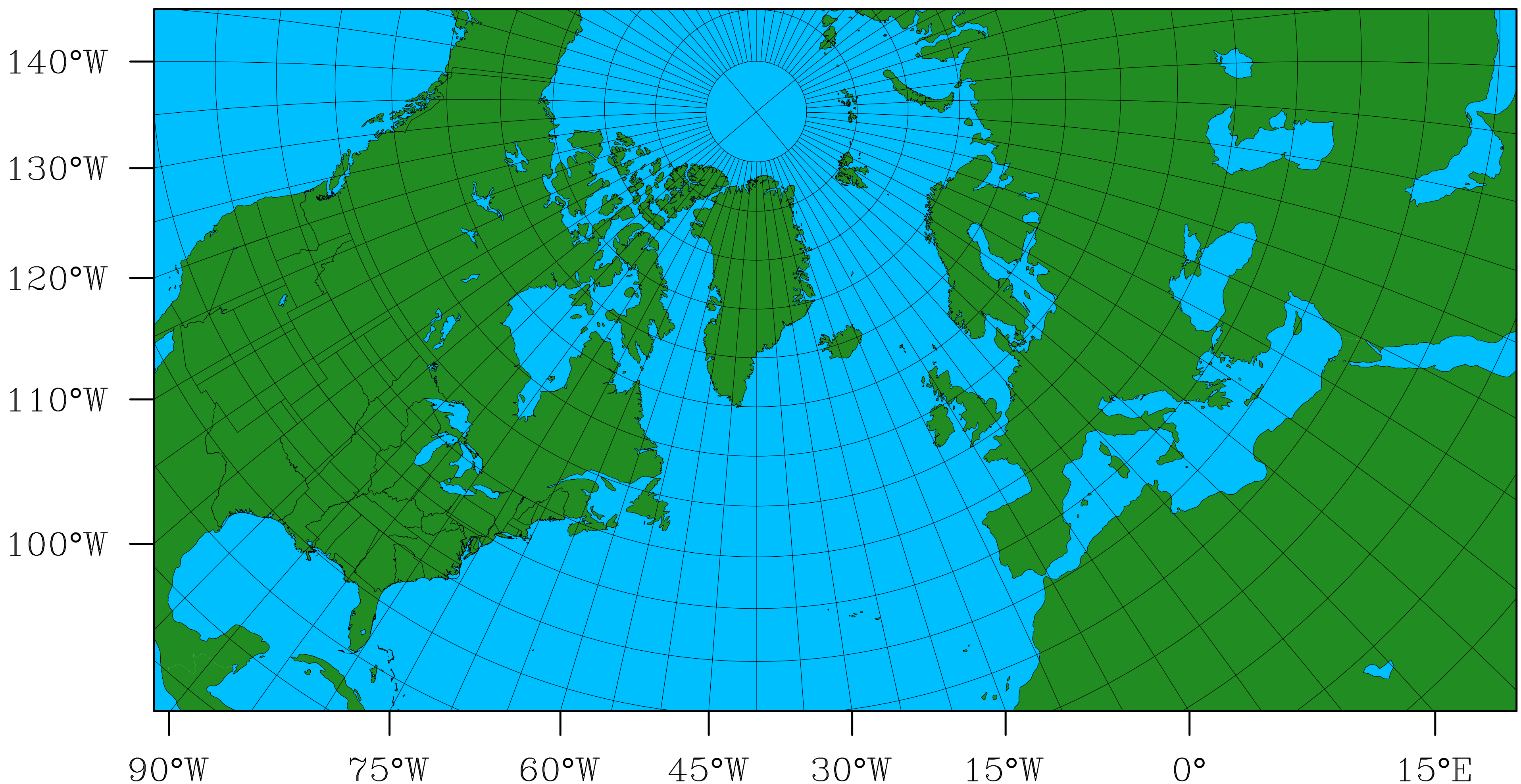
Nesting Suggestions

- Start with designing your inner-most domain. For a traditional forecast, you want everything important for that forecast to be entirely contained inside the domain.
- Then start adding parent domains at a 3:1 or 5:1 ratio. A parent should not have a smaller size (in grid points). Keep adding domains until the most coarse WRF grid has a no more than a 3:1 to 5:1 ratio to the external model (first guess) data.

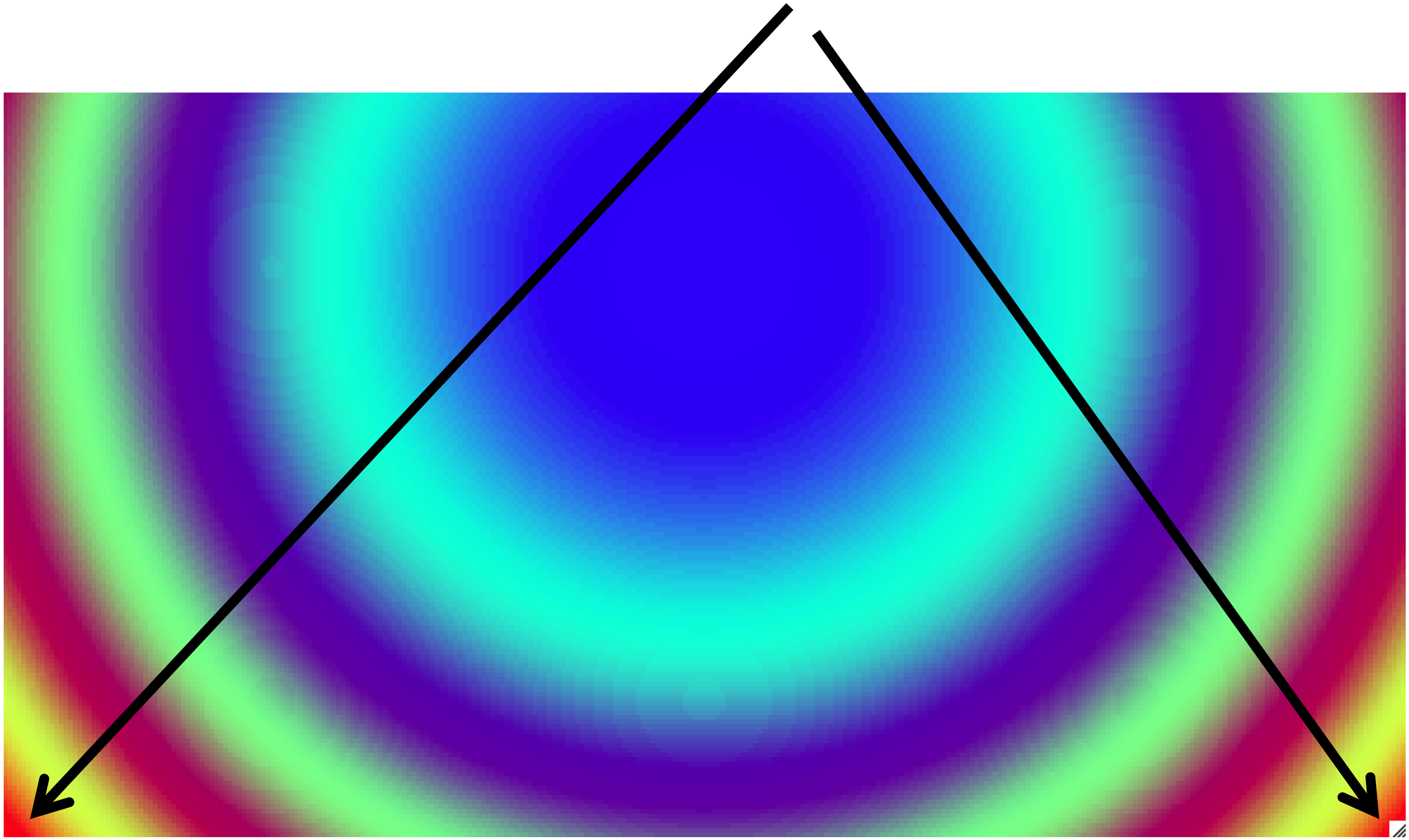
Nesting Suggestions

- Larger domains tend to be better than smaller domains.
- A 60 m/s parcel moves at > 200 km/h. A 2-km resolution grid with 100x100 grid points could have most of the upper-level initial data swept out of the domain within a couple of hours.

Nesting Suggestions



Map factors > 1.6



Nesting Suggestions

- The most-coarse domain may have a geographic extent that causes large map factors.

```
time_step              = 300 (BLOWS UP)
dx                     = 45000,15000,5000
grid_id                = 1,    ,2    ,3
parent_id              = 0,    ,1    ,2
parent_grid_ratio      = 1,    ,3    ,3
parent_time_step_ratio = 1,    ,3    ,3
```

Nesting Suggestions

- Reducing the time step so that the coarse grid is stable makes the model too expensive. 1.6x

```
time_step                = 180 (STABLE, PRICEY)
dx                       = 45000,15000,5000
grid_id                  = 1,    ,2    ,3
parent_id                = 0,    ,1    ,2
parent_grid_ratio        = 1,    ,3    ,3
parent_time_step_ratio   = 1,    ,3    ,3
```

Nesting Suggestions

- Only reduce the time step on the coarse grid, and keep the fine grid time steps at their approx original values.

```
time_step              = 180 (STABLE, CHEAP)
dx                     = 45000,15000,5000
grid_id                = 1,    ,2    ,3
parent_id              = 0,    ,1    ,2
parent_grid_ratio      = 1,    ,3    ,3
parent_time_step_ratio = 1,    ,2    ,3
```

Nesting Suggestions

- Model time step is always proportional to the time step of the most coarse grid.
- The coarse grid is the only grid impacted with large map factors: **`dt(s) = 6*dx(km)`** but the nominal grid distance needs to be scaled:

  dt(s) = 6*dx(km) / MAX (map factor in domain)
- Reducing the coarse grid time step does not significantly reduce model performance if you can tweak the time step ratio.

Nesting Suggestions

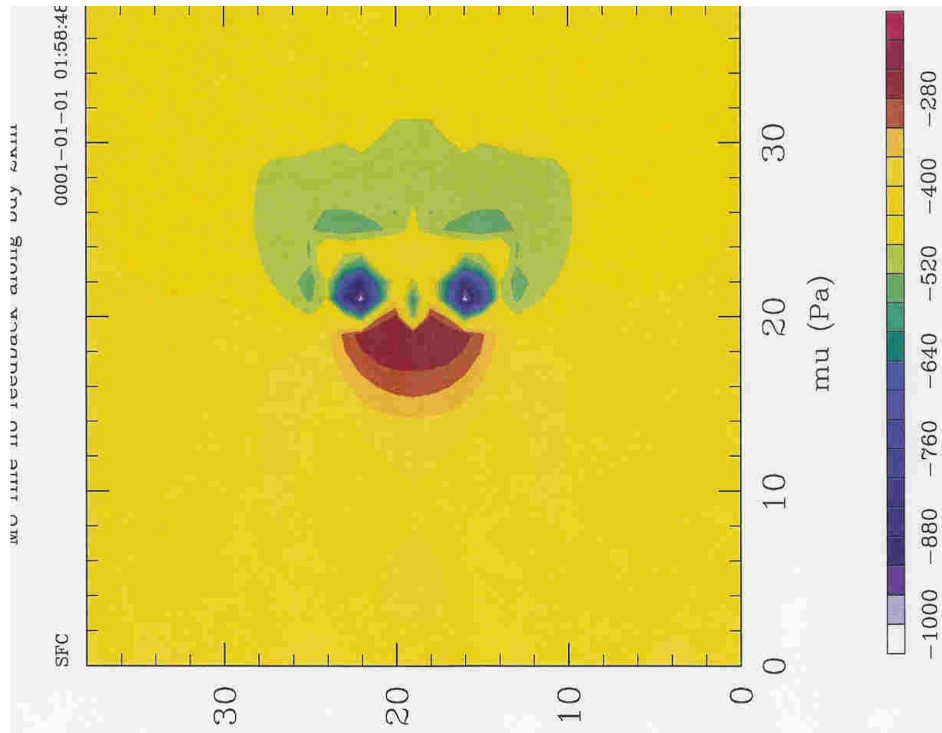
- The time step ratio and grid distance ratio are not necessarily identical, and may used effectively when large map factors in the coarse grid domain force a time step reduction for stability.
- If map factors are causing stability troubles, it is usually only the most coarse grid that is impacted since the fine grid is usually in the middle of the domain.

Nesting Suggestions

- Set up domain first to provide good valid forecast, then deal with efficiency
- Selecting a set of domains with the reason "it is all I can afford" gets you into trouble
- Numerically stable and computationally expedient do not imply scientifically or physically valid

Review

- Nesting: Journalism 101: Who, what, why, when, where
- Domains
  - OK *vs* semi-OK *vs* not OK
  - Variable staggering CG to FG
  - Lateral forcing
  - Feedback
  - Masked interpolation
  - Time stepping for multi-domain
- Concurrent *vs* Offline Nesting
- Registry
  - U D F S
  - i2
- Some suggestions
  - Performance
  - Location, location, location
  - Inside out, start with inner domain
  - Go big or go home
  - Map factors, stability, time step



Nesting PART 2

- Nesting steps inside of WRF
- Available source code options
- Choosing the nested interpolation type
- Building automatically accessed routines

Nesting Sequence Inside of WRF

- The WRF model always has the parent domain integrate a single time step, then the code checks to see if a child domain exists (valid time)

- The parent has current (t+dt) information stored in the _2 variables and the information from the previous time step stored in the _1 variables (for example t_1, t_2, etc).

- These two time levels of data allow the lateral boundary conditions for the fine grid to be handled similarly to that of the coarse grid: an initial value (the old time, _1) and a tendency to get to the next time are required

Nesting Sequence Inside of WRF

- The initial value and the tendency from the parent domain are horizontally interpolated onto the child domain

- For a nest ratio of 3:1, then three child time steps are required to get to the parent current time. The tendency during these three child time steps along the lateral boundaries remains constant

- At the end of the last child time step required to get to the parent's current time, for a two-way nest, the child information feedsback to the parent domain

Available Source Code Options

- The nesting inside of WRF requires a few types of routines:
  - Horizontally interpolate the parent to the child
  - Generate the lateral boundary conditions for the child
  - Feed information from the child back to the parent
  - Optionally smooth the area in the parent covered by the child domain after feedback
- All of these options are selected through the Registry

Available Source Code Options

- In the Registry, the nesting options are located with the I/O flags

```
Latitude: du=(copy_fcnm)
U: usdf=(bdy_interp:dt)
TSK: d=(interp_mask_field:lu_index,iswater)u=(copy_fcnm)
LANDMASK: d=(interp_fcnm_imask)u=(copy_fcnm)
SST: d=(interp_mask_field:lu_index,iswater)
```

- but could be as complicated as

```
SST: d=(interp_mask_field:lu_index,iswater)\
f=(p2c_mask:lu_index,tslb,num_soil_layers,iswater)
```

Available Source Code Options

- The syntax for horizontal interpolation from the parent to the child is “d” for “down”

```
d=(subroutine_name: optional arguments, comma separated)
```

- Default is interp_fcn

- The “d” option is handled only once per domain, at initialization

Available Source Code Options

- The syntax for feedback from the child back to the parent is "u" for "up"

```
d=(subroutine_name: optional arguments, comma separated)
```

- Default is copy_fcn

- The "u" is processed in the WRF model after the last required fine grid time step brings the child domain up to the same time as the parent

Available Source Code Options

- The syntax for the lateral boundary tendency computation is "f" for LBC "forcing"

```
f=(subroutine_name: optional arguments – typically time step)
```

- Default is interp_bdy (but specified because the time step argument is always used)

- The lateral boundary condition shares the "f" option. Any domain that would like to have the child domain given information at the end of each parent time step (such as lateral boundaries), may use the "f=()" Registry option. Some developers have subroutines that interpolate a child domain from the parent at EACH parent time step (SSTs and perturbations from SKEBS are examples)

Choosing the Nested Interpolation Type

- At run-time, the user may select the order of the horizontal interpolation to be used

```
&domains
interp_method_type = 1: bilinear
                     2: sint
                     3: nearest neighbor
                     4: quadratic
/
```

- The same order/type of interpolator is used the initial horizontal interpolation and the subsequent lateral boundary interpolation

Building Automatically Accessed Routines

- The registry program manufactures a default template for the subroutine call.

```
SUBROUTINE interp_fcn                                   &
   ( cfld,                                              &
     cids, cide, ckds, ckde, cjds, cjde,                &
     cims, cime, ckms, ckme, cjms, cjme,                &
     cits, cite, ckts, ckte, cjts, cjte,                &
     nfld,                                              &
     nids, nide, nkds, nkde, njds, njde,                &
     nims, nime, nkms, nkme, njms, njme,                &
     nits, nite, nkts, nkte, njts, njte,                &
     shw,                                               &
     imask,                                             &
     xstag, ystag,                                      &
     ipos, jpos,                                        &
     nri, nrj                                           )
```

Building Automatically Accessed Routines

- The lateral boundary routines (the “f=()” option) always get the eight boundary arrays appended (total of 16 arrays, 8 for parent, 8 for child).
- Any extra variables are ALWAYS tagged on to the end of the subroutine, and always in pairs: parent and child (for example: time step, land mask, etc).
- The user may place the new routine (called by the name given in the Registry file) in the interp_fcn.F file

Nesting PART 2

- Nesting steps inside of WRF
- Available source code options
- Choosing the nested interpolation type
- Building automatically accessed routines