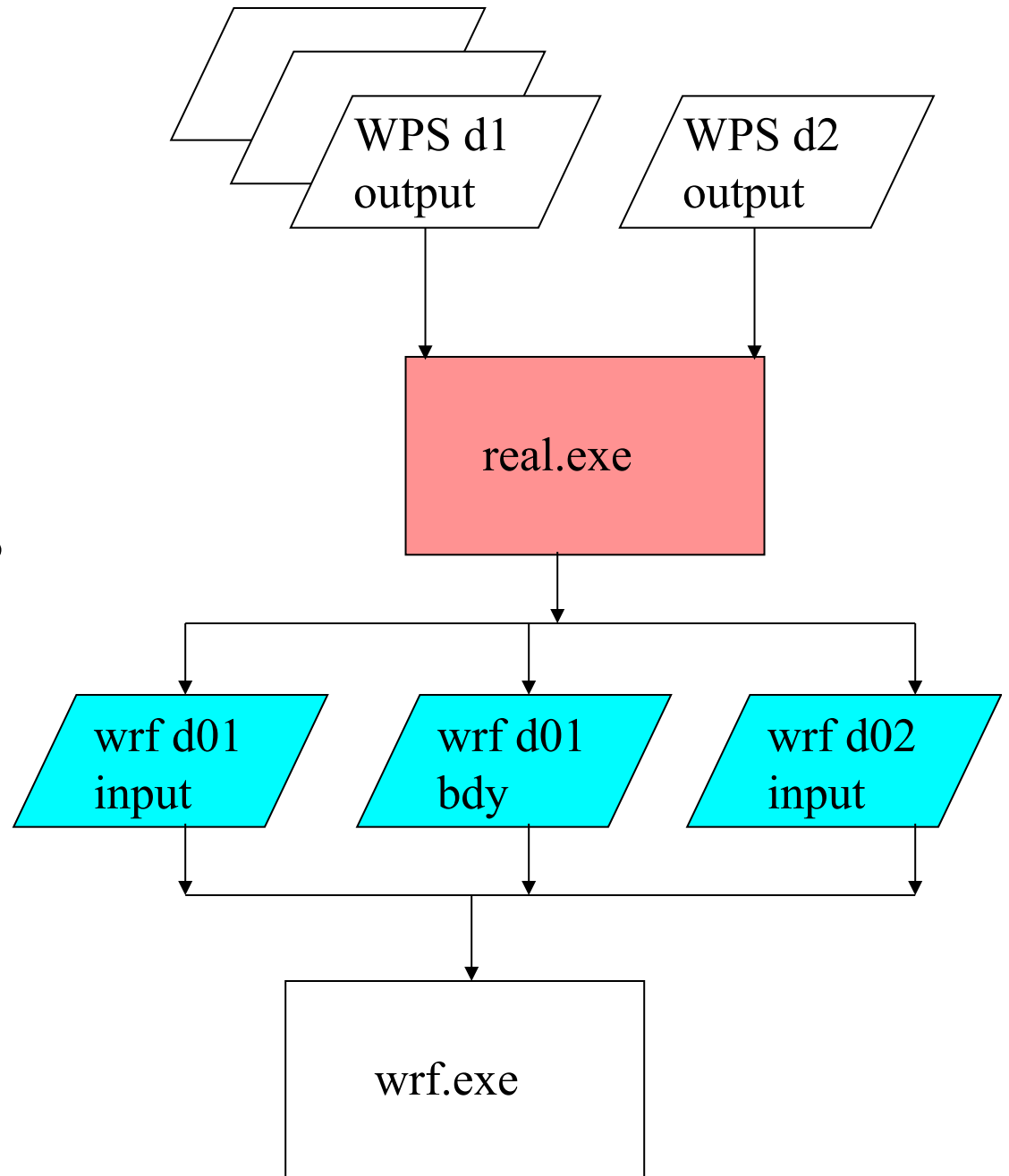


REAL

Description of General Functions

Dave Gill

gill@ucar.edu



Real program in a nutshell

- Function
- Standard input variables
- Base State
- Standard generated output
- Vertical interpolation
- Soil level interpolation

Function

- The WRF model pre-processor is *real.exe*
- The real.exe program is available *serial* or *DM parallel* (primarily for aggregate memory purposes, as opposed to timing performance)
- This program is automatically generated when the model is built and the requested use is for a real data case
- The real.exe program takes data *from WPS* and transform the data *for WRF*
- Similar to the ARW idealized data pre-processor, real.exe is tightly coupled to the WRF model through the *Registry*

Function

- *3D forecast* or simulation
- *Meteorological input* data that primarily originated from a previous forecast or analysis, probably via the WPS package
- Anticipated *utilization of physics* packages for microphysics, surface conditions, radiation, convection, and boundary layer (maybe usage of nudging capabilities)

Function

- A non-Cartesian *projected domain*
 - Lambert conformal, Mercator, polar stereographic, rotated latitude/longitude (global or regional)
- Selection of *realistic static fields* of topography, land use, vegetation, and soil category data
- Requirement of *time dependent* lateral boundary conditions for a regional forecast

Function

- Generation of *diagnostics* necessary for assumed WRF model input
- Input field *adjustment* for consistency of static and time dependent fields (land mask with soil temperature, etc.)
- ARW: computation of *reference* and *perturbation* fields
- Generation of *initial* state for each of the requested domains
- Creation of a *lateral boundary file* for the most coarse domain
- *Vertical interpolation* for 3d meteorological fields and for sub-surface soil data

Function

- **Run-time options**
 - specified in the Fortran namelist file (namelist.input for real and WRF)
- **Compile-time options**
 - Changes inside of the source code
 - Compiler flags
 - CPP ifdefs
 - Modifications to the Registry file

Standard Input Variables

- The metgrid program typically provides meteorological data to the real program.
- **Coordinate:**
 - The real program is able to input and correctly process any *strictly monotonic* vertical coordinate
 - Isobaric: OK
 - Sigma: OK
 - Hybrid: OK

Standard Input Variables

- The metgrid program typically provides meteorological data to the real program.
- **Mandatory:**
 - 3d and surface: horizontal winds, temperature, relative humidity, geopotential height
 - 3d soil: soil temperature
 - 2d fields: surface pressure, sea-level pressure, land mask
- **Optional** (but desirable):
 - 3d soil: soil moisture
 - 2d fields: topography elevation of input data, SST, sea-ice, skin temperature

Base State

- Several of the mass-point fields are *separated* into a time-independent *base state* (also called a reference state) and a *perturbation* from the base state
- The base state fields are only functions of the *topography* and a few user-selectable constants
- If the *topography changes*, such as with a moving nest, the base state fields are modified
- *Feedback* for 2-way nesting also impacts base state fields through topographic averaging – *inside of the WRF model*
- No base state computations are required *prior to the real program*

Hybrid Vertical Coordinate

- New in WRF v3.9 is the capability to have a hybrid vertical coordinate: a terrain following coordinate near the surface and relaxing to isobaric surfaces aloft

- A compile-time option is required:

configure -hyb

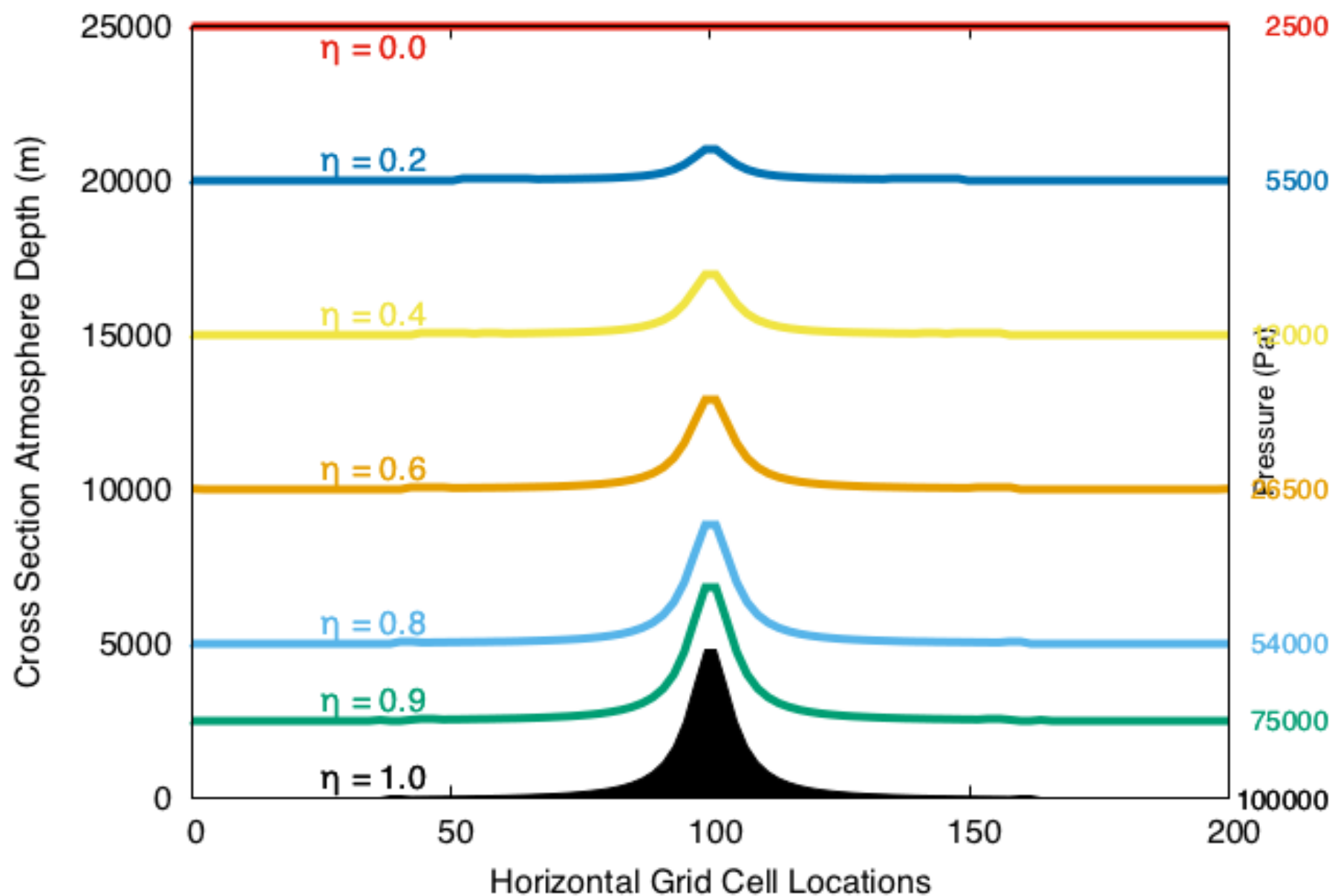
- A run-time option is required:

&dynamics

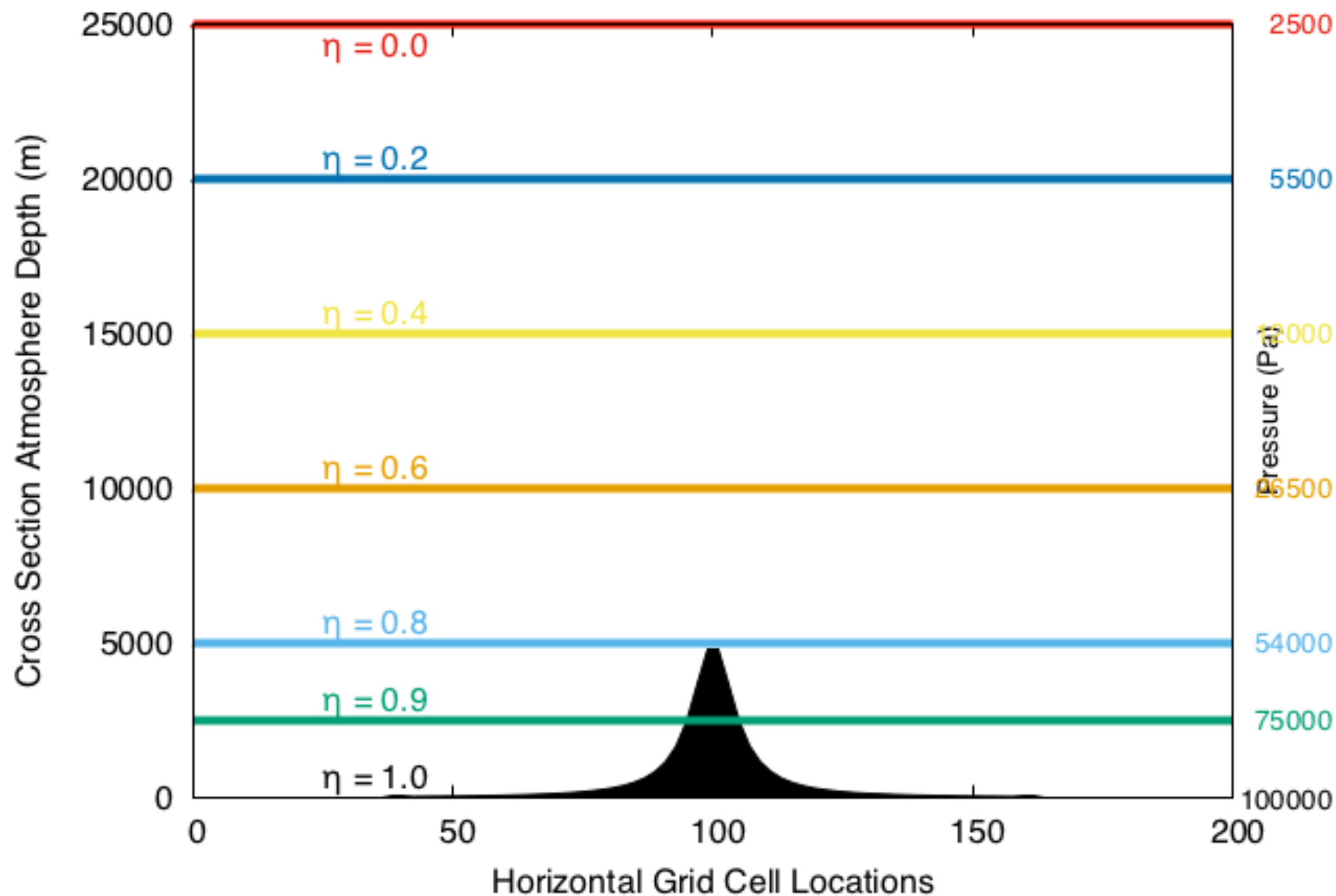
hybrid_opt = 2

/

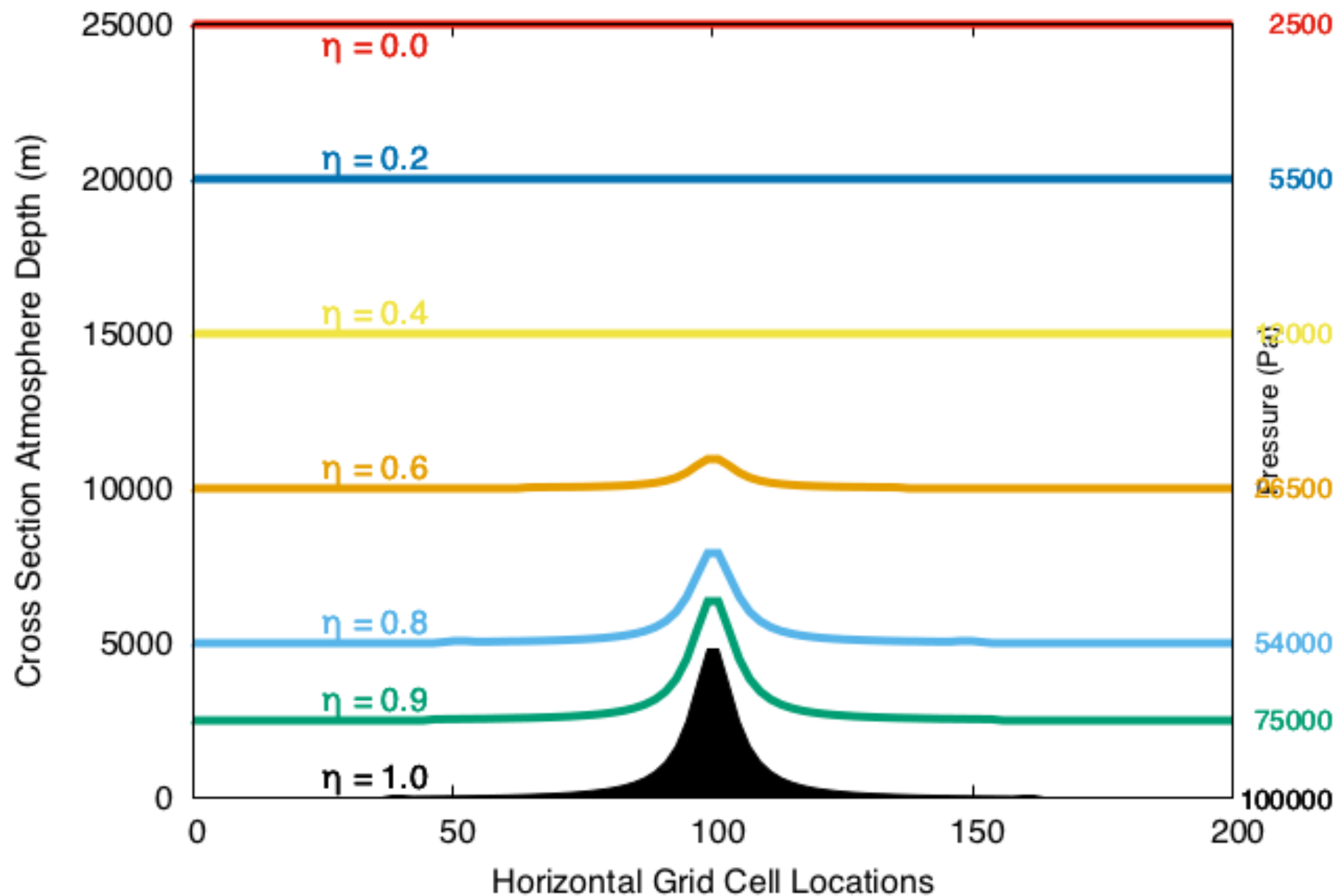
TERRAIN FOLLOWING Vertical Coordinate System



ISOBARIC Vertical Coordinate System



HYBRID Vertical Coordinate System



Standard Generated Output

- For regional forecasts, the real program generates both an initial (*wrfinput_d01*) and a lateral boundary (*wrfbdy_d01*)
- The boundary file is not required for *global forecasts* with ARW (look at MPAS for global simulations)
- The *initial condition* file contains a *single time period* of data
- These files contain data used directly by the WRF model
- The initial condition file may be ingested by the *WRFDA* code (referred to as a *cold-start*)
- If *n* times were processed with WPS and real, the lateral boundary file contains *n-1* time slices

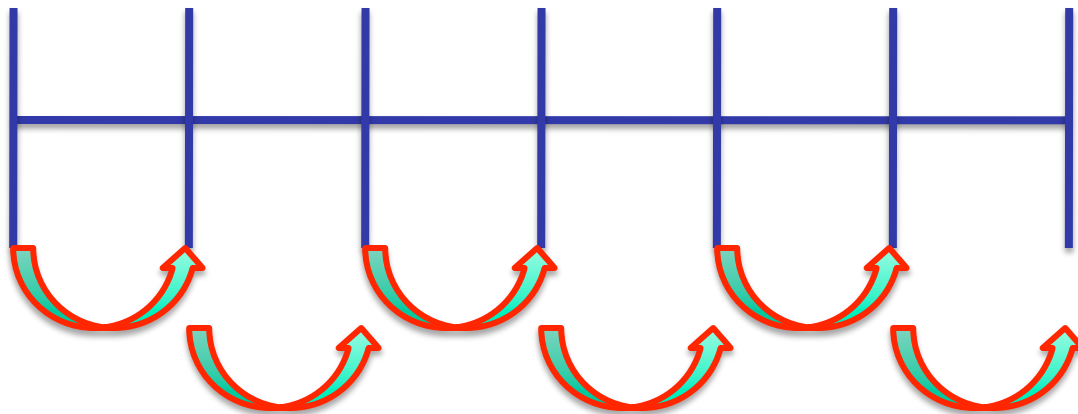
Lateral Boundary Condition Times

0 6 12 18 24 30 36 h

Time periods
from WPS

1 2 3 4 5 6 7

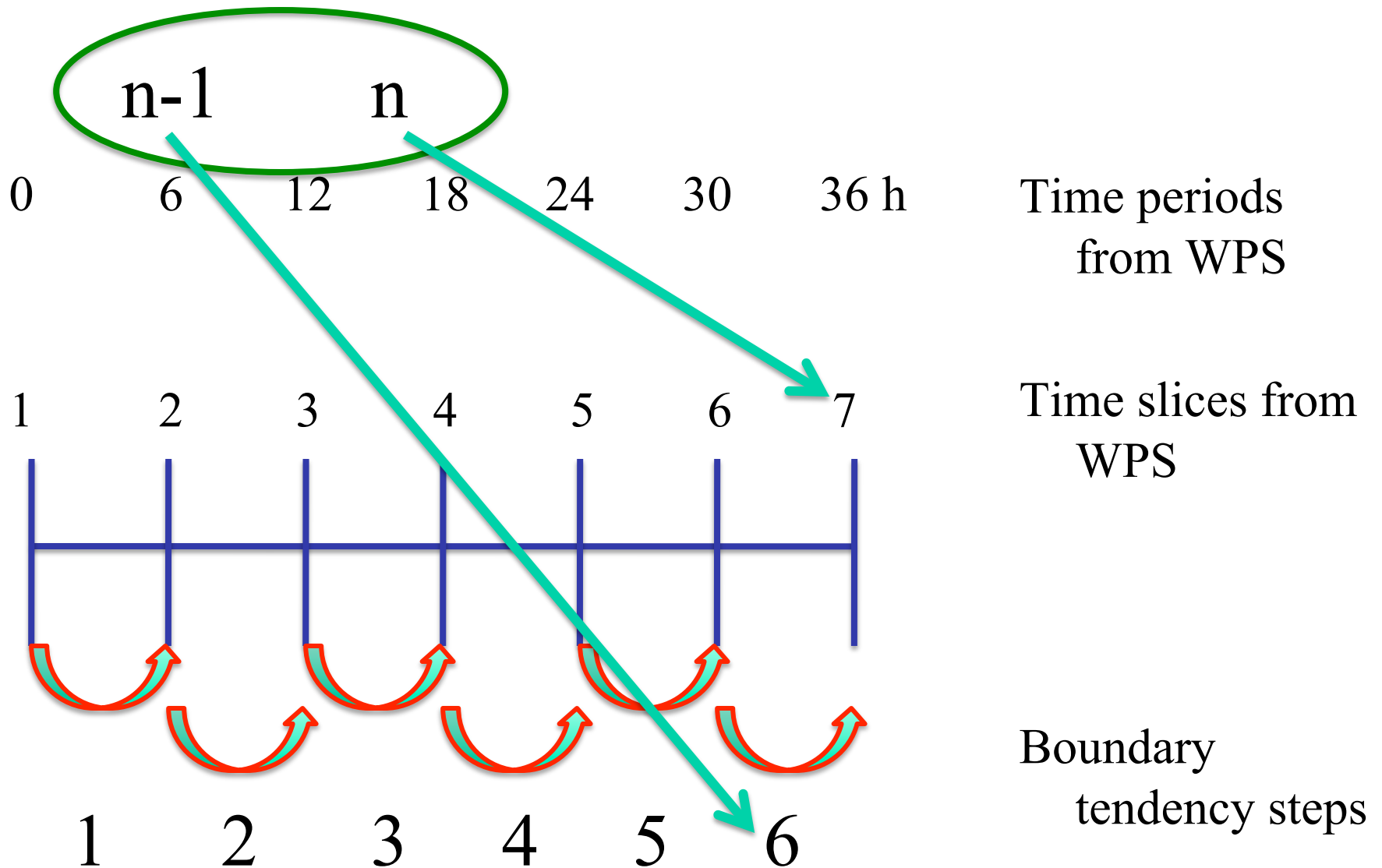
Time slices from
WPS



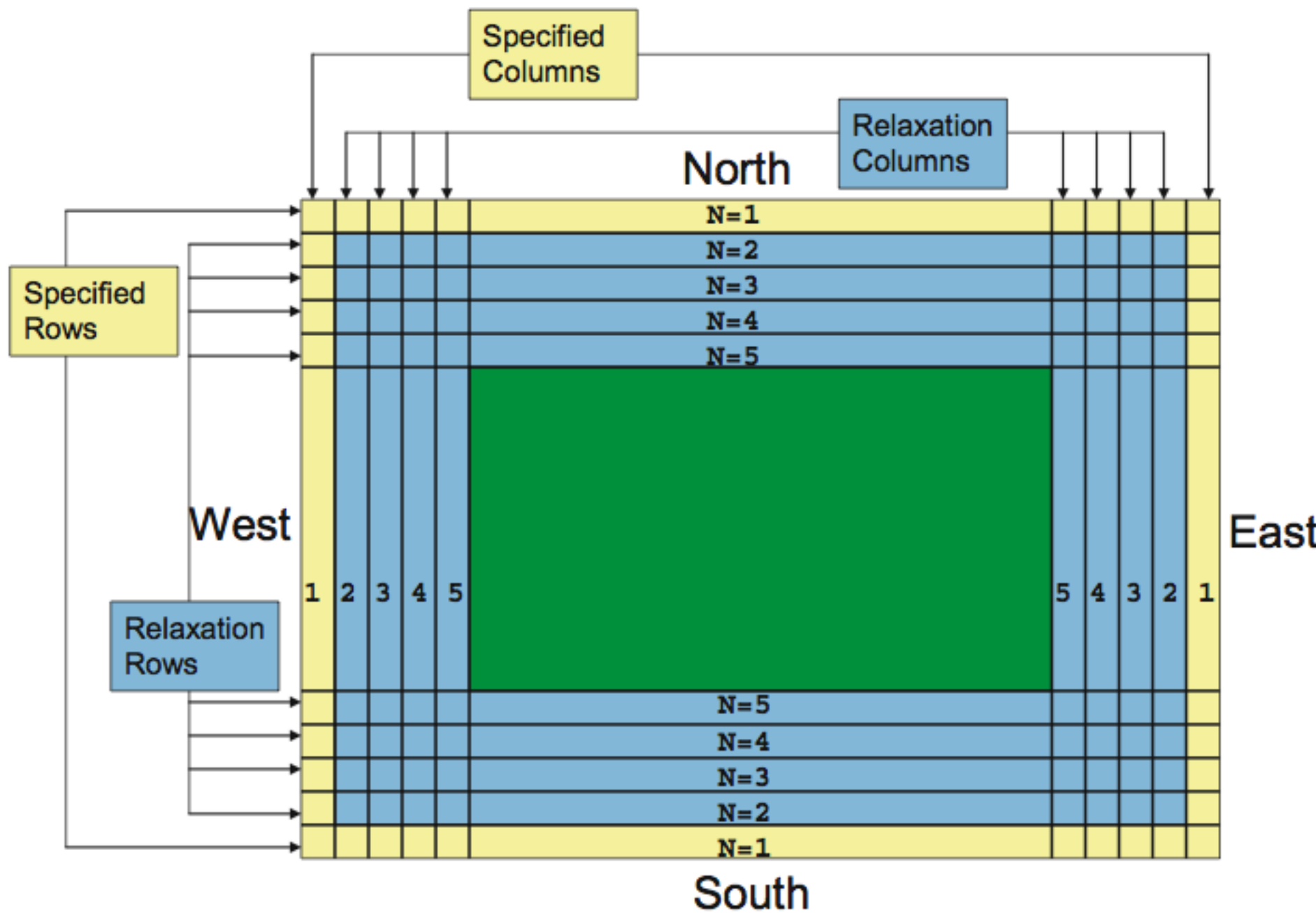
1 2 3 4 5 6

Boundary
tendency steps

Lateral Boundary Condition Times



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

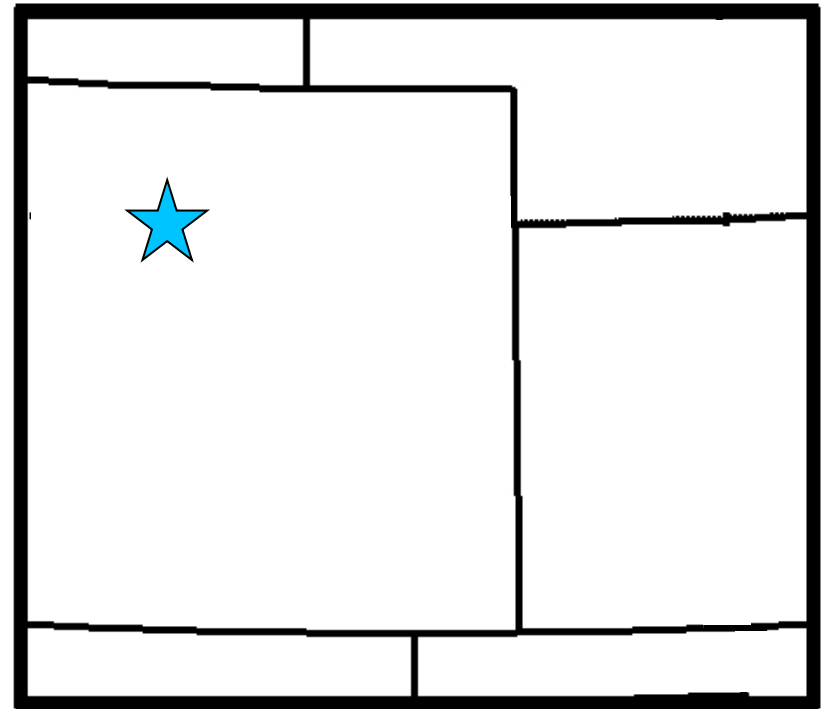
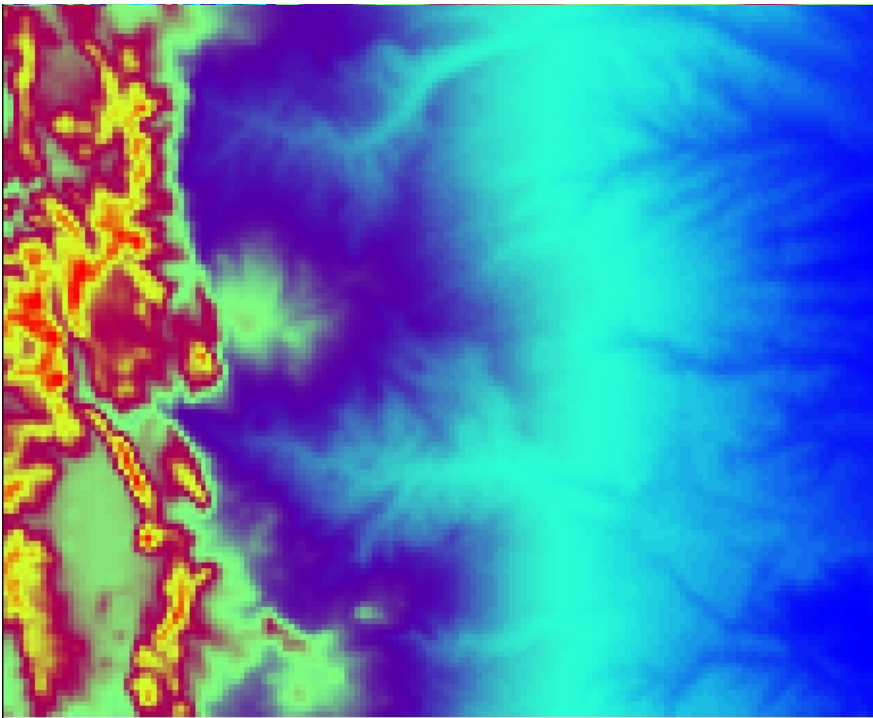


Vertical Interpolation

- A number of vertical interpolation options are available to users
- The options can have a significant impact on the initial conditions passed to the model
- More information is contained in the info file *README.namelist* in the *run* directory
- Options are located in the *&domains* namelist record of *namelist.input*

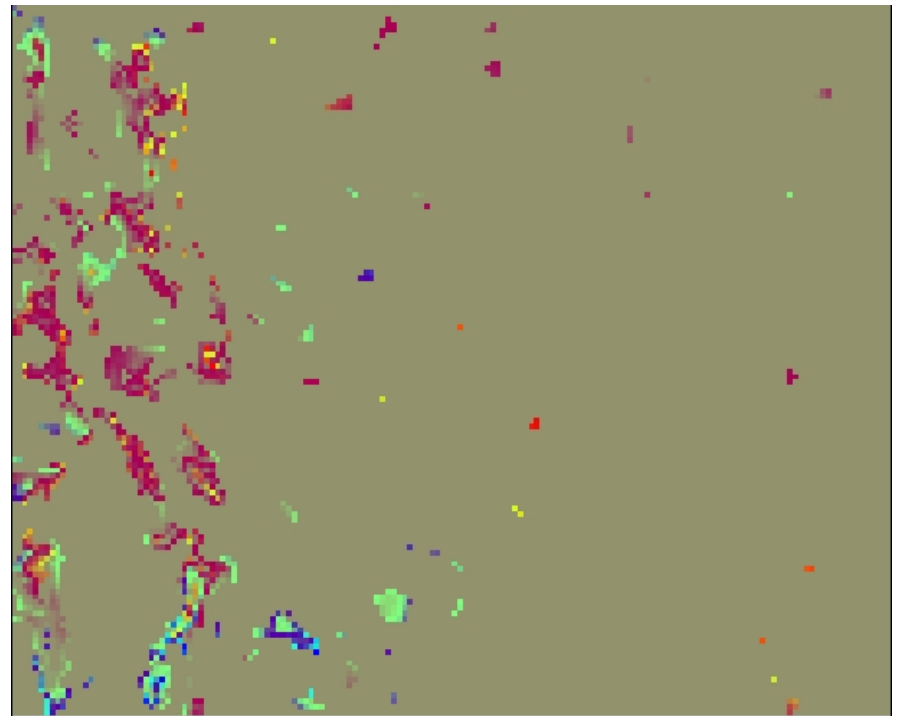
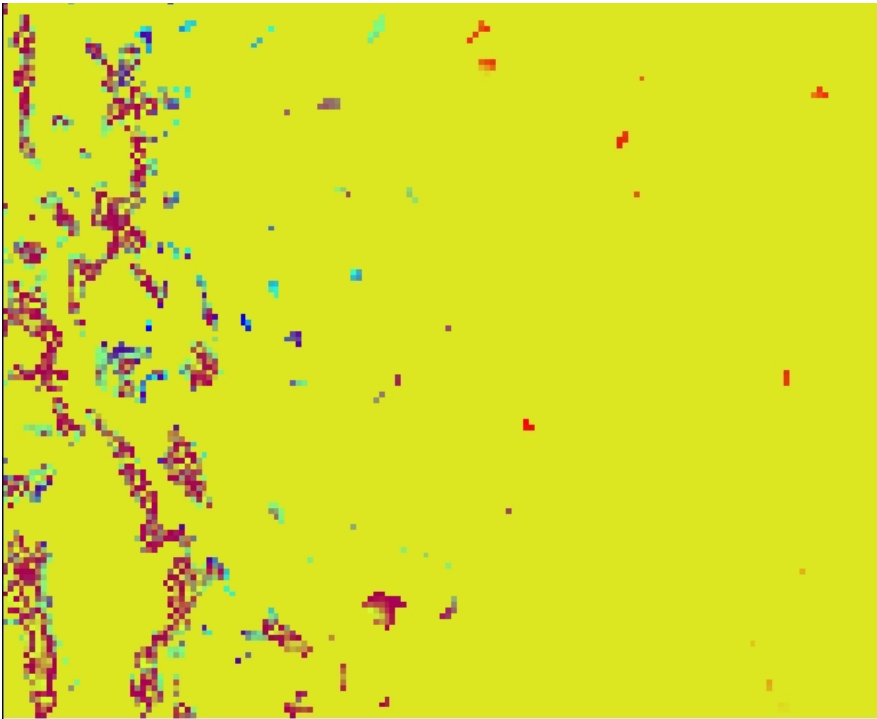
Vertical Interpolation

- Impact: *Expected region of changes*
 - *Non-standard setting*
 - Which level is being viewed
-
- Topography and domain for difference plots, 160x140, 4 km, input = 40 km NAM



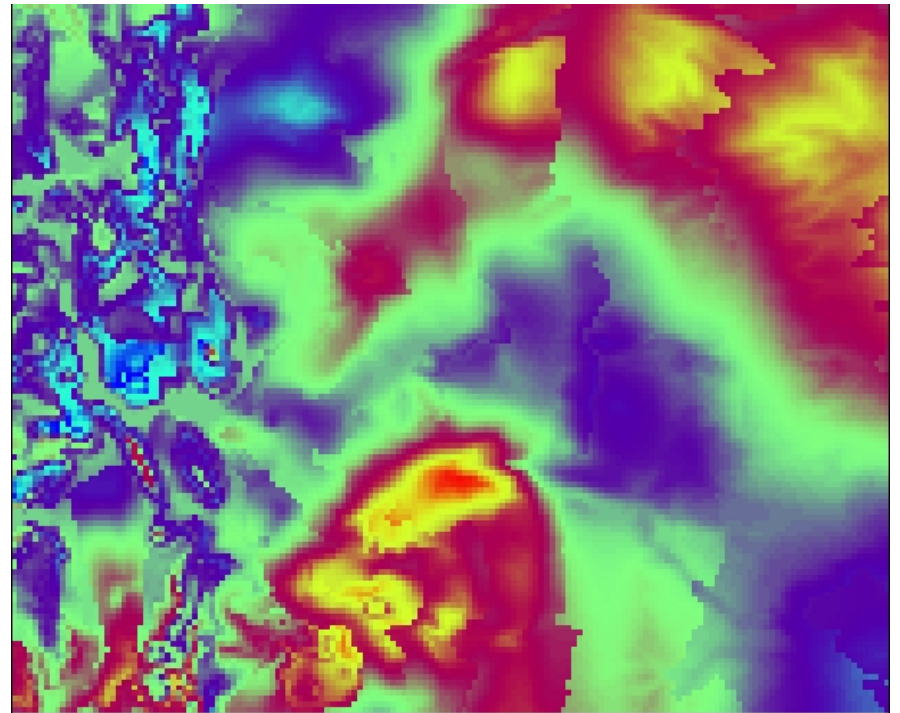
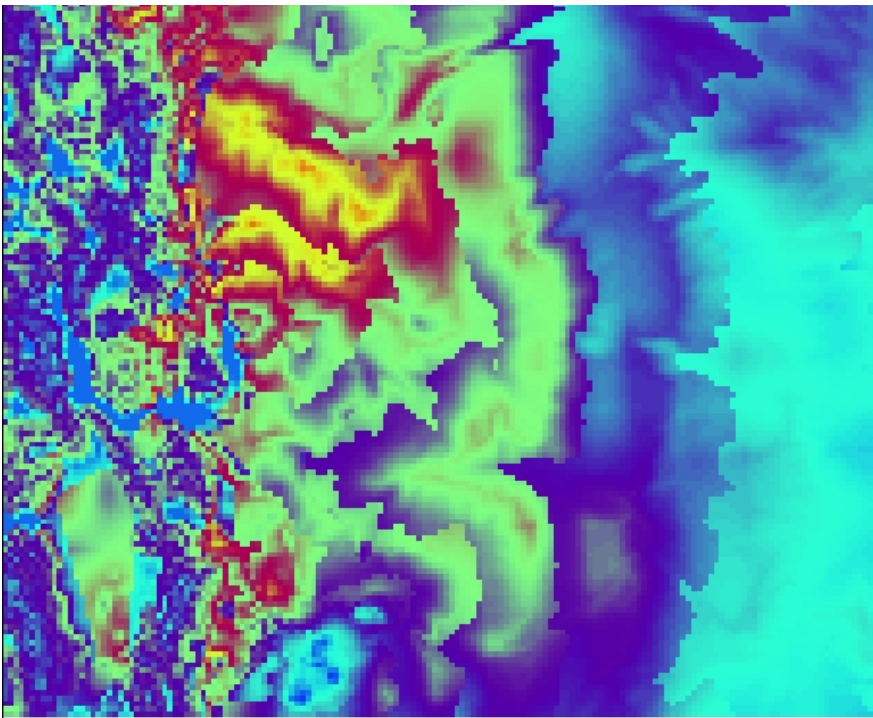
Vertical Interpolation

- Impact: few lowest levels only
 - $\text{force_sfc_in_vinterp} = 0$
 - h level 1
-
- Theta (−8 K blue, 0 K yellow)
 - U (−3 m/s blue, 2 m/s red)



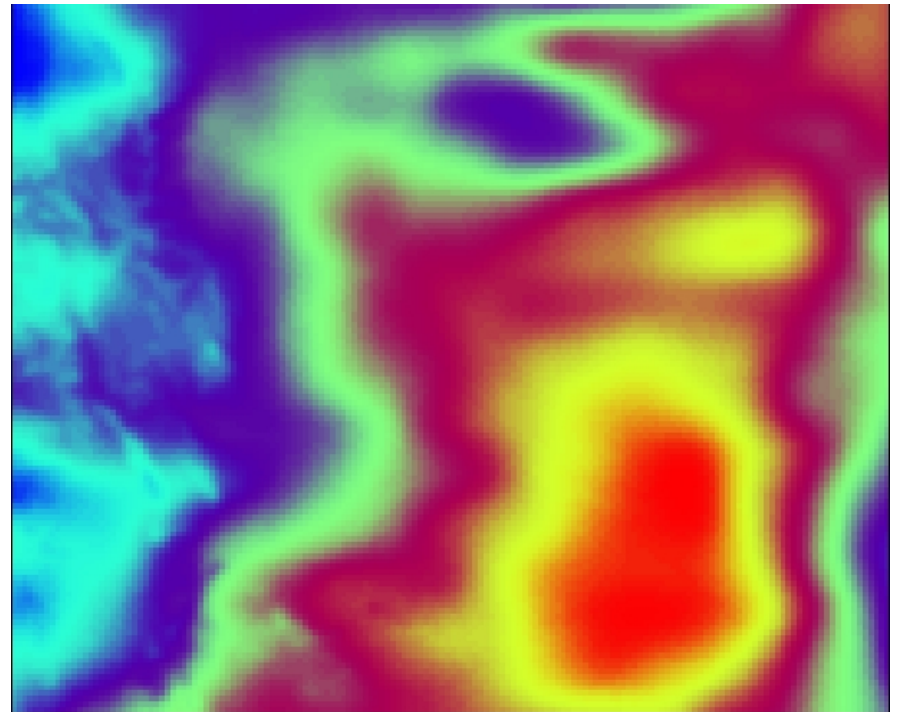
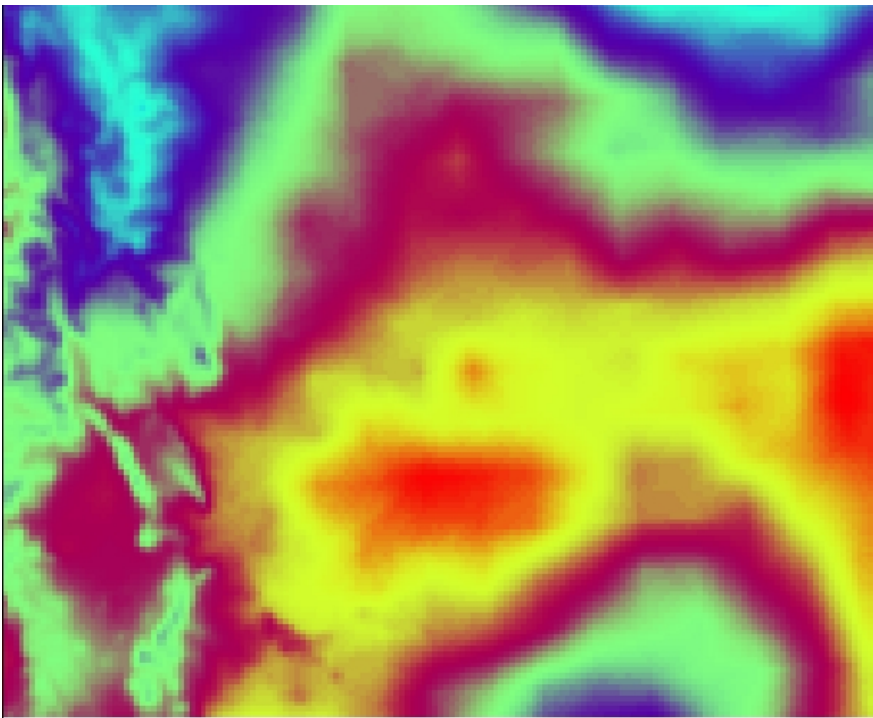
Vertical Interpolation

- Impact: few lowest levels only
- `force_sfc_in_vinterp = 6`
- `h level 4`
- Theta (0 K blue, 10 K red)
- U (-5 m/s blue, 6 m/s red)



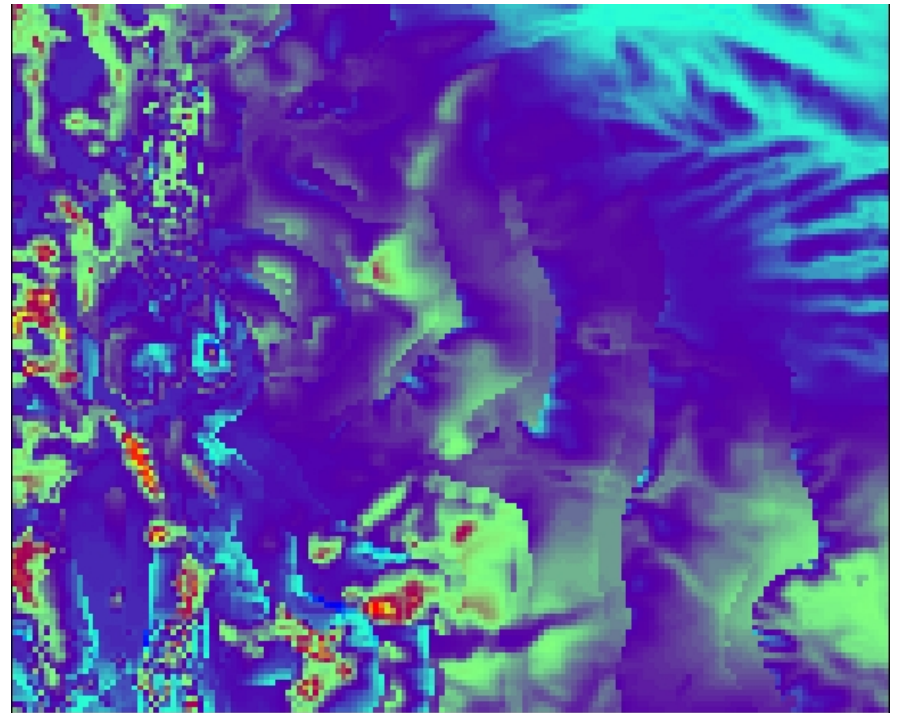
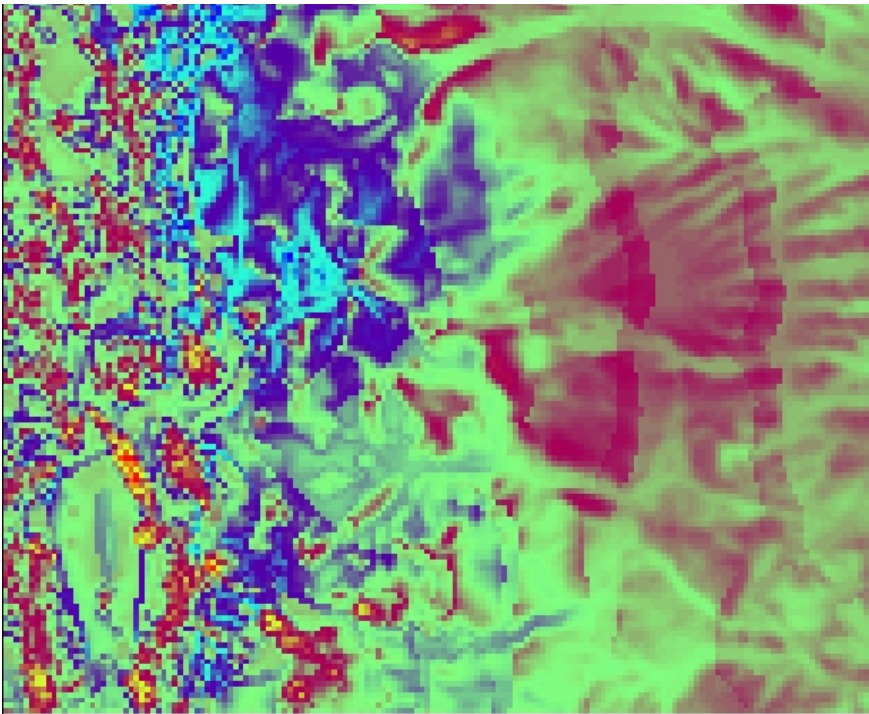
Vertical Interpolation

- Impact: above first 4 levels, most near tropopause
 - lagrange_order = 2
 - h level TOP
-
- Theta (0.7 K blue, 1.6 K red)
 - U (0.4 m/s blue, 1.4 m/s red)



Vertical Interpolation

- Impact: lowest level only
 - `lowest_lev_from_sfc = T`
 - `h level 1`
-
- Theta (–10 K blue, 8 K red)
 - U (–3 m/s blue, 7 m/s red)



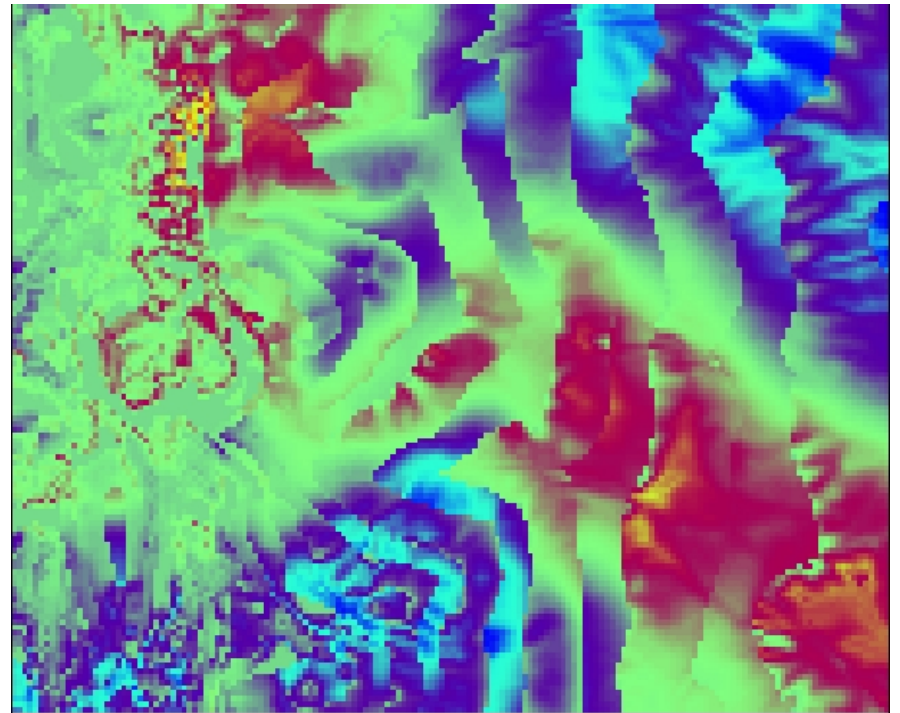
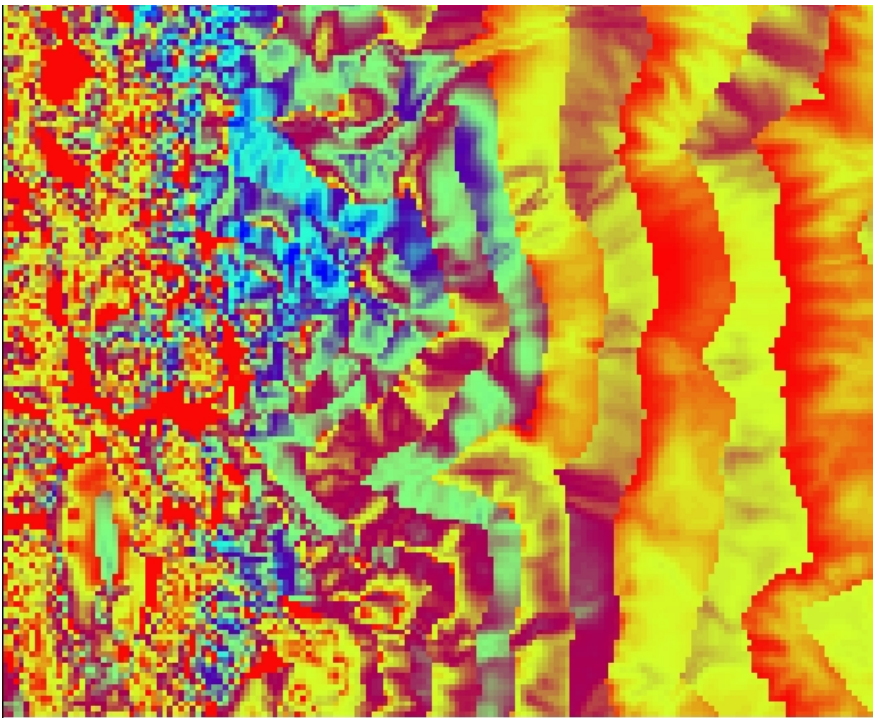
Vertical Interpolation

- Impact: outer few rows and column, amplitude damps upward
 - `smooth_cg_topo = T`
 - h level 1
-
- Theta (−10 K blue, 9 K red)
 - U (−6 m/s blue, 6 m/s red)



Vertical Interpolation

- Impact: lowest few levels
 - `use_surface = F`
 - h level 1
-
- Theta (−11 K blue, 0 K red)
 - U (−3 m/s blue, 4 m/s red)



Vertical Interpolation

Make sure input data is vertically *ordered* as expected

Input 3-D pressure and T, topo, Z, moisture used to
compute total *surface pressure*

Compute target *vertical coordinate* using normalized dry
column pressure pressure

The *eta surfaces* may be computed or selected

Vertically interpolate input fields in pressure to the eta
surfaces in dry pressure: default all variables linear in
log(pressure)

Vertical Interpolation

- Select reasonable eta levels, or let the real program do it for you
- Verify that the *“thicknesses” are acceptable*, generally about the same value in the free-atmosphere and less than 1000 m
- It is *SAFEST to NOT initially choose eta values*
 - Initially, *select the number* of eta levels
 - *Plot profiles* of the resultant heights
 - *Adjust the hlevels* accordingly
- A few namelist options, the terrain elevation, and eta levels completely define the model coordinate for the WRF code

Vertical Interpolation

- The *eta surfaces* are computed with a few NML parameters:

```
&domains
```

```
e_vert          = 50,      50,      50
```

```
p_top_requested = 1000,
```

```
&dynamics
```

```
base_temp       = 290.
```

```
iso_temp        = 200
```

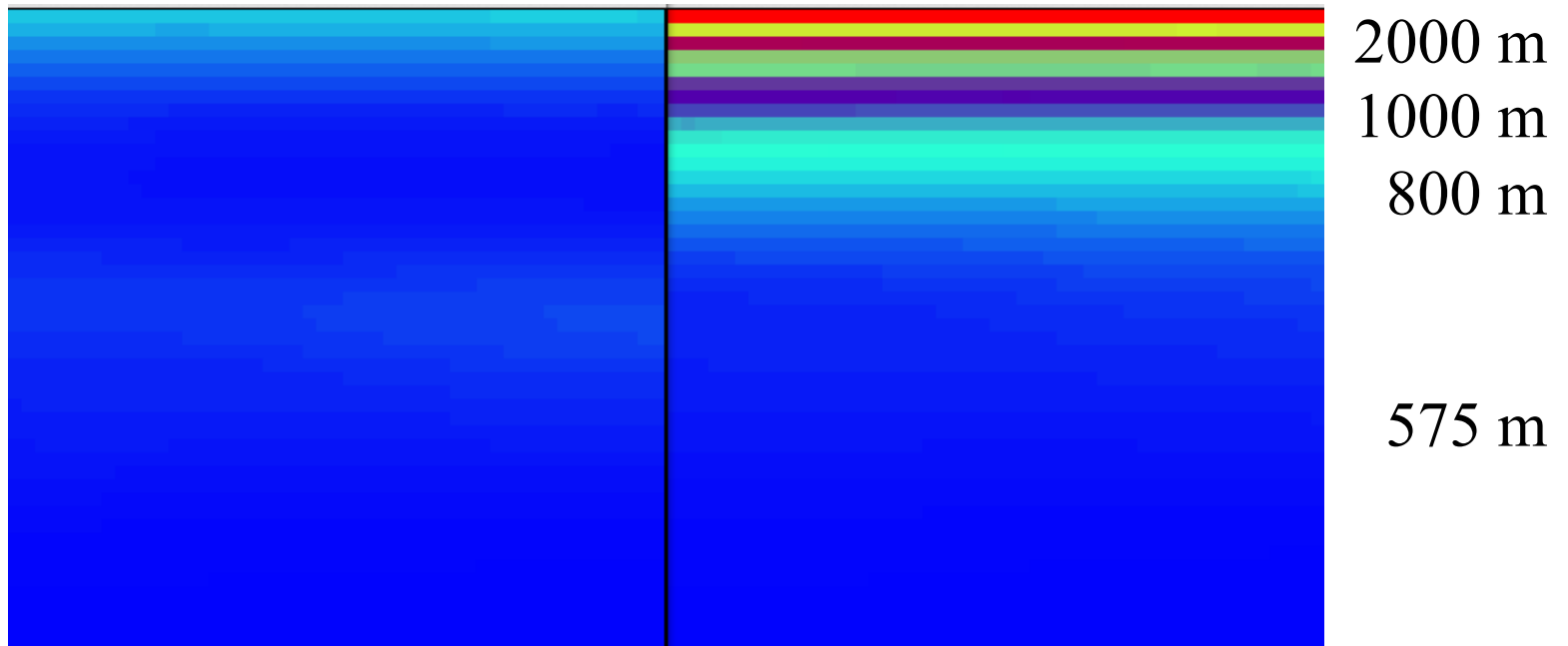
Vertical Interpolation

Vertical cross sections of THICKNESS of each model layer,
with 50 vertical levels above the PBL, $p_{\text{top}} = 10$ hPa.

Uniform layers

Exaggerated Stretching

720-820 m













Physical Parameterization Settings

- The real program and the WRF model are tightly coupled
- Most physical parameterization settings in the namlist.input are IGNORED by real
- EXCEPT
 - *sf_surface_physics*
 - Land surface model (processes soil temperature and soil moisture)
 - Different schemes in WRF use *differing numbers of layers*
 - The layers are defined in real from the metgrid output











Soil Level Interpolation

- The WRF model supports several Land Surface schemes:
 - `sf_surface_physics = 1`, Slab scheme
 - 5 layers
 - Defined with thicknesses: 1, 2, 4, 8, 16 cm

Noah		RUC	
Layers	Mid point		Levels
	000 – 010 cm -- 005 cm		000 cm
	010 – 040 cm -- 025 cm		005 cm
	040 – 100 cm -- 070 cm		020 cm
	100 – 200 cm -- 150 cm		040 cm
			160 cm
			300 cm











Soil Level Interpolation

- The WRF model supports several Land Surface schemes:
 - `sf_surface_physics = 2`, Unified Noah scheme
 - 4 layers
 - Defined with layers: 0-10, 10-40, 40-100, 100-200 cm

Noah		RUC	
Layers	Mid point		Levels
	000 – 010 cm -- 005 cm		000 cm
	010 – 040 cm -- 025 cm		005 cm
	040 – 100 cm -- 070 cm		020 cm
	100 – 200 cm -- 150 cm		040 cm
			160 cm
			300 cm











Soil Level Interpolation

- The WRF model supports several Land Surface schemes:
 - `sf_surface_physics = 3`, RUC scheme
 - 6 levels
 - Defined at levels: 0, 5, 20, 40, 160, 300 cm

Noah		RUC	
Layers	Mid point		Levels
	000 – 010 cm -- 005 cm		000 cm
	010 – 040 cm -- 025 cm		005 cm
	040 – 100 cm -- 070 cm		020 cm
	100 – 200 cm -- 150 cm		040 cm
			160 cm
			300 cm

Soil Level Interpolation

- The WRF model supports several Land Surface schemes:
 - `sf_surface_physics = 7`, PX scheme
 - 2 layers
 - Defined with layers: 0-1, 1-100 cm

Noah		RUC	
Layers	Mid point		Levels
	000 – 010 cm -- 005 cm		000 cm
	010 – 040 cm -- 025 cm		005 cm
	040 – 100 cm -- 070 cm		020 cm
	100 – 200 cm -- 150 cm		040 cm
			160 cm
			300 cm

Real program in a nutshell

- Function
- Standard input variables
- Base State
- Standard generated output
- Vertical interpolation
- Soil level interpolation

Real namelist

- The real program and WRF model SHARE a namelist file:
namelist.input
- Some entries are used by both programs: dimensions, starting date/time, LBC, FDDA
- Many entries are only for one of the programs: real does not care about advection, diffusion, DFI

WRFV3

/test

/em_real

/namelist.input

Real namelist

- The **namelist.input** file is separated into separate namelist records, usually with related areas

&time_control – dates, time, I/O

&domains – domain grid sizes, REAL options

&physics – land scheme and layers, land cats

&dynamics – base state, GWD

&bdy_control – lateral boundary conditions

Real namelist

`&time_control`

<code>start_year</code>	<code>= 2017, 2008,</code>
<code>start_month</code>	<code>= 05, 01,</code>
<code>start_day</code>	<code>= 30, 01,</code>
<code>start_hour</code>	<code>= 12, 00,</code>
<code>end_year</code>	<code>= 2017, 2008,</code>
<code>end_month</code>	<code>= 05, 01,</code>
<code>end_day</code>	<code>= 30, 01,</code>
<code>end_hour</code>	<code>= 12, 00,</code>
<code>interval_seconds</code>	<code>= 21600</code>
<code>input_from_file</code>	<code>= .t., .t.,</code>
<code>io_form_input</code>	<code>= 2</code>
<code>io_form_boundary</code>	<code>= 2</code>

Real namelist

`&time_control`

`start_year = 2017, 2008,`

`start_month = 05, 01,`

`start_day = 30, 01,`

`start_hour = 12, 00,`

`end_year = 2017, 2008,`

`end_month = 05, 01,`

`end_day = 30, 01,`

`end_hour = 12, 00,`

`interval_seconds = 21600`

`input_from_file = .t., .t.,`

`io_form_input = 2`

`io_form_boundary = 2`

START:

Same as metgrid

Real namelist

&time_control

start_year = 2017, 2008,

start_month = 05, 01,

start_day = 30, 01,

start_hour = 12, 00,

end_year = 2017, 2008,

end_month = 05, 01,

end_day = 30, 01,

end_hour = 12, 00,

interval_seconds = 21600

input_from_file = .t., .t.,

io_form_input = 2

io_form_boundary = 2

END:

Same as metgrid

Real namelist

&time_control

start_year = 2017, 2008,

start_month = 05, 01,

start_day = 30, 01,

start_hour = 12, 00,

end_year = 2017, 2008,

end_month = 05, 01,

end_day = 30, 01,

end_hour = 12, 00,

interval_seconds = 21600

input_from_file = .t., .t.,

io_form_input = 2

io_form_boundary = 2

Interval:

Same as metgrid

Real namelist

&time_control

start_year = 2017, 2008,

start_month = 05, 01,

start_day = 30, 01,

start_hour = 12, 00,

end_year = 2017, 2008,

end_month = 05, 01,

end_day = 30, 01,

end_hour = 12, 00,

interval_seconds = 21600

input_from_file = .t., .t.,

io_form_input = 2

io_form_boundary = 2

How many domains
from geogrid and
metgrid

Real namelist

`&time_control`

`start_year = 2017, 2008,`

`start_month = 05, 01,`

`start_day = 30, 01,`

`start_hour = 12, 00,`

`end_year = 2017, 2008,`

`end_month = 05, 01,`

`end_day = 30, 01,`

`end_hour = 12, 00,`

`interval_seconds = 21600`

`input_from_file = .t., .t.,`

`io_form_input = 2`

`io_form_boundary = 2`

Leave default = 2
NETCDF

Real namelist

&domains

max_dom	=	1,	
e_we	=	74,	112,
e_sn	=	61,	97,
e_vert	=	30,	30,
p_top_requested	=	5000,	
num_metgrid_levels	=	27,	
num_metgrid_soil_levels	=	4,	
dx	=	30000,	10000,
dy	=	30000,	10000,
grid_id	=	1,	2,
parent_id	=	0,	1,
i_parent_start	=	1,	31,
j_parent_start	=	1,	17,
parent_grid_ratio	=	1,	3,

Real namelist

```
&domains
max_dom           = 1,
e_we              = 74,    112,
e_sn              = 61,    97,
e_vert           = 30,    30,
p_top_requested   = 5000,
num_metgrid_levels = 27,
num_metgrid_soil_levels = 4,
dx               = 30000, 10000,
dy               = 30000, 10000,
grid_id          = 1,    2,
parent_id        = 0,    1,
i_parent_start   = 1,    31,
j_parent_start   = 1,    17,
parent_grid_ratio = 1,    3,
```

REAL: Total
number of domains
on INPUT

Real namelist

&domains

```
max_dom          = 1,  
e_we             = 74,      112,  
e_sn             = 61,      97,  
e_vert           = 30,      30,  
p_top_requested  = 5000,  
num_metgrid_levels = 27,  
num_metgrid_soil_levels = 4,  
dx               = 30000, 10000,  
dy               = 30000, 10000,  
grid_id          = 1,      2,  
parent_id        = 0,      1,  
i_parent_start   = 1,      31,  
j_parent_start   = 1,      17,  
parent_grid_ratio = 1,      3,
```

Domain size in grid
cells (u,v,w)

Real namelist

&domains

```
max_dom          = 1,  
e_we             = 74,      112,  
e_sn             = 61,      97,  
e_vert           = 30,      30,  
p_top_requested  = 5000,  
num_metgrid_levels = 27,  
num_metgrid_soil_levels = 4,  
dx               = 30000, 10000,  
dy               = 30000, 10000,  
grid_id          = 1,      2,  
parent_id        = 0,      1,  
i_parent_start   = 1,      31,  
j_parent_start   = 1,      17,  
parent_grid_ratio = 1,      3,
```

Model lid (Pa)

5000 Pa (20 km),

No lower

Real namelist

&domains

```
max_dom          = 1,  
e_we             = 74,      112,  
e_sn             = 61,      97,  
e_vert           = 30,      30,  
p_top_requested  = 5000,  
num_metgrid_levels = 27,  
num_metgrid_soil_levels = 4,  
dx               = 30000, 10000,  
dy               = 30000, 10000,  
grid_id          = 1,      2,  
parent_id        = 0,      1,  
i_parent_start   = 1,      31,  
j_parent_start   = 1,      17,  
parent_grid_ratio = 1,      3,
```

Consistent with
dimensions from
metgrid

Real namelist

&domains

```
max_dom          = 1,  
e_we             = 74,      112,  
e_sn             = 61,      97,  
e_vert           = 30,      30,  
p_top_requested  = 5000,  
num_metgrid_levels = 27,  
num_metgrid_soil_levels = 4,  
dx               = 30000, 10000,  
dy               = 30000, 10000,  
grid_id          = 1,      2,  
parent_id        = 0,      1,  
i_parent_start   = 1,      31,  
j_parent_start   = 1,      17,  
parent_grid_ratio = 1,      3,
```

Grid distance (m)
dx=dy, except for
lat/lon domains

Real namelist

&domains

```
max_dom          = 1,  
e_we             = 74,      112,  
e_sn             = 61,      97,  
e_vert           = 30,      30,  
p_top_requested  = 5000,  
num_metgrid_levels = 27,  
num_metgrid_soil_levels = 4,  
dx               = 30000, 10000,  
dy               = 30000, 10000,  
grid_id          = 1,      2,  
parent_id        = 0,      1,  
i_parent_start   = 1,      31,  
j_parent_start   = 1,      17,  
parent_grid_ratio = 1,      3,
```

Parent/child
information, same as
metgrid

Real namelist

&domains

```
max_dom           = 1,  
e_we              = 74,      112,  
e_sn              = 61,      97,  
e_vert           = 30,      30,  
p_top_requested   = 5000,  
num_metgrid_levels = 27,  
num_metgrid_soil_levels = 4,  
dx               = 30000, 10000,  
dy               = 30000, 10000,  
grid_id          = 1,      2,  
parent_id        = 0,      1,  
i_parent_start   = 1,      31,  
j_parent_start   = 1,      17,  
parent_grid_ratio = 1,      3,  
smooth_cg_topo   = .t.
```

With high topo on
CG boundaries, turn
this ON

Real namelist

&physics

```
sf_surface_physics = 2,      2,  
num_soil_layers    = 4,  
num_land_cat       = 21,  
sf_urban_physics   = 0,      0,
```

Real namelist

```
&physics  
sf_surface_physics = 2,      2,  
num_soil_layers    = 4,  
num_land_cat       = 21,  
sf_urban_physics   = 0,      0,
```

Real and WRF have to be consistent with the surface layer scheme due to the dimensions of the soil temp and moisture

Real namelist

```
&physics  
sf_surface_physics = 2,      2,  
num_soil_layers    = 4,  
num_land_cat       = 21,  
sf_urban_physics   = 0,      0,
```

The dimensions of
the land categories
must match the
selected data source
from geogrid

Real namelist

&physics

sf_surface_physics = 2, 2,

num_soil_layers = 4,

num_land_cat = 21,

sf_urban_physics = 0, 0,

Arrays between real
and WRF need to be
consistently
dimensioned

Real namelist

&dynamics

base_temp = 290.

gwd_opt = 1,

Real namelist

```
&dynamics  
base_temp      = 290.  
gwd_opt        = 1,
```

Atmospheric
temperature (K) at
sea level (NOT SST)
in the middle of
your domain

Must not change
between real and
WRF

Real namelist

&dynamics

base_temp = 290.

gwd_opt = 1,

GWD allocates
space which must be
consistent between
real and WRF

Real namelist

&bdy_control

```
spec_bdy_width = 5,  
spec_zone      = 1,  
relax_zone     = 4,  
specified      = .t., .f.,  
nested         = .f., .t.,
```

Real namelist

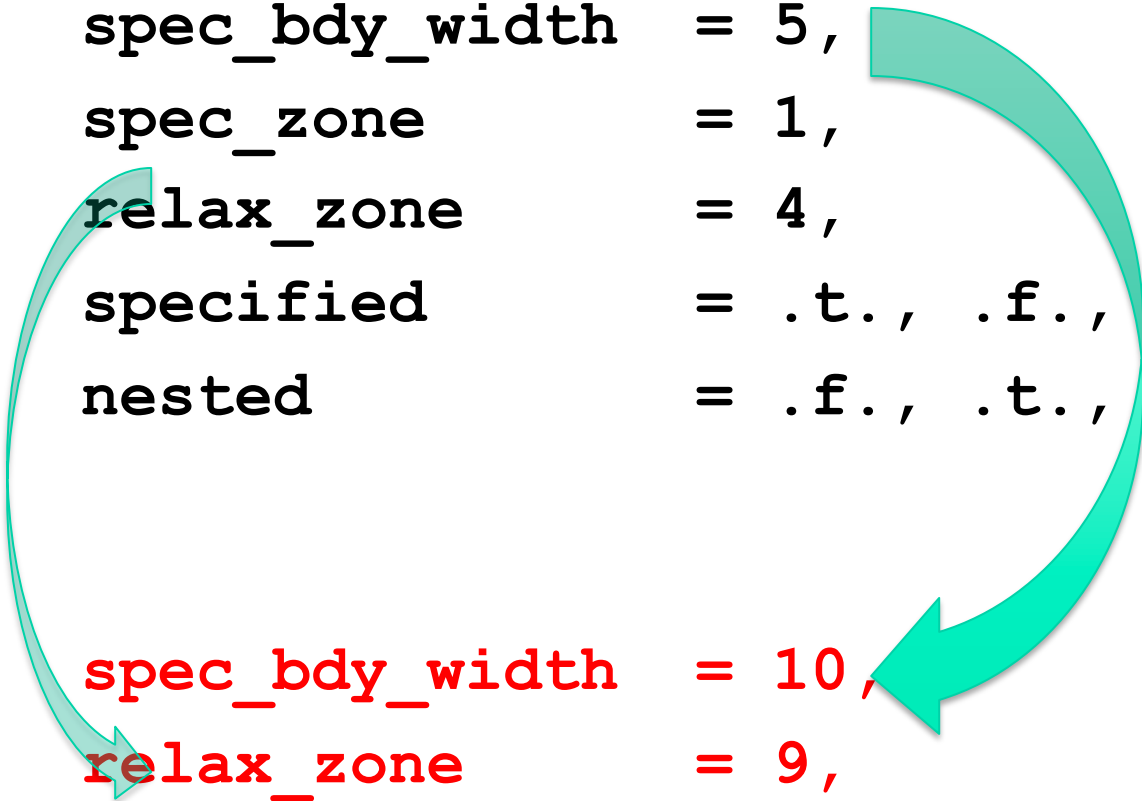
```
&bdy_control  
spec_bdy_width = 5,  
spec_zone      = 1,  
relax_zone     = 4,  
specified      = .t., .f.,  
nested         = .f., .t.,
```

```
spec_bdy_width =  
spec_zone +  
relax_zone  
  
spec_zone = 1
```

Real namelist

&bdy_control

spec_bdy_width = 5,
spec_zone = 1,
relax_zone = 4,
specified = .t., .f.,
nested = .f., .t.,



spec_bdy_width = 10,
relax_zone = 9,
spec_exp = 0.33

Can choose larger
relaxation zone
since domain is big

Can choose
exponential decay

Real namelist

```
&bdy_control  
spec_bdy_width = 5,  
spec_zone      = 1,  
relax_zone     = 4,  
specified      = .t., .f.,  
nested         = .f., .t.,
```

REAL:

d01 always
specified=T

All other domains
have specified=F

Real namelist

&bdy_control

spec_bdy_width = 5,

spec_zone = 1,

relax_zone = 4,

specified = .t., .f.,

nested = .f., .t.,

d01 always
nested=F

All other domains
have nested=T

Real program in a nutshell: PART 2

- Access to everything
- Eta levels
- Metgrid flags
- Adding a variable for vertical interpolation
- Vertical interpolation
- Tracers
- Trajectories
- Options

Access to Everything

- The primary location to modify the real program is the **dyn_em/module_initialize_real.F** file
- Contains:
 - Registry information
 - All of the namelist settings selected
 - Variables **from** the metgrid program
 - Variables to be **sent to** the WRF model
- Called for **every time period**, for **every domain**

Access to Everything

- The value of **every variable input** into the WRF model is controlled through module_initialize_real.F
- All variables are accessed through the **derived data type** “grid”

```
DO j=jts,MIN(jde-1,jte)
  DO i=its,MIN(ide-1,ite)
    grid%sst(i,j) = grid%sst(i,j) + 1
  END DO
END DO
```

Access to Everything

- The dynamics variables have **two time levels**, indicated by the `_1` and `_2` suffixes. Only the `_2` variables are sent to WRF.
- Some variables sent to WRF are **diagnostic** only

```
DO j = jts, min(jde-1,jte)
  DO i = its, min(ide,ite)
    grid%u10(i,j)=grid%u_gc(i,1,j)
  END DO
END DO
```

Eta Levels

- The **vertical coordinate**, eta, used in the WRF model is defined inside of the real program.
- The user may allow the real program to choose the levels (select only the number of levels in the namelist.input file)

```
&domains
```

```
e_vert      = 30,      30,      30,  
/
```

```
&domains
```

```
e_vert      = 30,      40,      50,  
/
```

Eta Levels

- Often the user needs to **specify the eta levels** (coordinate this with your model top)
- Use the automatic generation to your advantage
- Specify how many levels **ABOVE the PBL** that you require. Add 8 to this value. For example, you require 50 vertical levels above the PBL.

&domains

e_vert = 58, 58, 58,
/

Eta Levels

- Run the real program (single or **small domain, one time level**), make sure the level thicknesses are OK (< 1000 m)

Converged znw(kte) should be about 0.0 = -5.2081142E-04

Full level index =	1	Height =	0.0 m	
Full level index =	2	Height =	56.6 m	Thickness = 56.6 m
Full level index =	3	Height =	137.9 m	Thickness = 81.4 m
Full level index =	4	Height =	244.7 m	Thickness = 106.8 m
Full level index =	5	Height =	377.6 m	Thickness = 132.9 m
Full level index =	6	Height =	546.3 m	Thickness = 168.7 m
Full level index =	7	Height =	761.1 m	Thickness = 214.8 m
Full level index =	8	Height =	1016.2 m	Thickness = 255.0 m
Full level index =	9	Height =	1207.1 m	Thickness = 190.9 m
Full level index =	10	Height =	1401.8 m	Thickness = 194.6 m
Full level index =	11	Height =	1600.3 m	Thickness = 198.5 m
Full level index =	12	Height =	1802.8 m	Thickness = 202.5 m
Full level index =	13	Height =	2196.1 m	Thickness = 393.3 m

Eta Levels

- Get the computed levels from ncdump, after running the real program

```
> ncdump -v ZNW wrfinput_d01
```

```
data:
```

```
ZNW =
```

```
1, 0.993, 0.983, 0.97, 0.954, 0.934, 0.909, 0.88, 0.8587637, 0.8375274,  
0.8162911, 0.7950548, 0.7550299, 0.7165666, 0.6796144, 0.6441237,  
0.6100466, 0.5773363, 0.5459476, 0.5158363, 0.4869595, 0.4592754,  
0.4327437, 0.407325, 0.382981, 0.3596745, 0.3373697, 0.3160312,  
0.2956253, 0.2761188, 0.2574798, 0.2396769, 0.2226802, 0.2064602,  
0.1909885, 0.1762376, 0.1621807, 0.1487919, 0.1360459, 0.1239184,  
0.1124378, 0.1017038, 0.09166772, 0.08228429, 0.07351105, 0.06530831,  
0.05763897, 0.05046835, 0.04376402, 0.03749565, 0.0316349, 0.02615526,  
0.02103195, 0.01624179, 0.01176313, 0.007575703, 0.003660574, 0 ;
```

Eta Levels

- Re-run the real program (all domains, all time periods) with the new levels in the nml variable **eta_levels**
- Replace the **PBL values** with those of your choosing.
- Augment the number of vertical levels (e_vert)
- Note that both e_vert and eta_levels are **full levels**

Eta Levels

```
&domains
eta_levels =
  1, 0.993, 0.983, 0.97, 0.954, 0.934, 0.909, 0.88,
  0.8587637, 0.8375274,
  0.8162911, 0.7950548, 0.7550299, 0.7165666, 0.6796144, 0.6441237,
  0.6100466, 0.5773363, 0.5459476, 0.5158363, 0.4869595, 0.4592754,
  0.4327437, 0.407325, 0.382981, 0.3596745, 0.3373697, 0.3160312,
  0.2956253, 0.2761188, 0.2574798, 0.2396769, 0.2226802, 0.2064602,
  0.1909885, 0.1762376, 0.1621807, 0.1487919, 0.1360459, 0.1239184,
  0.1124378, 0.1017038, 0.09166772, 0.08228429, 0.07351105, 0.06530831,
  0.05763897, 0.05046835, 0.04376402, 0.03749565, 0.0316349, 0.02615526,
  0.02103195, 0.01624179, 0.01176313, 0.007575703, 0.003660574, 0
/
```

- Maybe replace with

```
1, 0.999, 0.998, 0.996, 0.993, 0.990, 0.980, 0.970, 0.960, 0.950,
  0.940, 0.930, 0.920, 0.910, 0.900, 0.890, 0.880, 0.870,
```

Eta Levels

- For **vertical nesting refinement**, follow the similar procedure for each domain.
- **Each domain** will need a specification of eta levels
- The assignment of the single **eta_levels array is split** into pieces for easier understanding

Eta Levels

```
&domains
max_dom          = 2,

e_vert           = 35,      45,
eta_levels(1:35) = 1., 0.993, 0.983, 0.97, 0.954, 0.934,
                  0.909, 0.88, 0.840, 0.801, 0.761, 0.722,
                  0.652, 0.587, 0.527, 0.472, 0.421, 0.374,
                  0.331, 0.291, 0.255, 0.222, 0.191, 0.163,
                  0.138, 0.115, 0.095, 0.077, 0.061, 0.047,
                  0.035, 0.024, 0.015, 0.007, 0.
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.
vert_refine_method = 0,      2,
```

The metgrid Flags

- The **real program and the WRF model** are able to communicate directly through the **Registry** file
- The real program is only able to talk with the **metgrid** program through the **input data** stream
- Specific information about the incoming data is contained in **special flags** that the user may set in the metgrid table file – usually, related to THIS VARIABLE EXISTS

```
=====
name=PMSL
      interp_option=sixteen_pt+four_pt+average_4pt
      flag_in_output=FLAG_SLP
=====
```

The metgrid Flags

```
> ncdump -h met_em.d01.2000-01-24_12:00:00.nc | grep FLAG
:FLAG_METGRID = 1 ;
:FLAG_EXCLUDED_MIDDLE = 0 ;
:FLAG_SOIL_LAYERS = 1 ;
:FLAG_SNOW = 1 ;
:FLAG_PSFC = 1 ;
:FLAG_SM000010 = 1 ;
:FLAG_SM010040 = 1 ;
:FLAG_SM040100 = 1 ;
:FLAG_SM100200 = 1 ;
:FLAG_ST000010 = 1 ;
:FLAG_ST010040 = 1 ;
:FLAG_ST040100 = 1 ;
:FLAG_ST100200 = 1 ;
:FLAG_SLP = 1 ;
:FLAG_TAVGSFC = 1 ;
:FLAG_QNWFA = 1 ;
:FLAG_QNIFA = 1 ;
:FLAG_SOILHGT = 1 ;
:FLAG_MF_XY = 1 ;
```

The metgrid Flags

- The real program uses this **information** when deciding how to do many operations:
 - Is the input from metgrid?
 - Method to compute surface pressure
 - Use RH vs mixing ratio vs specific humidity computations
 - Excluded middle processing
 - Average surface air temperature for lake temperatures
 - Water/Ice friendly vertical interpolation
 - Which levels of soil data are present
- All **flags** for the metgrid to real data transfer are contained in **share/module_optional_input.F**

The metgrid Flags

```
flag_slp      = 0
```

```
flag_name(1:8) = 'SLP      '
```

```
CALL wrf_get_dom_ti_integer ( fid, 'FLAG_' // &  
    flag_name, itmp, 1, icnt, ierr )
```

```
IF ( ierr .EQ. 0 ) THEN
```

```
    flag_slp      = itmp
```

```
END IF
```

Adding a Variable for Vertical Interpolation

- This process is **manual**
- Every new **input 3d variable** that needs to be interpolated needs to have an **explicit block of code** added
- **Mass-point variables** (such as would be used in all physics schemes) are straight forward, as they may be largely copied using the existing templates already in place
- Most vertical interpolation options are supplied from the namelist.input file
- All interpolation is handled in **dry pressure**

Adding a Variable for Vertical Interpolation

```
CALL vert_interp ( grid%t_gc , grid%pd_gc , &
  grid%t_2 , grid%pb , &
  grid%tmaxw , grid%ttrop , grid%pmaxw , grid%ptrop , &
  grid%pmaxwnn , grid%ptropnn , &
  flag_tmaxw , flag_ttrop , &
  config_flags%maxw_horiz_pres_diff , &
  config_flags%trop_horiz_pres_diff , &
  config_flags%maxw_above_this_level , &
  num_metgrid_levels , 'T' , &
  interp_type , lagrange_order , t_extrap_type , &
  lowest_lev_from_sfc , use_levels_below_ground , &
  use_surface , zap_close_levels , force_sfc_in_vinterp , &
  ids , ide , jds , jde , kds , kde , &
  ims , ime , jms , jme , kms , kme , &
  its , ite , jts , jte , kts , kte )
```

Tracers

- The WRF model is able to **advect arrays of passive scalars** (tracer 4d array)
- As with all other variables going into the WRF model, this data is available to **be set in the real program**
- These variables must be **coordinated with the Registry names**, as the tracer index is an automatically manufactured name

```
# Tracer Scalars
```

```
#
```

```
state real tr17_1 ikjftb  tracer 1 -  irhusdf=(bdy_interp:dt) \  
    "tr17_1" "tr17_1" "Dimensionless"
```

Tracers

- As with all 4d arrays, no space is allocated unless the packaged variables are requested for processing at run-time

```
package    tracer_test1  tracer_opt==2 -  tracer:tr17_1
```

Tracers

```
! Template for initializing tracer arrays.  
! A small plane in the middle of the domain at  
! lowest model level is defined.
```

```
IF (config_flags%tracer_opt .eq. 2) THEN  
  DO j = (jde + jds)/2 - 4, (jde + jds)/2 + 4, 1  
    DO i = (ide + ids)/2 - 4, (ide + ids)/2 + 4, 1  
      IF ( ( its .LE. i .and. ite .GE. i ) .and. &  
          ( jts .LE. j .and. jte .GE. j ) ) THEN  
        tracer(i, 1, j, P_tr17_1) = 1.  
      END IF  
    END DO  
  END DO  
END IF
```

Trajectories

- The user may **specify (i,j,k) locations** in the model domain to follow parcels: traj_i, traj_j, traj_k (hard coded in the module_initialize_real.F file)
- The current **number of trajectory locations** is small, 25, and is a run-time option that the **user sets in the nml file**

&domain

num_traj = 25,

&physics

traj_opt = 1,

Trajectories

- The trajectory code uses the lat,lon locations, so the initial (i,j) value of the lat,lon is assigned

```
IF (config_flags%num_traj .gt. 0 .and.  
    config_flags%traj_opt .gt. 0) THEN  
  DO j = (jde + jds)/2 - 2, (jde + jds)/2 + 2, 1  
    DO i = (ide + ids)/2 - 2, (ide + ids)/2 + 2, 1  
      IF ( its .LE. i      .and. ite .GE. i      .and. &  
          jts .LE. j      .and. jte .GE. j ) THEN  
        grid%traj_i    (icount) = i  
        grid%traj_j    (icount) = j  
        grid%traj_k    (icount) = 10  
        grid%traj_lat  (icount) = grid%xlat(i,j)  
        grid%traj_long(icount) = grid%xlong(i,j)  
      END IF  
    END DO  
  END DO  
END IF
```

Options

- When there are **strong normal topo gradients** along the outer rows and columns of the most-coarse domain, smoothing the topography to match the incoming first guess data is a good idea.
- This is **the same** sort processing that is done to make the child and parent domains more consistent in the area of the **LBC** forcing

&domains

smooth_cg_topo = .true.

/

Options

- **Time varying fields** for longer simulations are available from the technique set up for “SST Update”
- A new field will be automatically added to the input file to the WRF model (provided by the real program) with a few changes to the Registry file (**Registry.EM_COMMON**), specifying **stream 4**

```
state  real my_new_field  ij misc 1  -  \  
i024rhdu "MY_NEW_FIELD" \  
"SOME DESCRIPTION" "SOME UNITS"
```

Options

- Information for **using time varying data** is specified at run-time in the namelist file

```
&time_control
```

```
  auxinput4_inname      = "wrflowinp_d<domain>"
```

```
  auxinput4_interval    = 360
```

```
  io_form_auxinput4     = 2
```

```
&physics
```

```
  sst_update            = 1
```

Real program in a nutshell: PART 2

- Access to everything
- Eta levels
- Metgrid flags
- Adding a variable for vertical interpolation
- Vertical interpolation
- Tracers
- Trajectories
- Options

Real program in a nutshell: PART 2

- Access to everything The Derived Data Type: grid
- Eta levels
- Metgrid flags Example: grid%sst
- Adding a variable for vertical interpolation
- Vertical interpolation
- Tracers
- Trajectories
- Options

Real program in a nutshell: PART 2

- Access to everything Completely user defined
- Eta levels
- Metgrid flags May be different per domain
- Adding a variable for vertical interpolation
- Vertical interpolation
- Tracers Be careful of the thicknesses
- Trajectories
- Options Tightly coupled with the model lid

Real program in a nutshell: PART 2

- Access to everything
 - Eta levels
 - Metgrid flags
 - Adding a variable for vertical interpolation
 - Vertical interpolation
 - Tracers
 - Trajectories
 - Options
- The metgrid program provides flags for some internal communication real to metgrid
- These flags are defined inside the METGRID.TBL file (for WPS) and in the file share/
module_optional_input.F (real)

Real program in a nutshell: PART 2

- Access to everything
 - Eta levels
 - Metgrid flags
 - Adding a variable for vertical interpolation
 - Vertical interpolation
 - Tracers
 - Trajectories
 - Options
- Requires new code inside real
Examples are easily available

Real program in a nutshell: PART 2

- Access to everything Always in dry pressure
- Eta levels
- Metgrid flags Input vertical coordinate neutral
- Adding a variable for vertical interpolation
- Vertical interpolation
- Tracers
- Trajectories
- Options

Real program in a nutshell: PART 2

- Access to everything
 - Eta levels
 - Metgrid flags
 - Adding a variable for vertical interpolation
 - Vertical interpolation
 - Tracers
 - Trajectories
 - Options
- Simple way to initialize passive scalars
- Users should provide info for which tracers in the Registry, and select the accompanying option in the namelist

Real program in a nutshell: PART 2

- Access to everything
 - Eta levels
 - Metgrid flags
 - Adding a variable for vertical interpolation
 - Vertical interpolation
 - Tracers
 - Trajectories
 - Options
- A simple (i,j,k) initialization for the starting locations of trajectory points is available
- Choose the number of trajectory points

Real program in a nutshell: PART 2

- Access to everything
 - Eta levels
 - Metgrid flags
 - Adding a variable for vertical interpolation
 - Vertical interpolation
 - Tracers
 - Trajectories
 - Options
- Users may smooth the outer rows and columns so that the topography on the coarse grid and the external model are consistent
- Users may add variables to streams easily, an example is that the SST update option could have a new field included (for example, soil moisture)