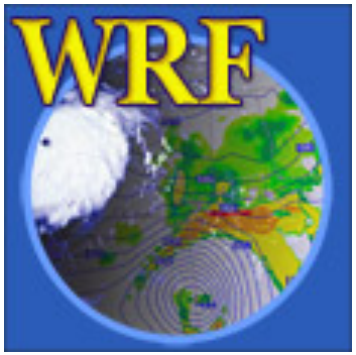
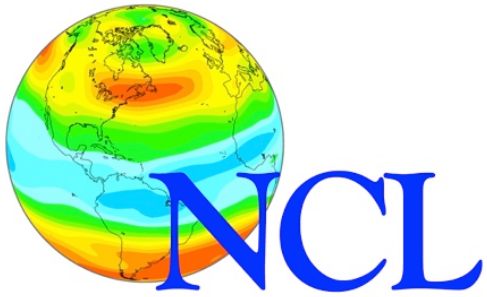


Post-processing WRF-ARW data using NCL



15th Annual WRF User's Workshop

Mary Haley • CISL / Cindy Bruyère & Abby Jaye • NESL

June 23-27, 2014

The National Center for Atmospheric Research is sponsored by the National Science Foundation.

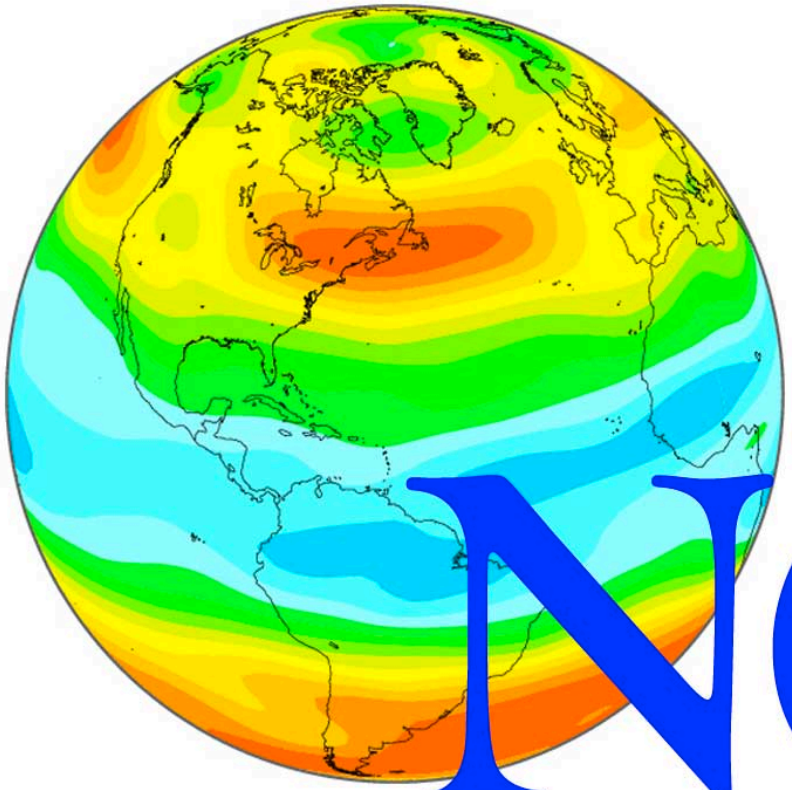
Four main goals

1. Introduce you to NCL and WRF-NCL
2. *Get you familiar with WRF-NCL scripts*
   - *Opening and examining a WRF output data file*
   - *Reading and querying variables*
   - *Plotting variables*
3. Sneak in tips and information for existing users
4. Tell you what's new in V6.2.0 and future plans

Topics

- Overview
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Tips, where to get help
- What's new

A scripting language developed at NCAR and tailored for the analysis and visualization of geoscientific data



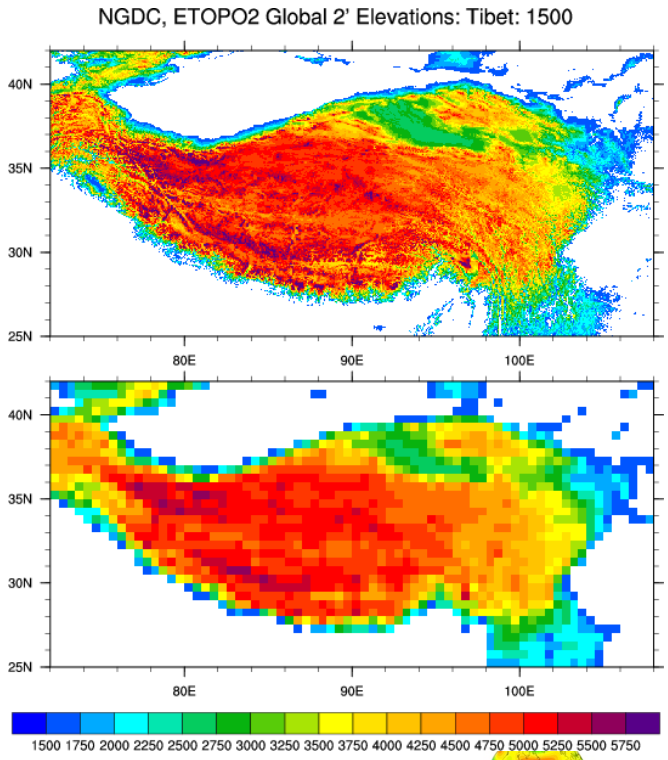
http://www.ncl.ucar.edu/

NCL

NCAR Command Language

NCL

- Developed in NCAR/CISL in close collaboration with CGD & MMM staff
- Open source, binaries and source available
- Extensive NCL website, hundreds of examples
- Hands-on workshops
- Well-supported



What is NCL?

- A scripting language similar to Matlab, Python, IDL
- Tailored to climate and weather sciences
- Has variable types, “if-then-end if”, “do” loops, arithmetic operators
- F90-like array arithmetic that automatically ignores missing values (where it makes sense)
- Can call your own Fortran 77/90 or C routines

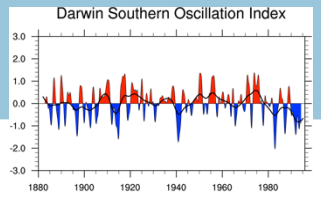
1. Simple, robust file input/output
2. Hundreds of data analysis routines
3. Publication-quality graphics that are highly customizable

NCL: File input and output

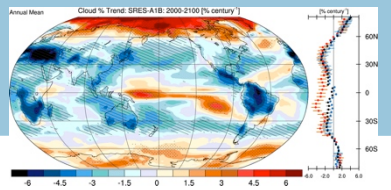
- Data model based on netCDF model (metadata describes data)
- **One function** reads all supported data formats:
  - NetCDF3, GRIB 1 and 2, HDF4, HDF5, HDF-EOS2, HDF-EOS5, shapefiles, NetCDF4 (groups, compound data, variable length arrays)
  - Writes NetCDF3, NetCDF4, and HDF4
- OPeNDAP-enabled client available
- ASCII, binary (read and write)
- "Never fear a data format"

NCL: Data analysis

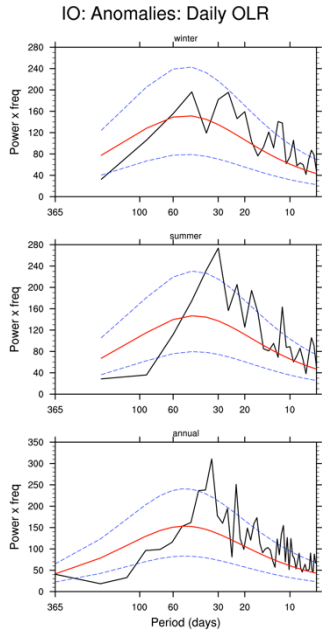
- Array-based math
- Hundreds of functions
  - WRF-ARW specific functions ("`wrf_user_getvar`" is one)
  - Spherical harmonics
  - Scalar and vector regridding
  - Vertical interpolation
  - EOFs
- Many tailored to geosciences
- Most automatically handle missing data
- Can call C and Fortran routines - WRAPIT

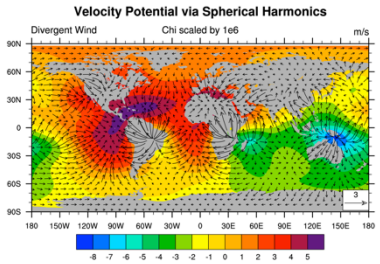


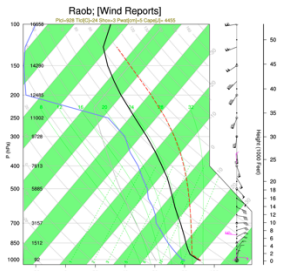
NCL: Visualization

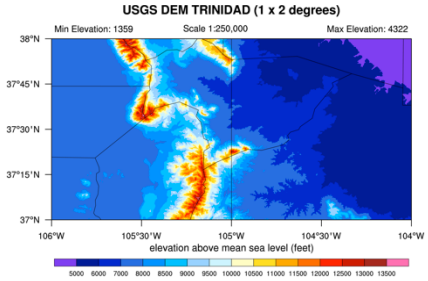


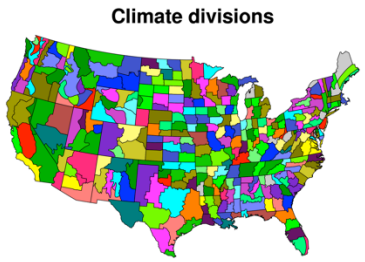
- High-quality and customizable visualizations
- Contours, XY, vectors, wind barbs, streamlines
- Maps with common map projections
- Handles data on rectilinear, curvilinear, and unstructured grids (MPAS, triangular meshes)
- Specialized scripts for meteograms, skew-T, wind roses, histograms, cross section, panels
- Suite of `wrf_xxxx` functions: simplifies visualization for WRF-ARW data
- Over 1,400 visualization "options"





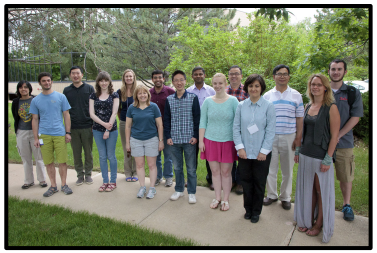






NCL Training Workshops

- First training workshop in 2000 (72 total, 1120 attendees)
  - 4-5 local workshops a year
  - 1-2 annual workshops at U.S. universities
  - One annual invited international workshop
- Lectures taught by a scientist and a software engineer
- Includes special lecture on various data formats used in geosciences – lots of students working with WRF!
- Four hands-on labs sessions; students encouraged to bring their own datasets









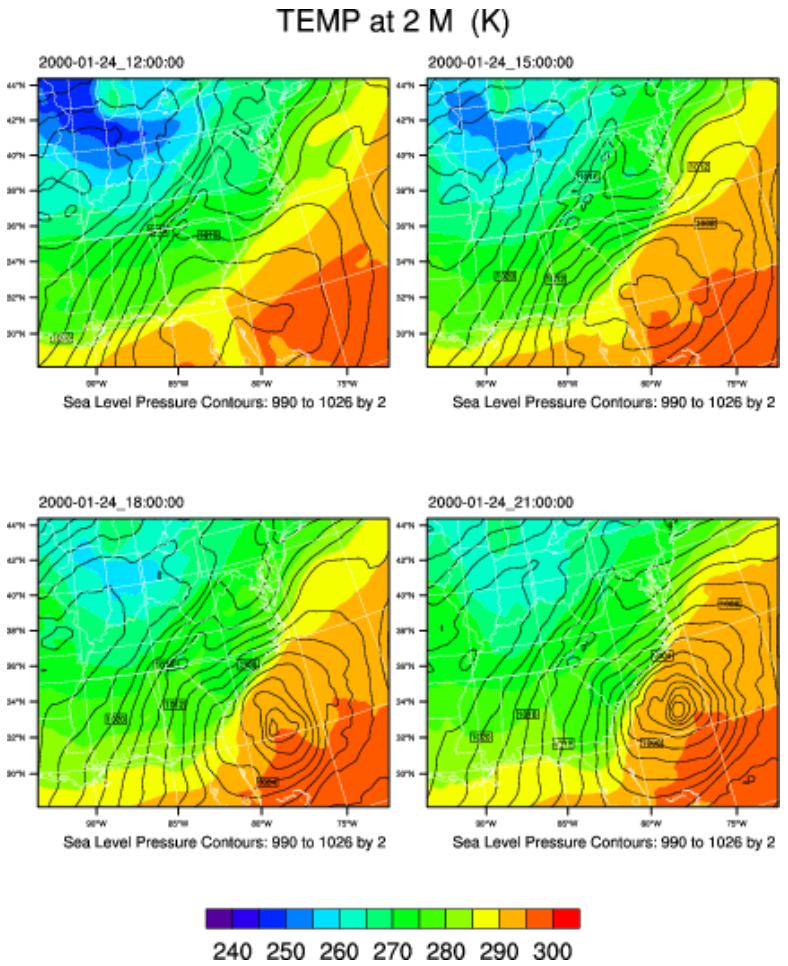

WRF-NCL

Suite of analysis and visualization functions tailored for WRF-ARW model data

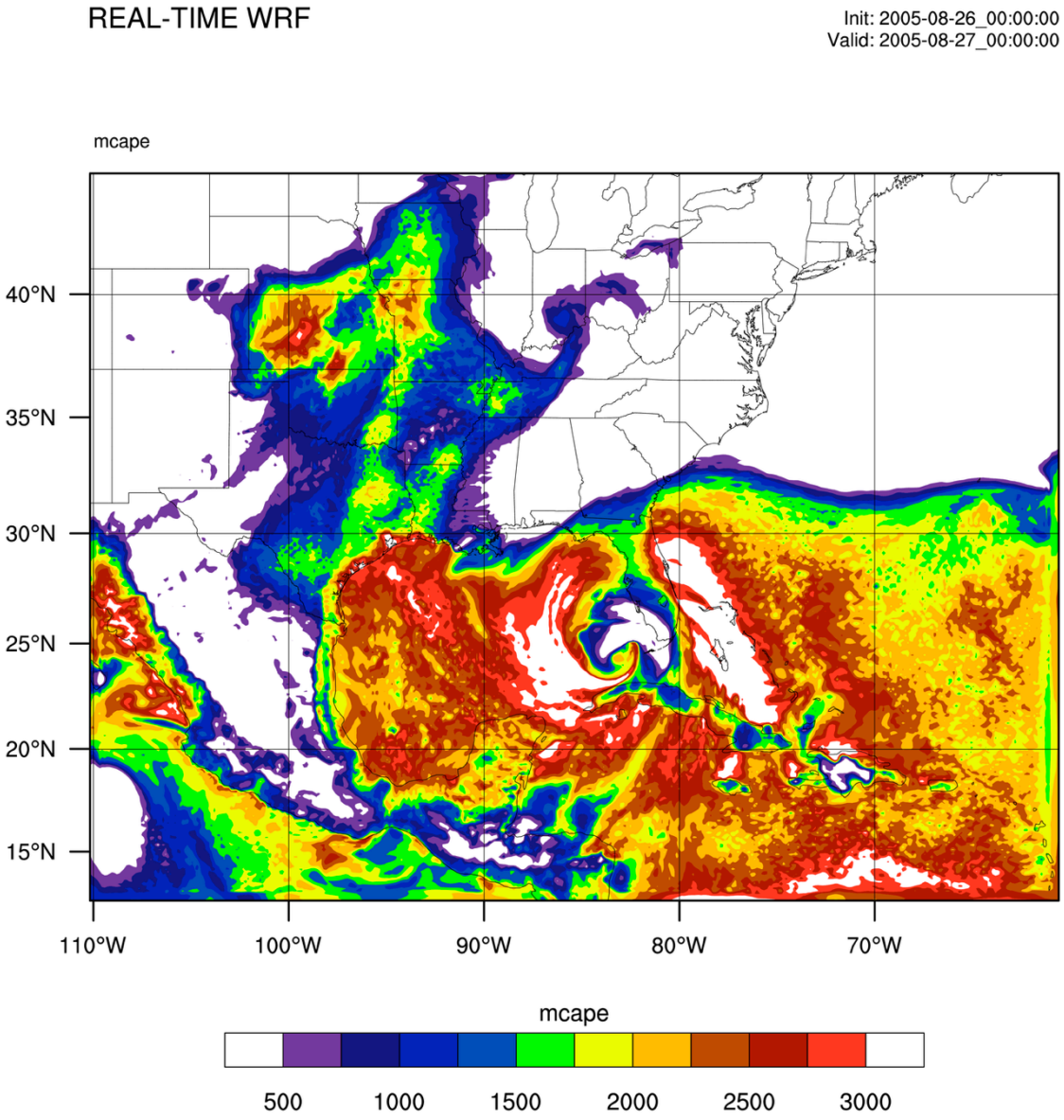
http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/

WRF-NCL Overview

- **Included with NCL since 2006**
- Developed by staff in MMM
- Maintained by MMM and CISL
- Functions for calculating basic diagnostics (**`wrf_user_getvar`**)
- Functions for customized visualizations
- Website with lots of analysis and visualization examples
- Workshops and tutorials

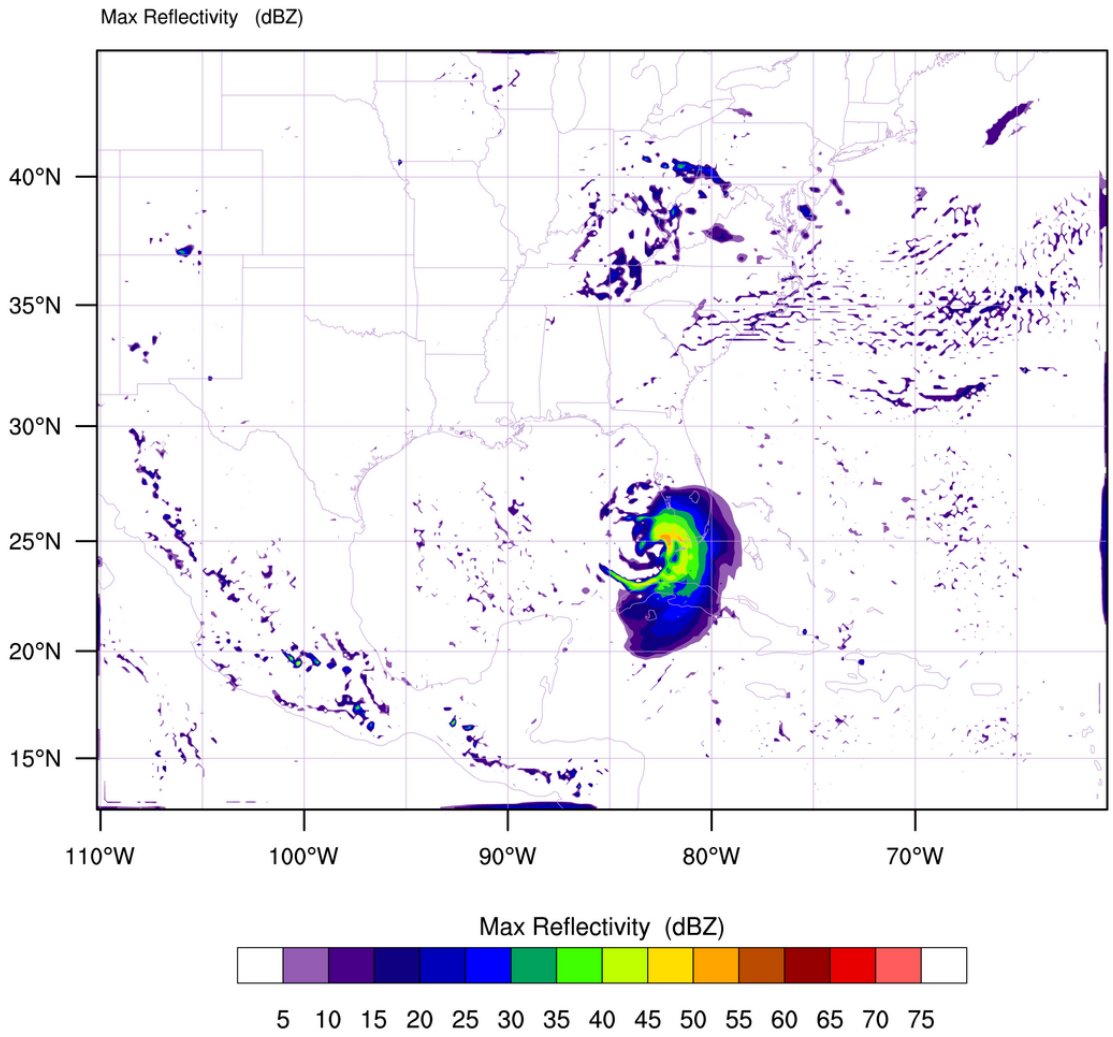


Sample WRF-NCL visualizations



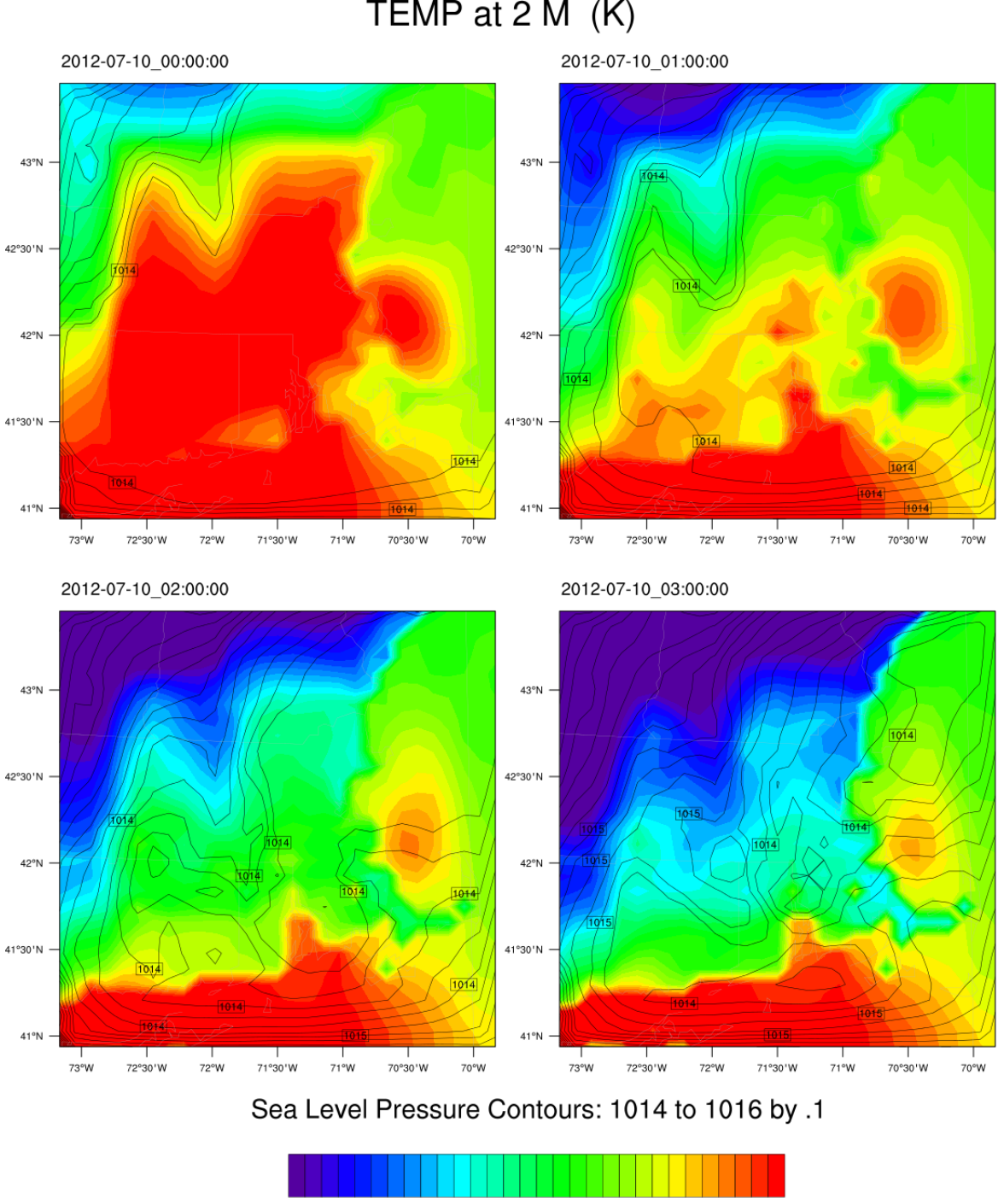
REAL-TIME WRF

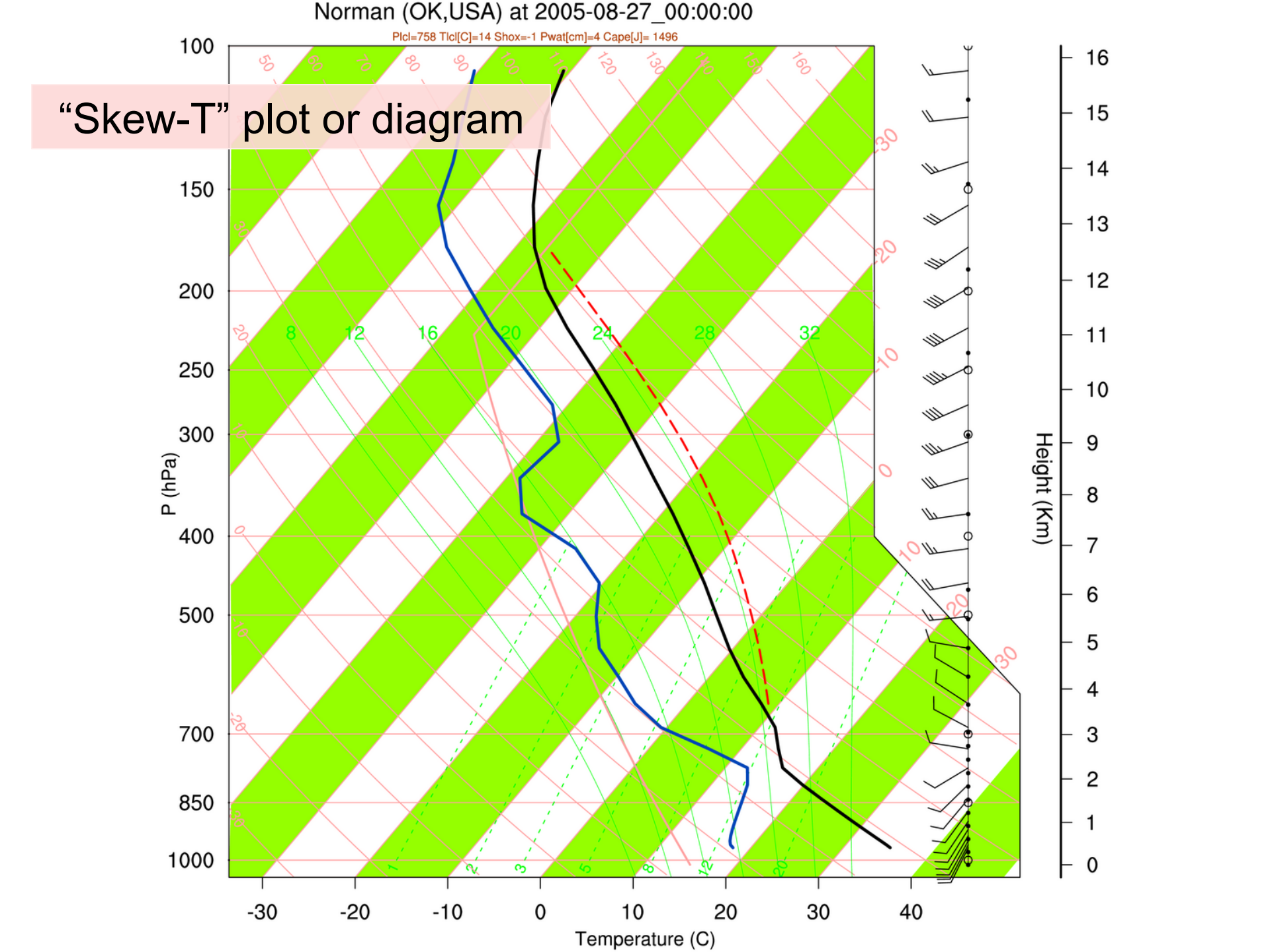
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



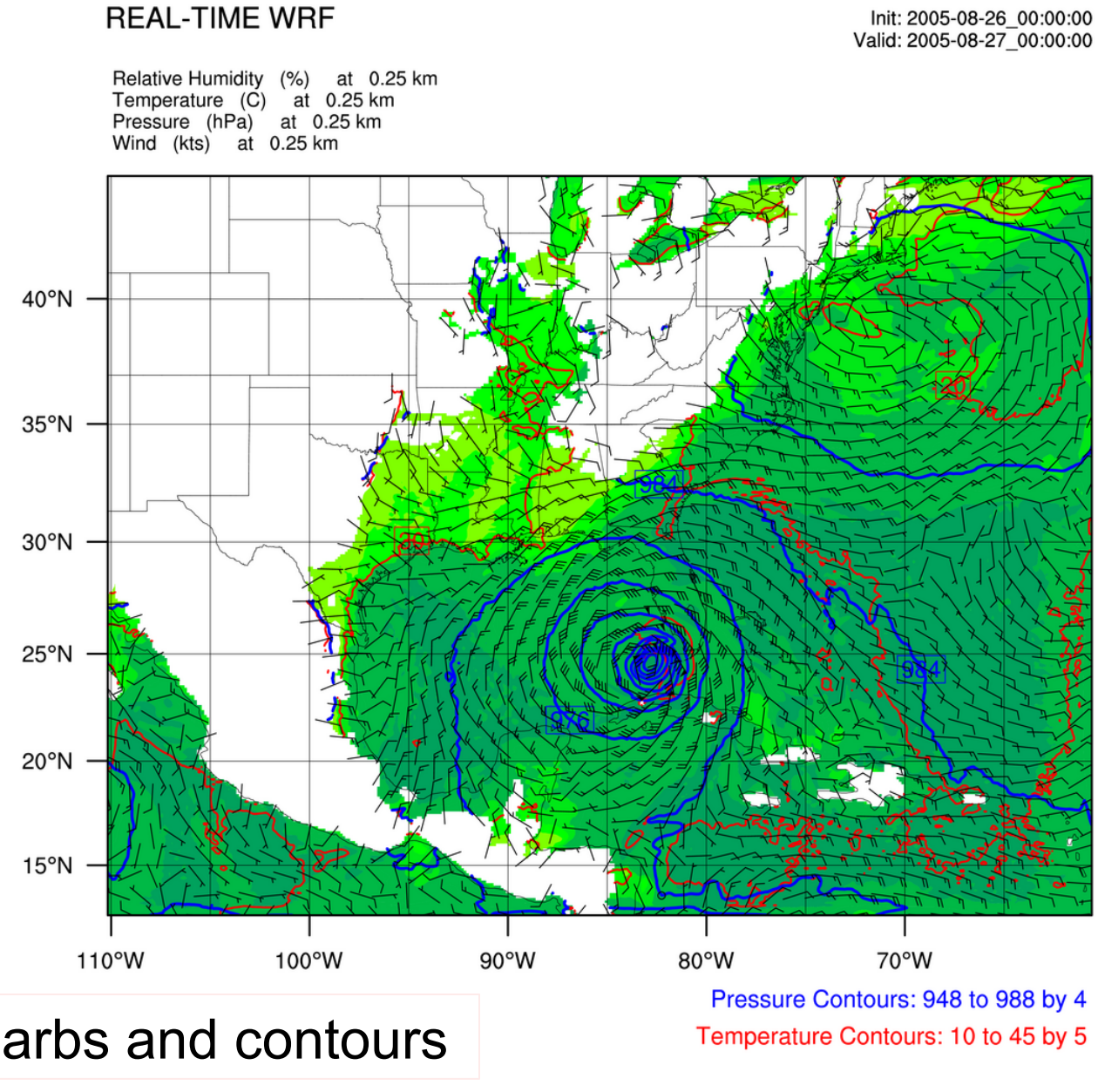
OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

These are called “panel” plots

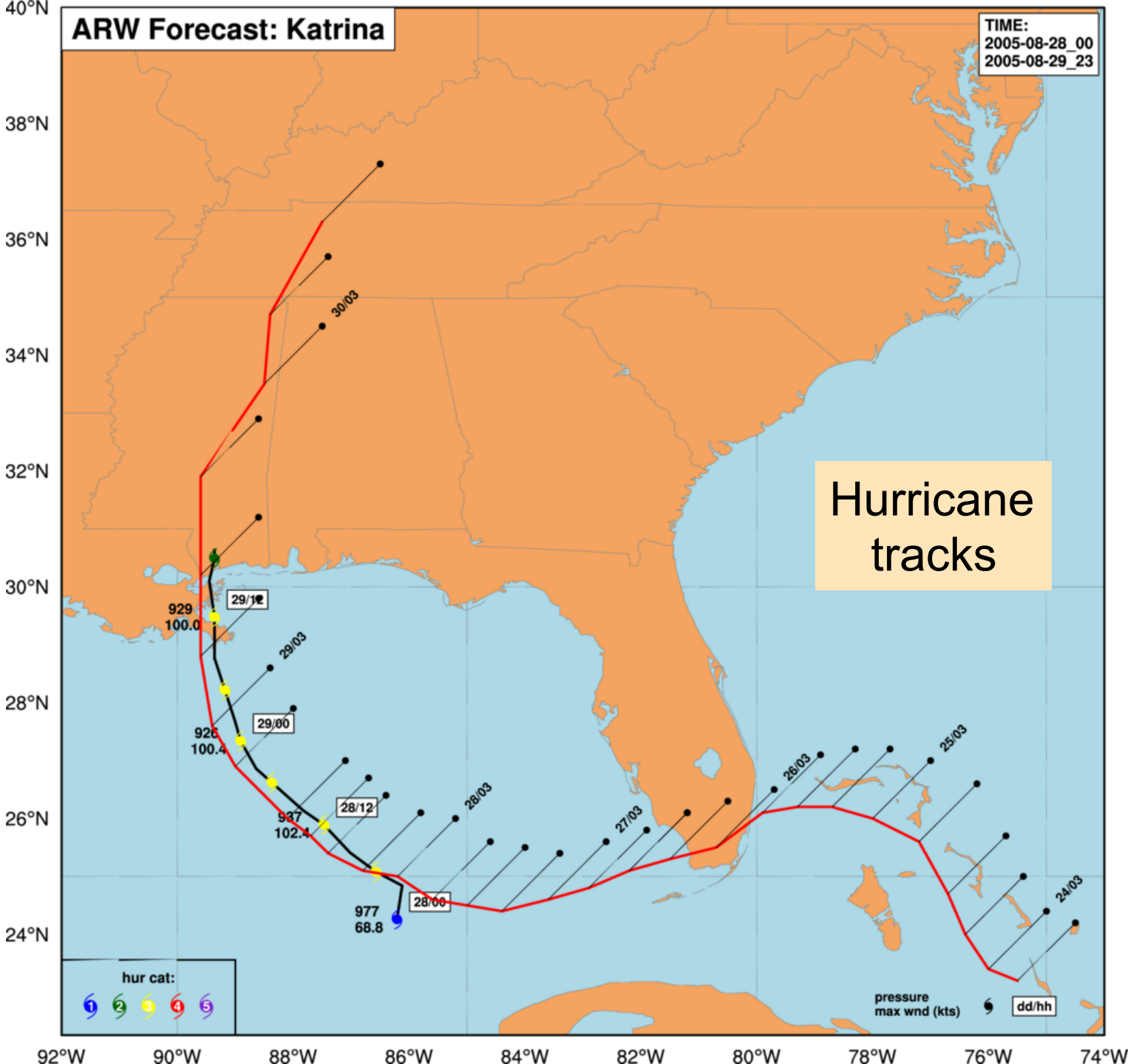




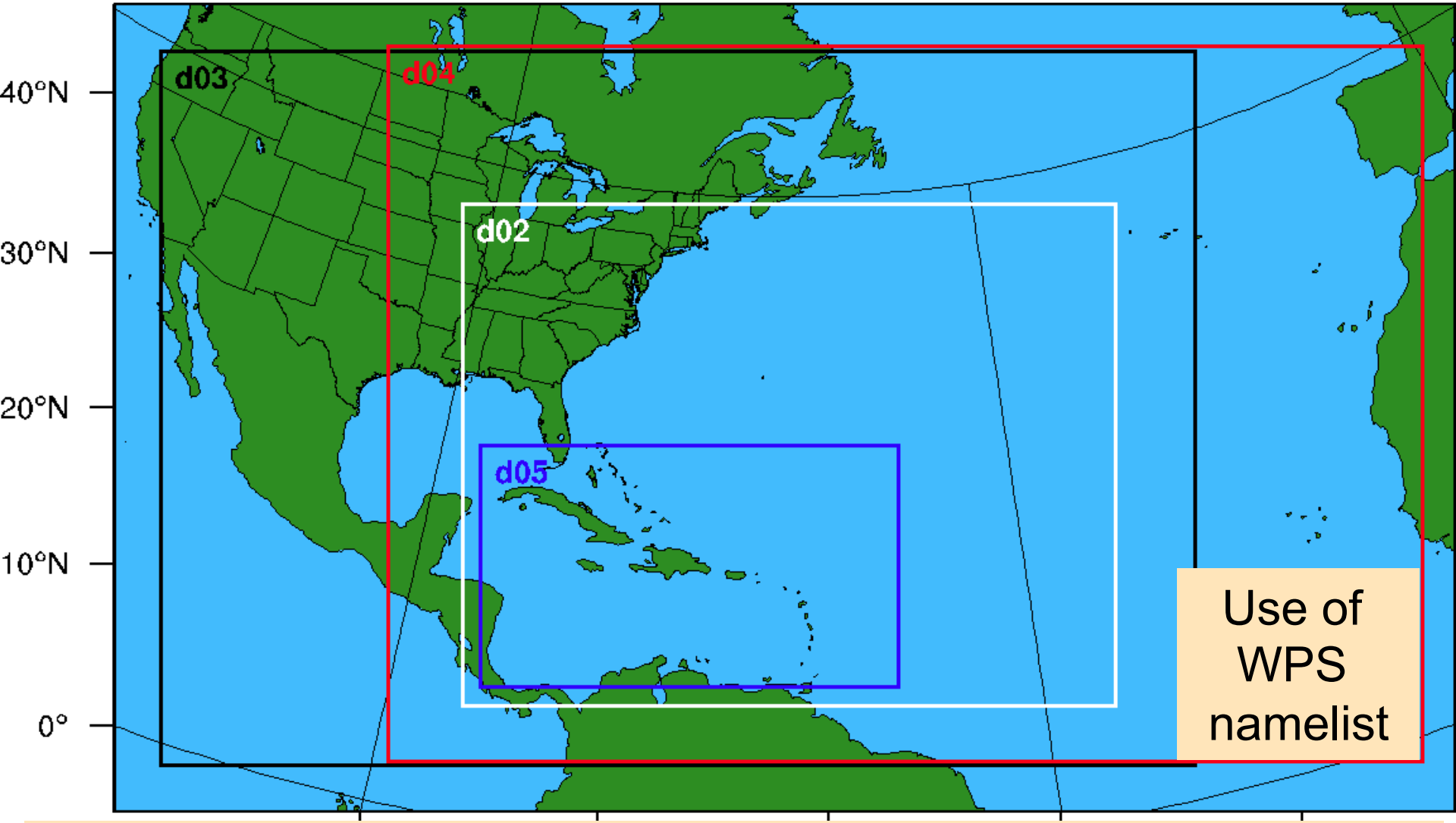
“Skew-T” plot or diagram



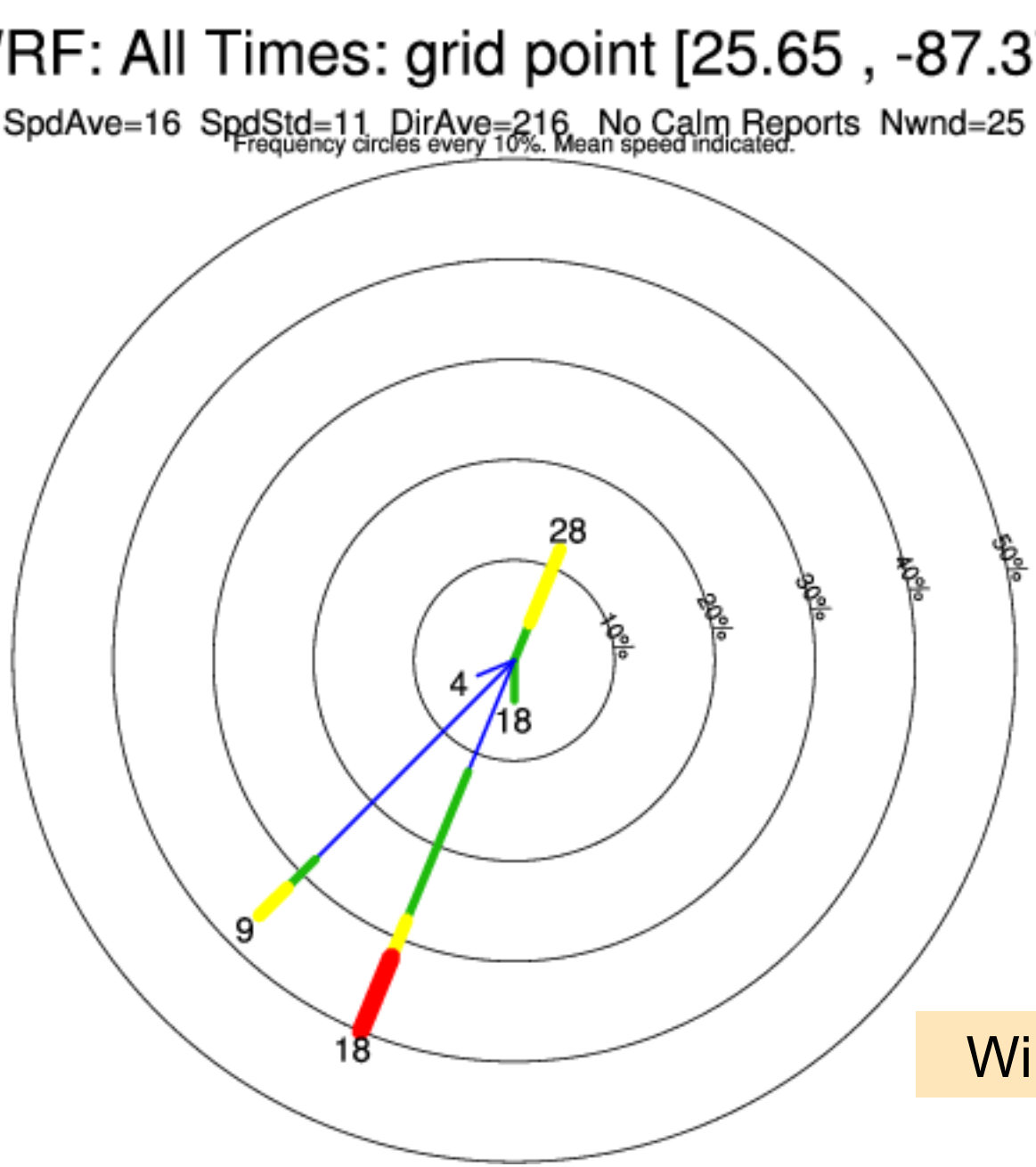
Wind barbs and contours


ARW Forecast: Katrina
TIME:
2005-08-28_00
2005-08-29_23
Hurricane tracks
40°N
38°N
36°N
34°N
32°N
30°N
28°N
26°N
24°N
92°W
90°W
88°W
86°W
84°W
82°W
80°W
78°W
76°W
74°W
30/03
29/12
29/03
29/00
28/12
28/03
28/00
27/03
26/03
25/03
24/03
929
100.0
926
100.4
937
102.4
977
68.8
hur cat:
1
2
3
4
5
pressure
max wnd (kts)
dd/hh

WPS Domain Configuration

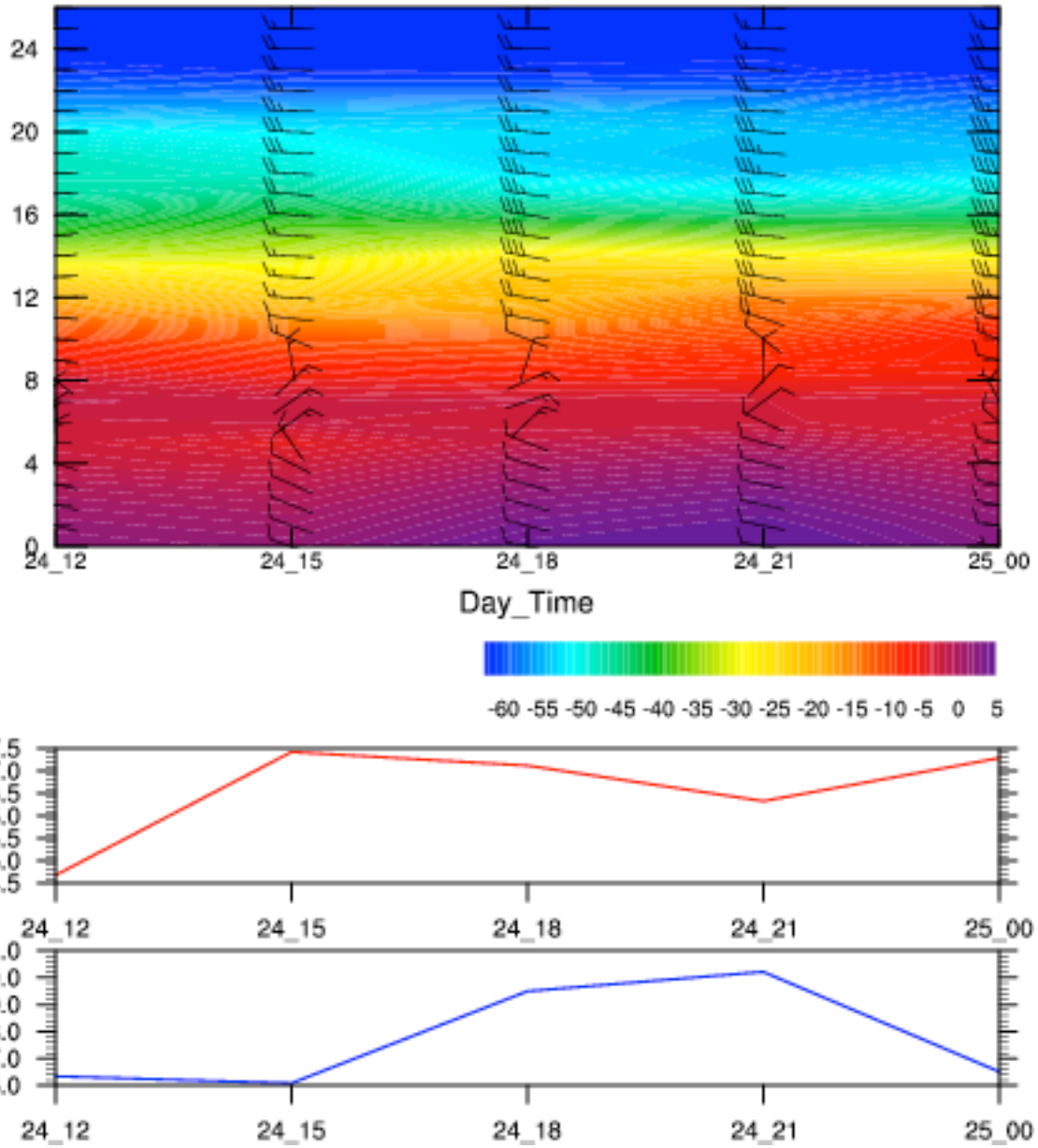


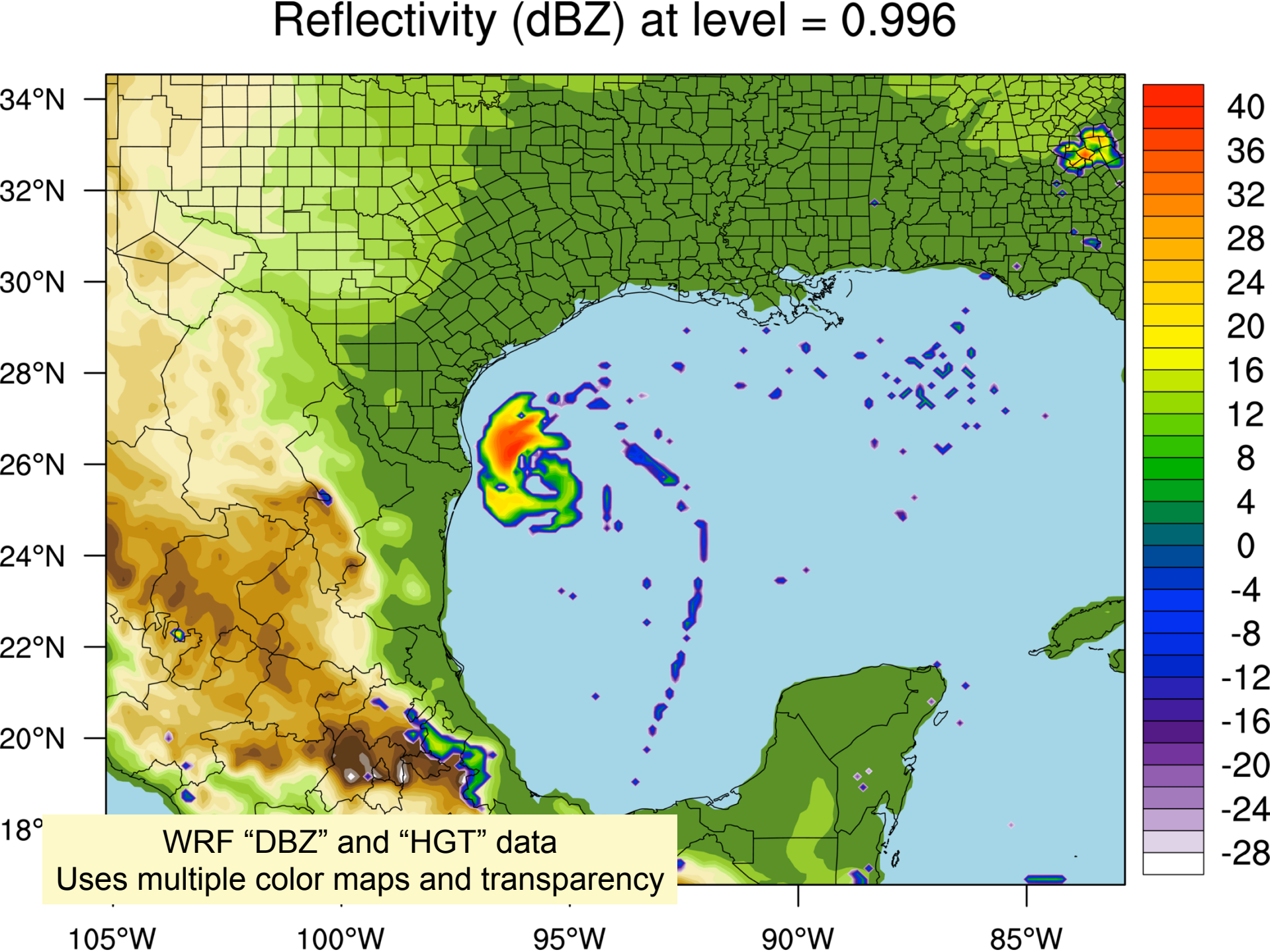
http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/Examples/EXPERIMENTAL/wrf_show_wps_som_namelist.htm

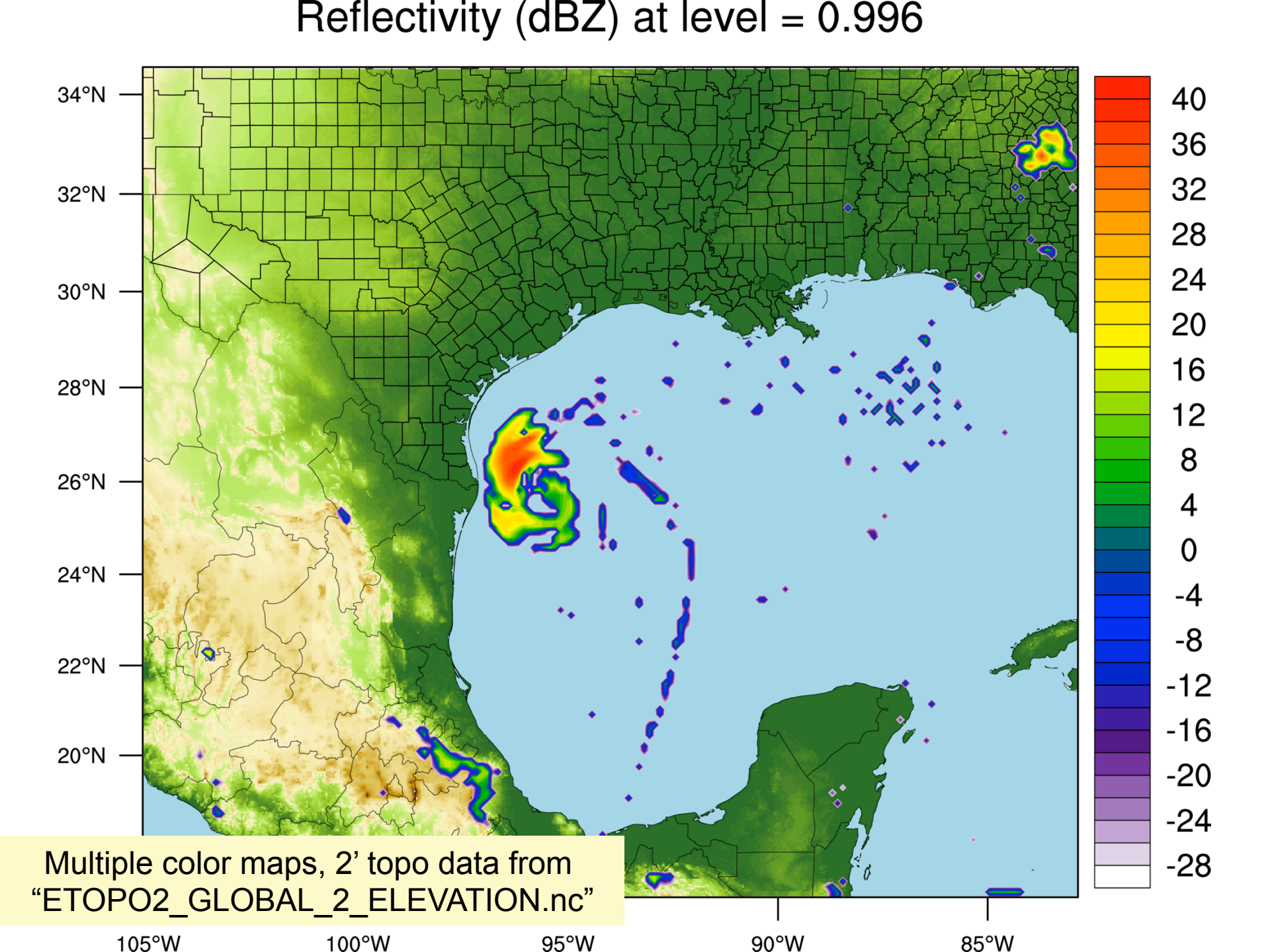

WRF: All Times: grid point [25.65 , -87.37]
SpdAve=16 SpdStd=11 DirAve=216 No Calm Reports Nwnd=25
Frequency circles every 10%. Mean speed indicated.
10%
20%
30%
40%
50%
28
4
18
9
18

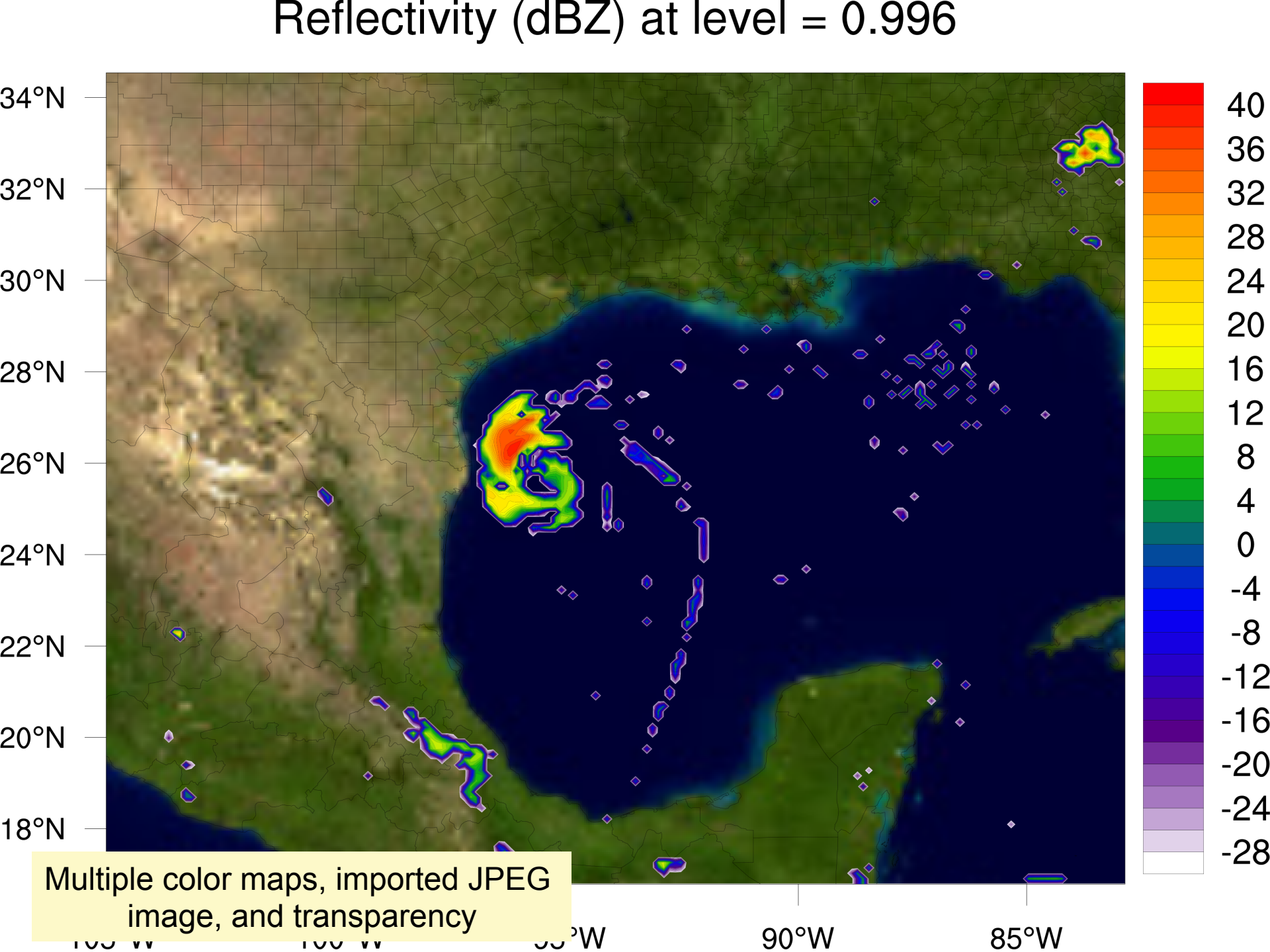
Wind roses

Meteogram for lat=32.5 ; lon=-87



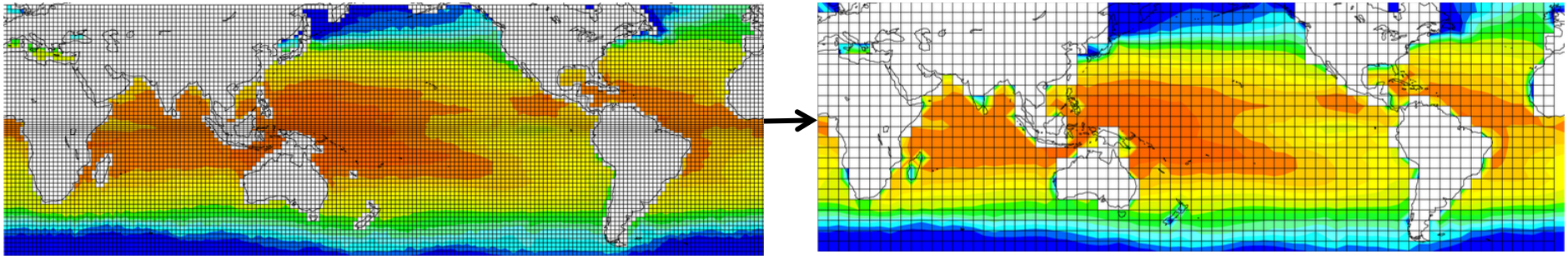

Reflectivity (dBZ) at level = 0.996
34°N
32°N
30°N
28°N
26°N
24°N
22°N
20°N
18°
105°W
100°W
95°W
90°W
85°W
40
36
32
28
24
20
16
12
8
4
0
-4
-8
-12
-16
-20
-24
-28
WRF “DBZ” and “HGT” data
Uses multiple color maps and transparency


Reflectivity (dBZ) at level = 0.996
34°N
32°N
30°N
28°N
26°N
24°N
22°N
20°N
105°W
100°W
95°W
90°W
85°W
40
36
32
28
24
20
16
12
8
4
0
-4
-8
-12
-16
-20
-24
-28
Multiple color maps, 2' topo data from
"ETOPO2_GLOBAL_2_ELEVATION.nc"

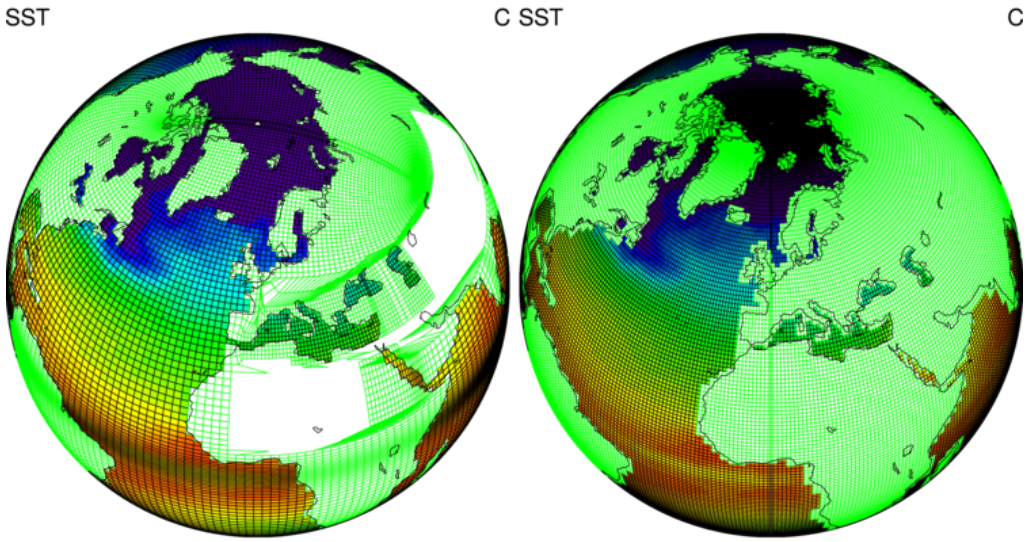

Reflectivity (dBZ) at level = 0.996
34°N
32°N
30°N
28°N
26°N
24°N
22°N
20°N
18°N
90°W
85°W
40
36
32
28
24
20
16
12
8
4
0
-4
-8
-12
-16
-20
-24
-28
Multiple color maps, imported JPEG image, and transparency

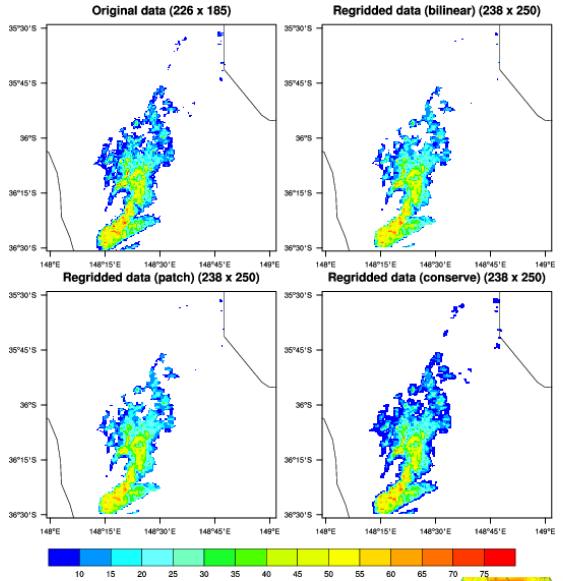
ESMF regridding

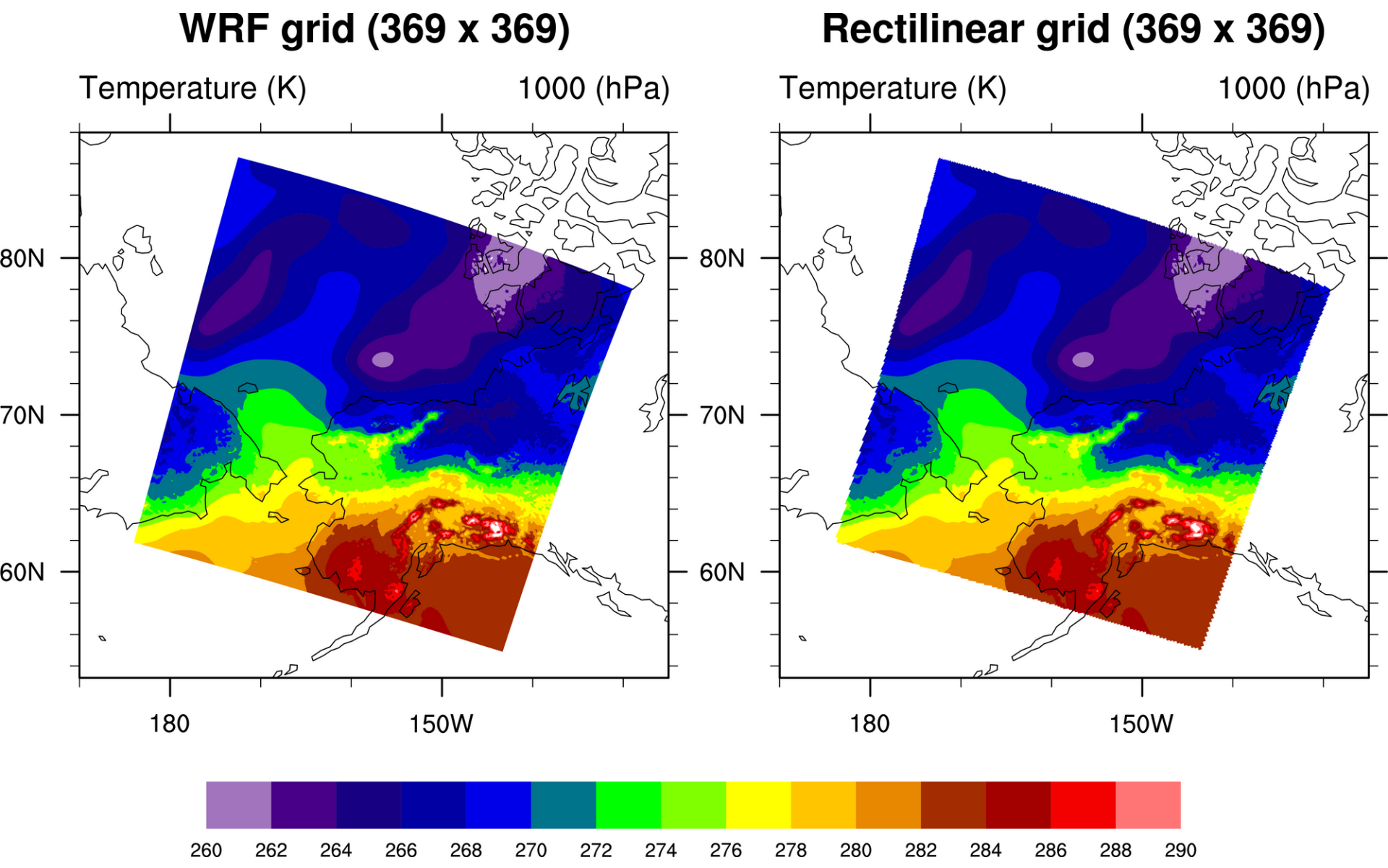
Regridding is the process of interpolating data from one grid (rectangular, rectilinear, curvilinear, unstructured) to another while preserving the qualities of the original data.



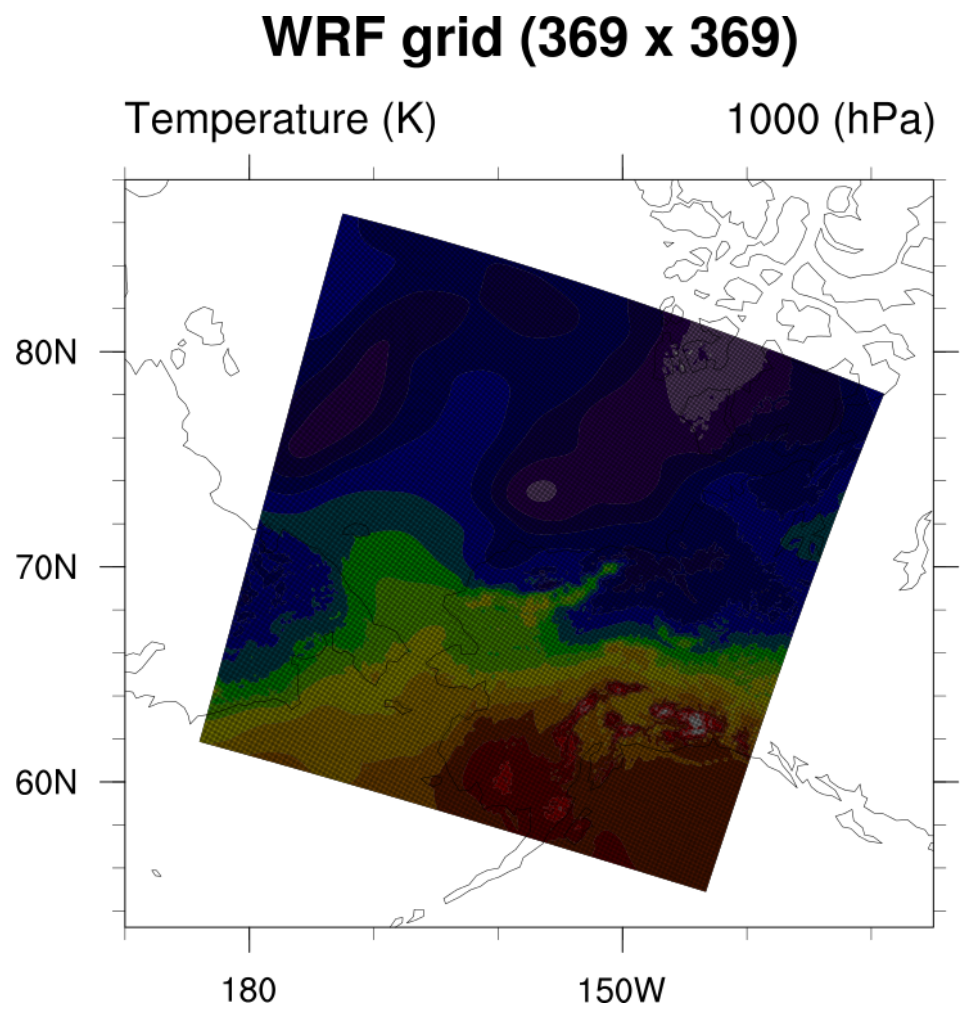
Original ORCA grid (149 x 182) regridded to 1 degree (180 x 360)

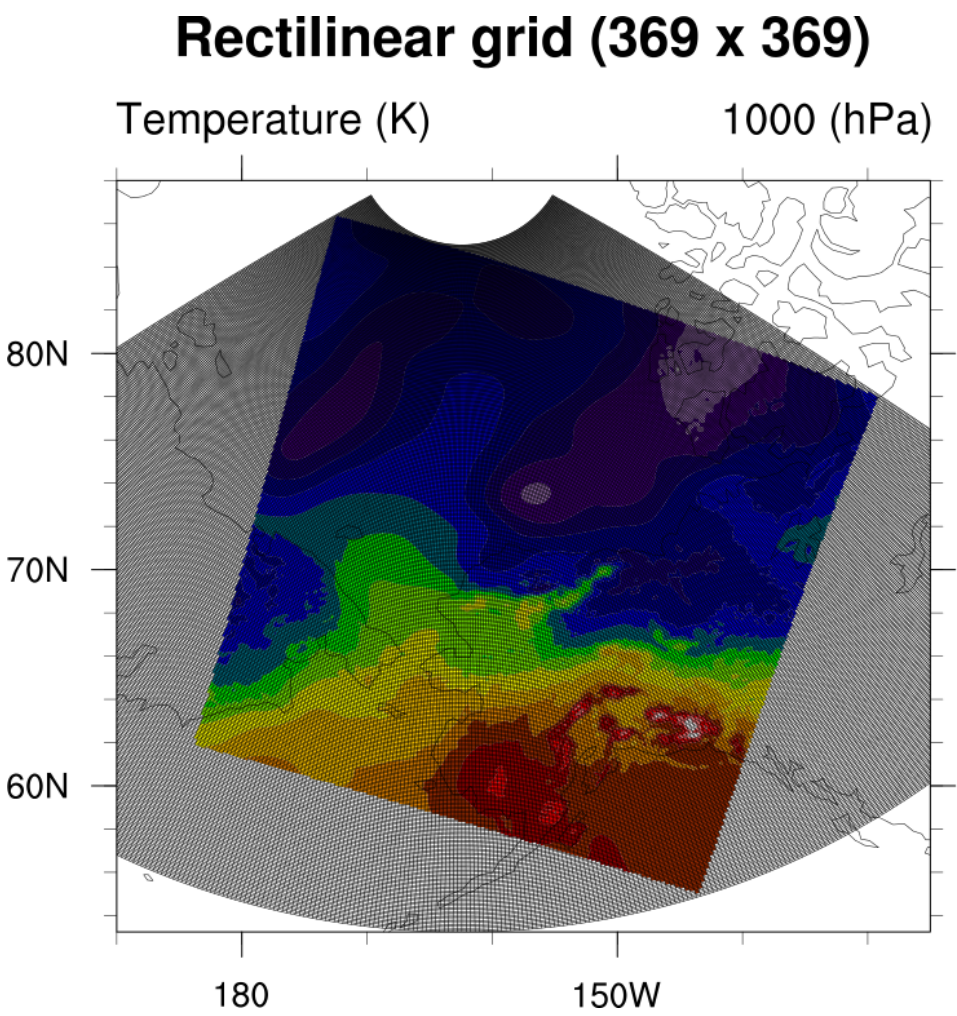




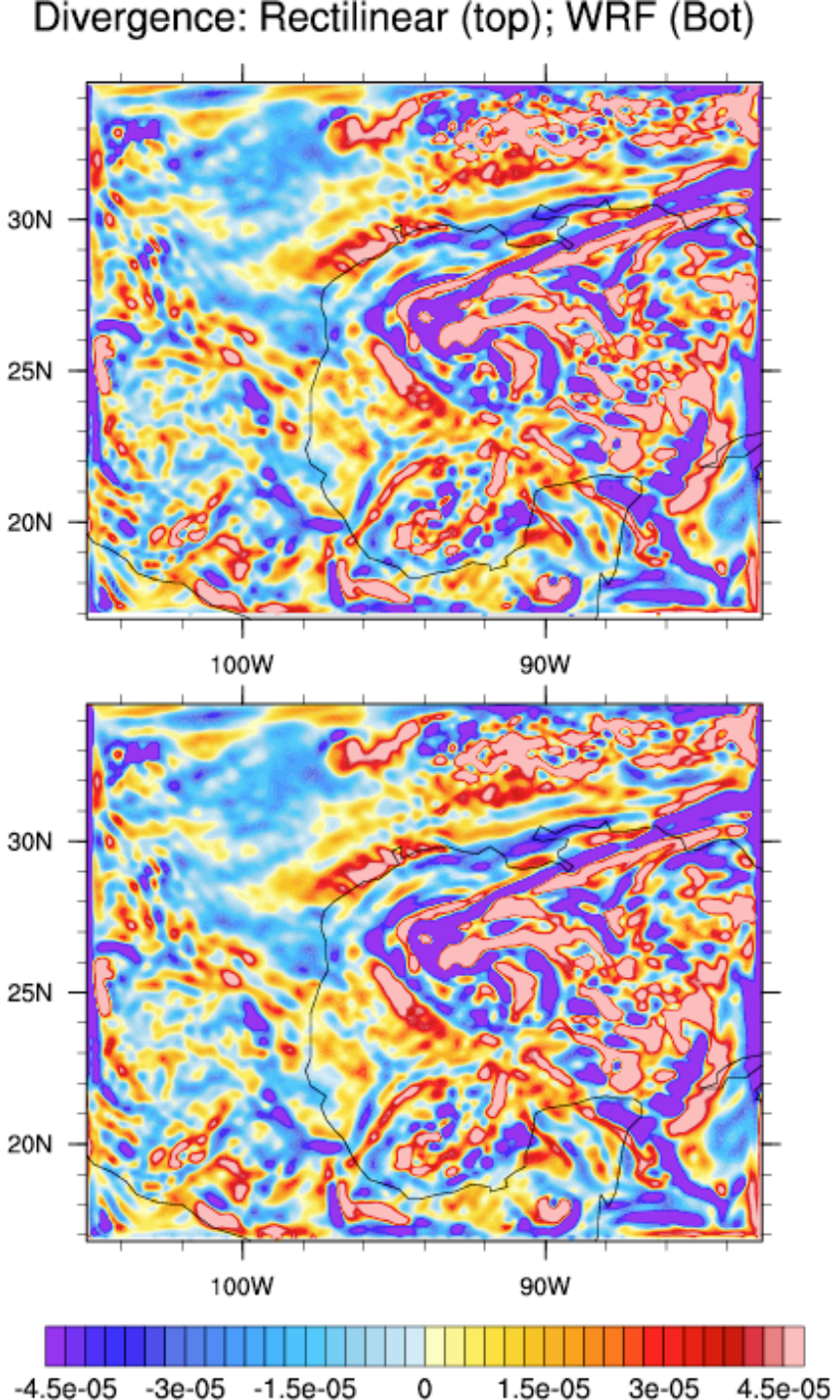

WRF grid (369 x 369)
Temperature (K)
1000 (hPa)
80N
70N
60N
180
150W
Rectilinear grid (369 x 369)
Temperature (K)
1000 (hPa)
80N
70N
60N
180
150W
260
262
264
266
268
270
272
274
276
278
280
282
284
286
288
290

Same plots with the grid included

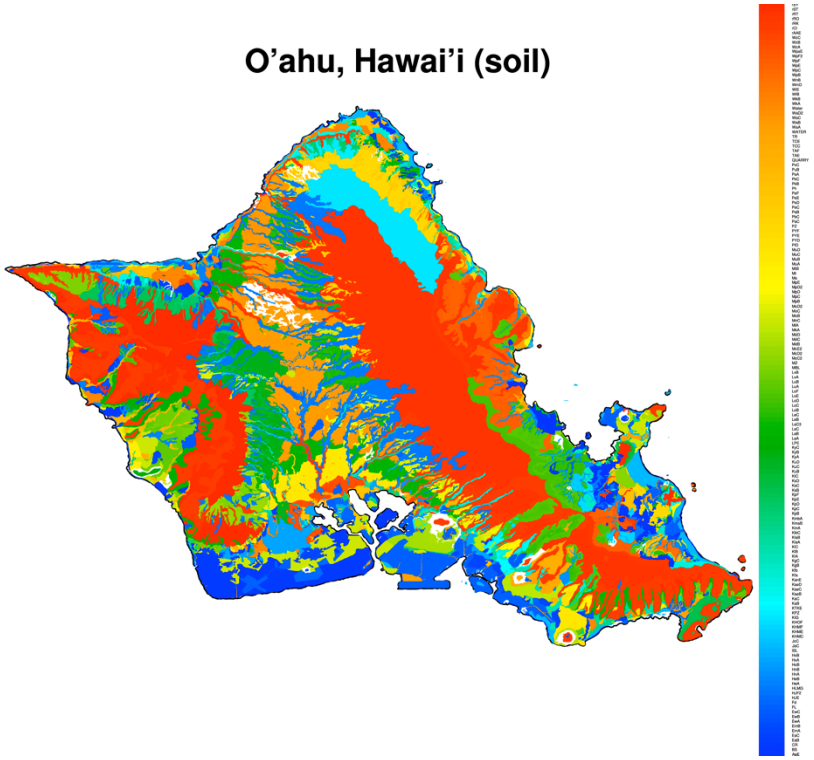


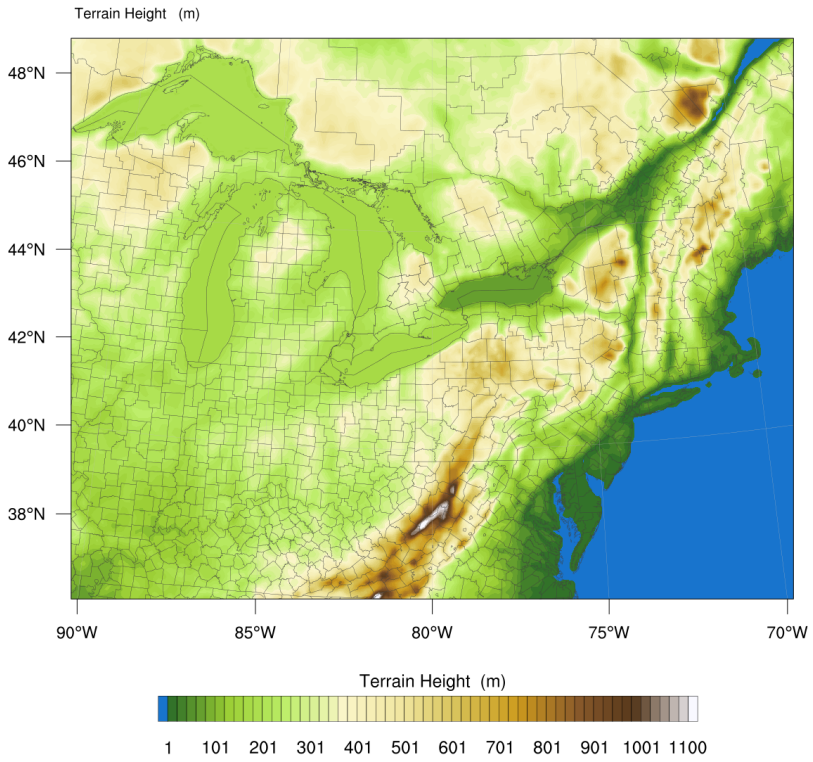


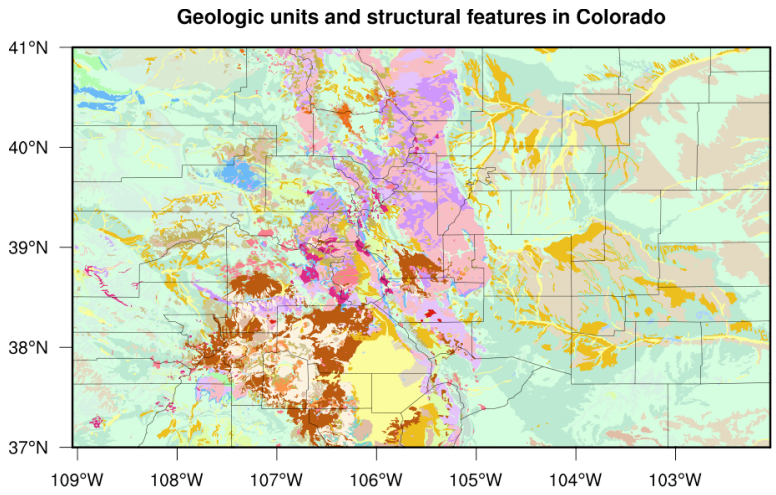
- Interpolate the zonal and meridional wind components using WRF-to-rectilinear weights file
- Compute divergence on the rectilinear grid using uv2dv_cfd
- Interpolate the derived divergence onto the original WRF grid using the rectilinear-to-WRF weights file



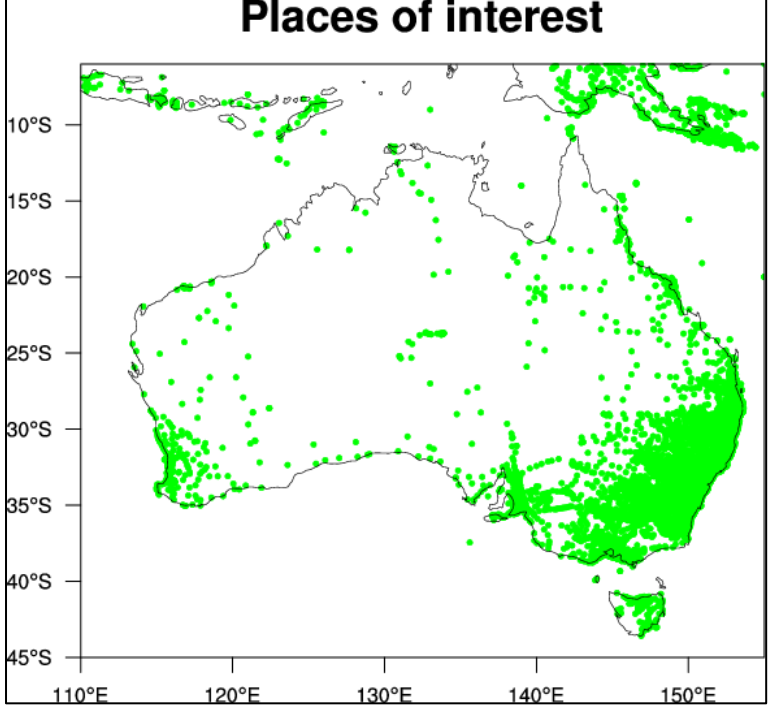
NCL has support for shapefiles, allowing you to use the numerous free shapefiles for adding your own map outlines







OUTPUT FROM WRF V3.3 MODEL
WE = 225 ; SN = 175 ; Levels = 28 ; Dis = 8km ; Phys Opt = 6 ; PBL Opt = 1 ; Cu Opt = 1

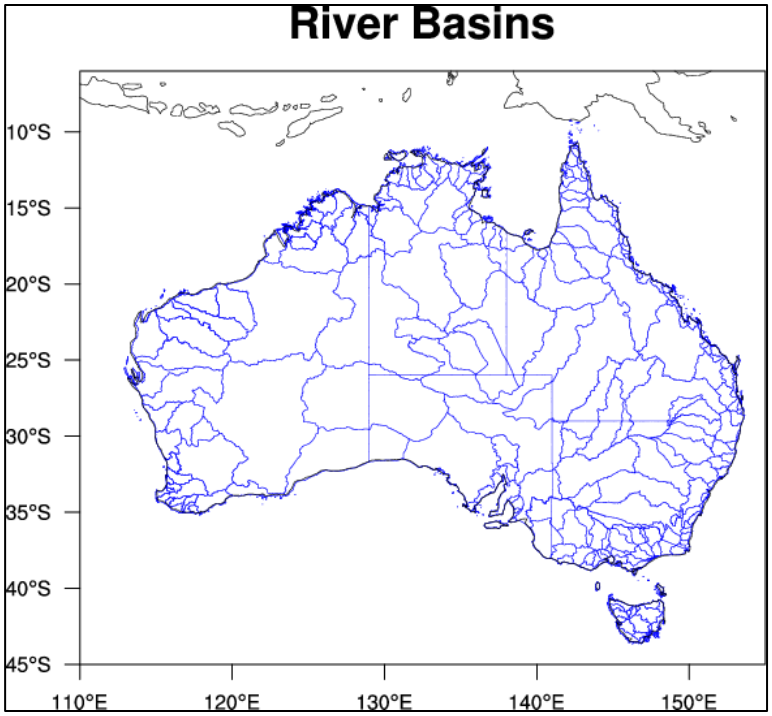


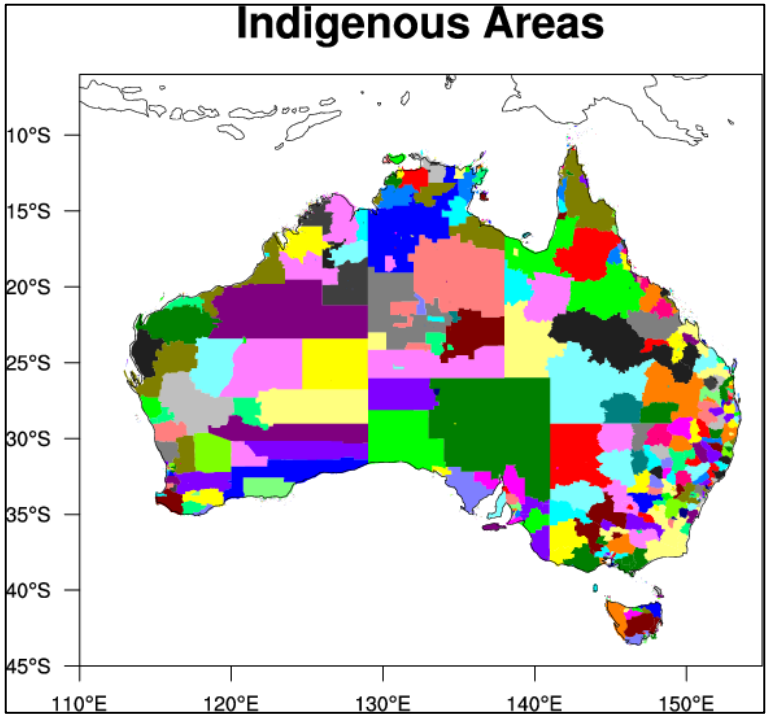
The three types of shapefiles supported by NCL:

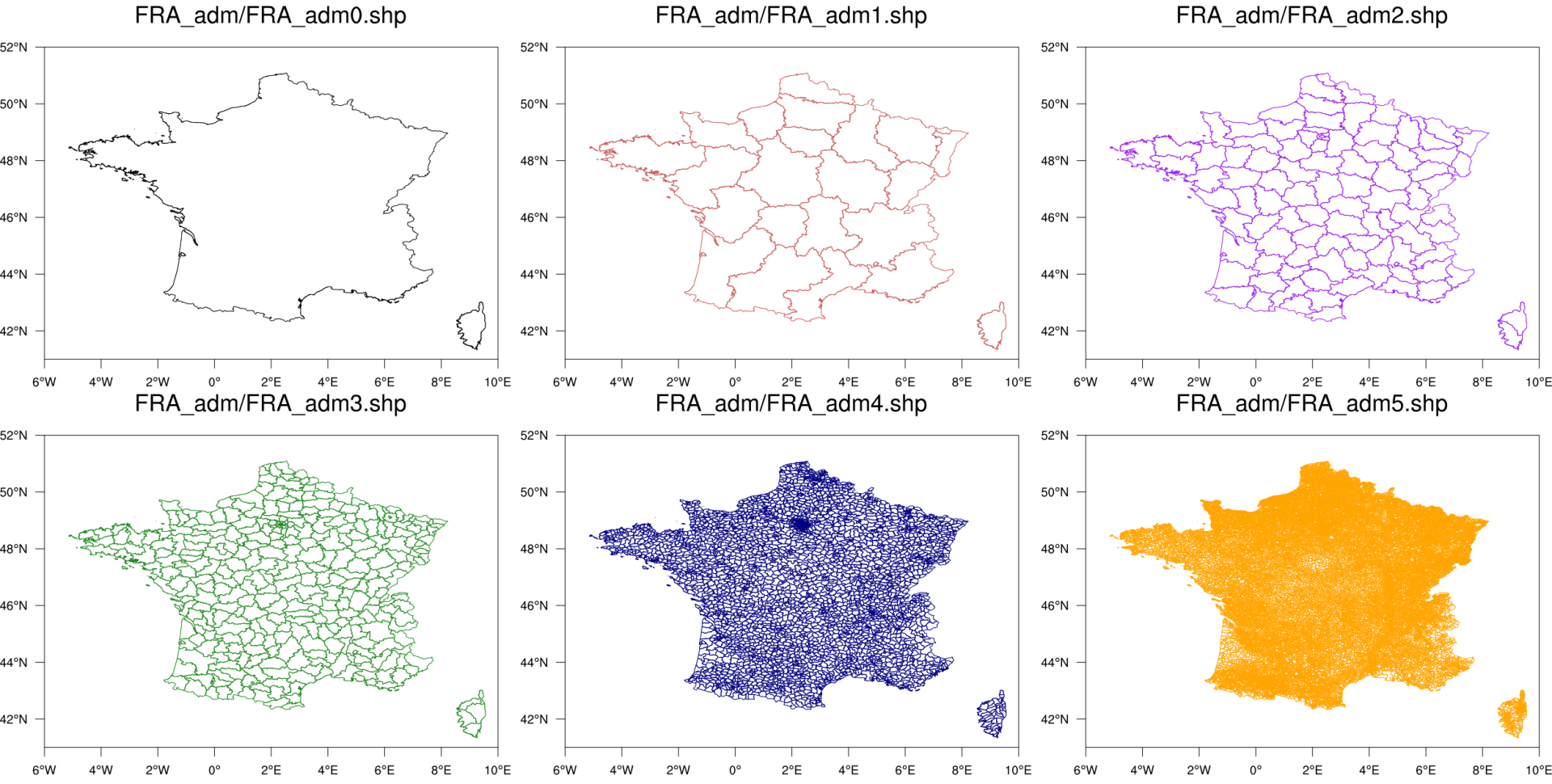
Point – locations of cities, population data, etc

Line – rivers, roads, trails, etc

Polygon – counties, lakes, etc

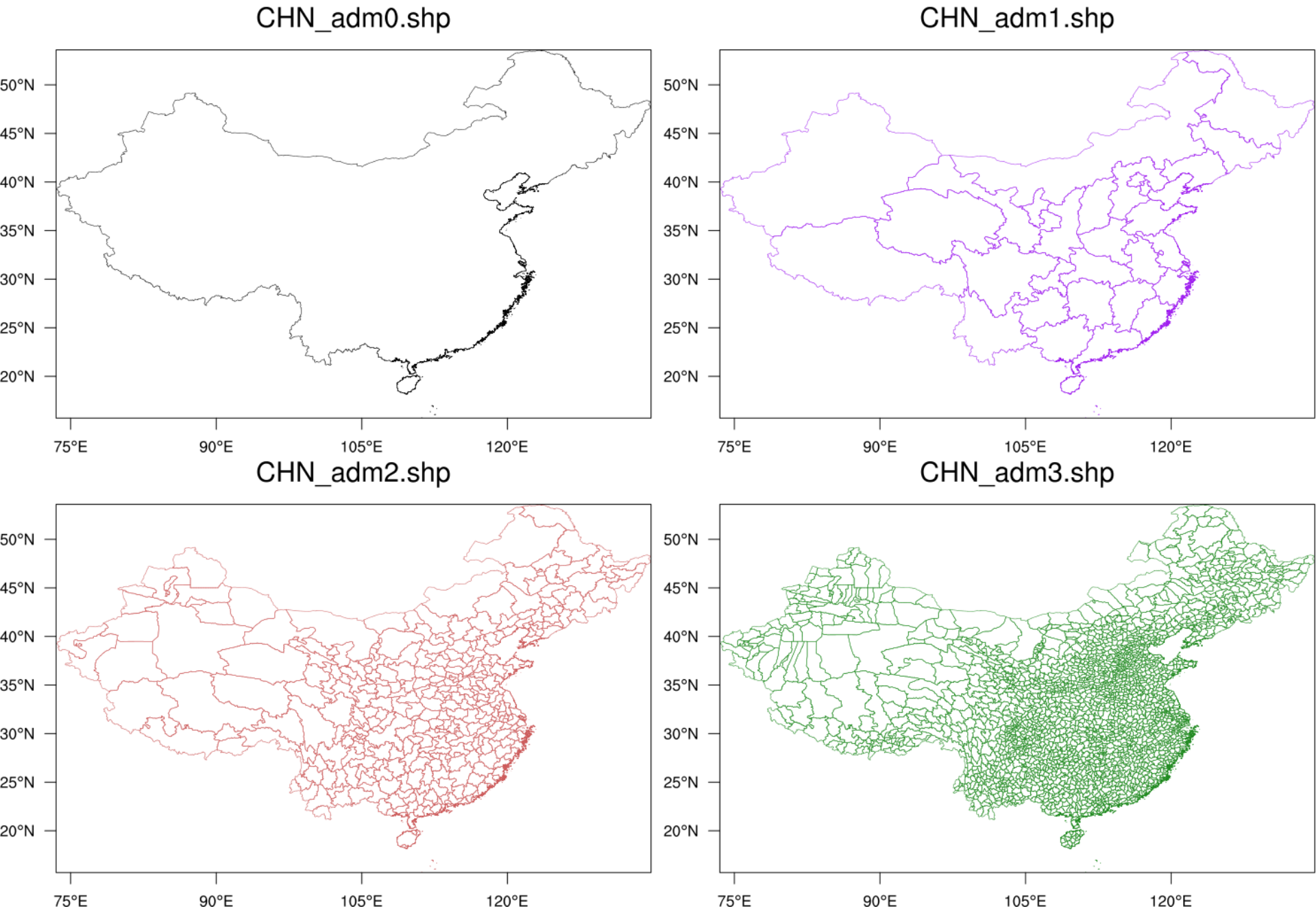






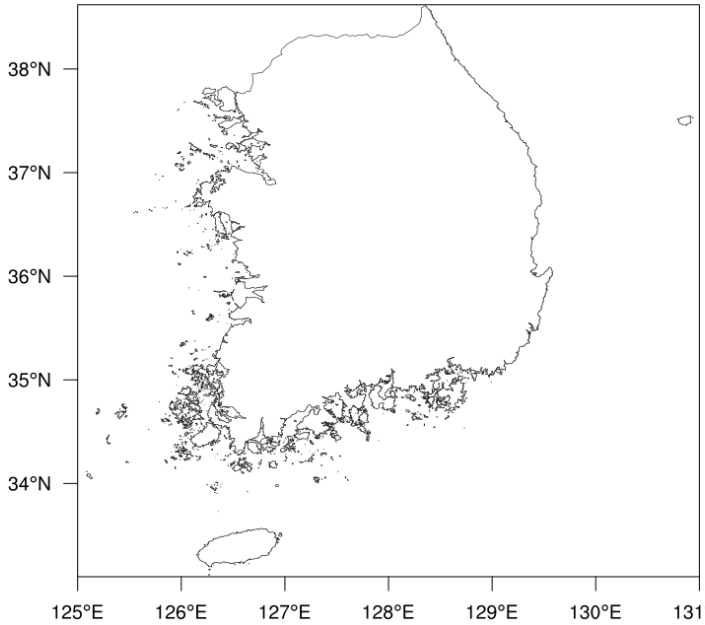
Global Administrative Areas database (**http://www.gadm.org**) offers consistent administrative boundaries at many levels. The level 0 database (nations) is good to use for global or mesoscale results, level 1 is the first level of sub-national administration (typically states/provinces and territories) while level 2 offers the second level of administration and is potentially useful for high-resolution plots.

China shapefiles from gadm.org/country

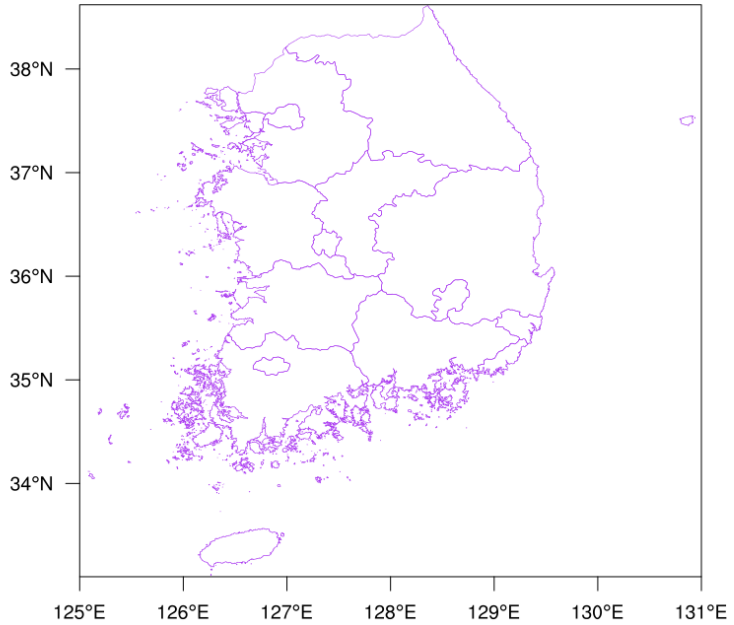


Korea shapefiles from gadm.org/country

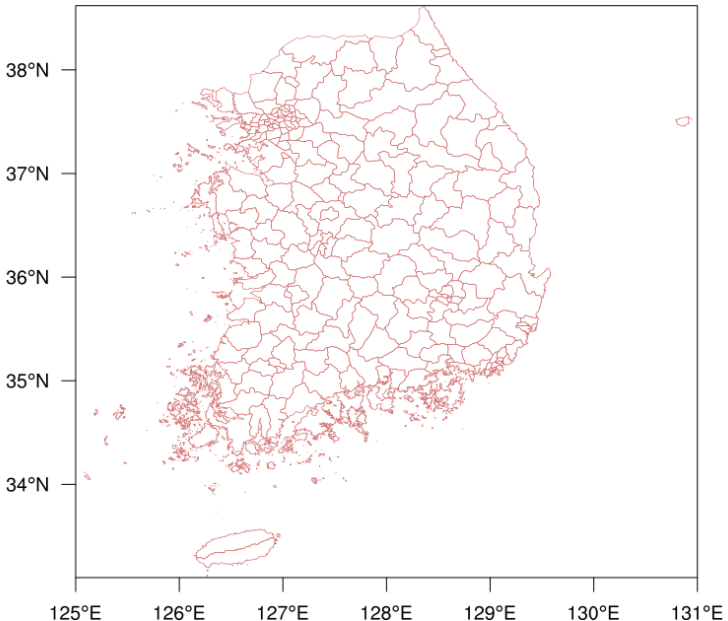
KOR_adm0.shp

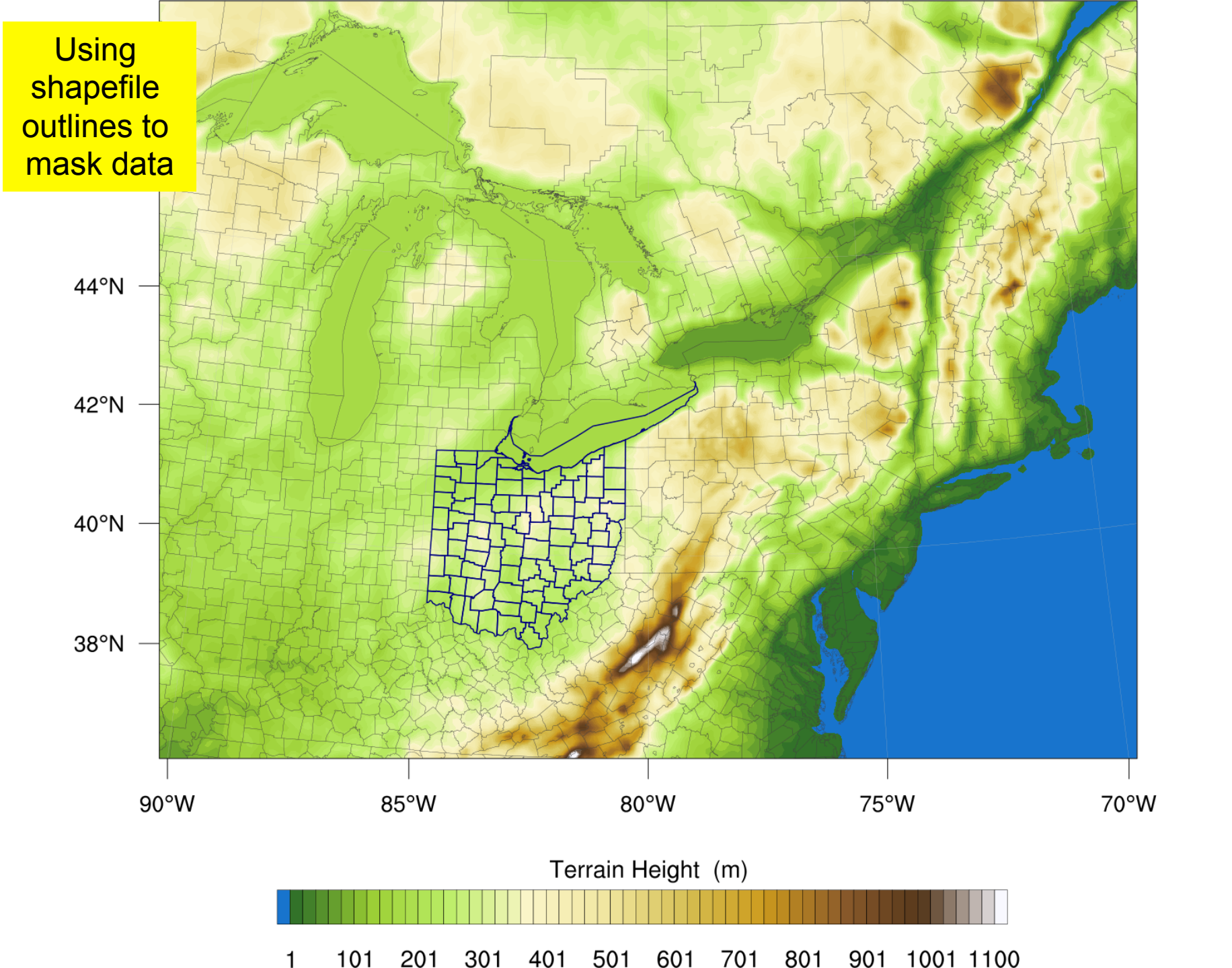


KOR_adm1.shp



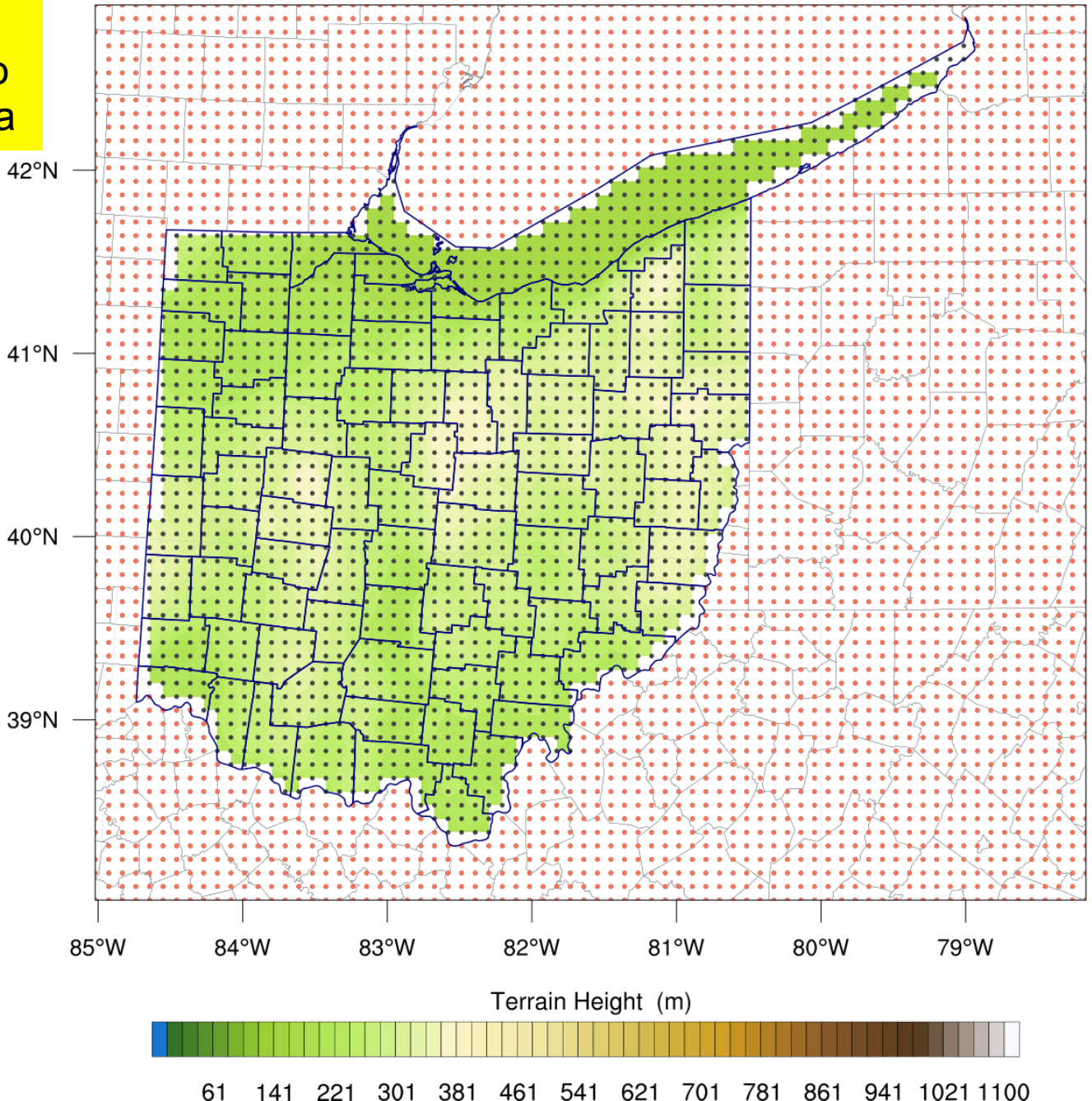
KOR_adm2.shp




Using shapefile outlines to mask data
44°N
42°N
40°N
38°N
90°W
85°W
80°W
75°W
70°W
Terrain Height (m)
1
101
201
301
401
501
601
701
801
901
1001
1100

Using shapefile outlines to mask data

Masking wrfout grid against state of Ohio



Topics

- Overview
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Tips, where to get help
- What's new

NCL language basics

- Running an NCL script
- Overview of an NCL script
- Assigning values and doing simple calculations
- Converting types
- Handling arrays and doing array arithmetic
- Metadata (including missing values)
- Array subscripting

To “run” or “execute” an NCL script:

- *Assuming you have installed NCL…*
- Create an NCL script using an editor (emacs, vi, nedit, etc) that contains NCL script commands, say, “myfile.ncl”.

  Use examples on WRF-ARW online tutorial for help!
  http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/

- Run the file on the command line with:

```
ncl myfile.ncl
```

- Look at resultant output data or view graphical file

Syntax, types, metadata, functions, arrays

A lot of information will follow.

This is mainly to get you familiar with NCL scripts, especially if you are currently having to modify them

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

begin
  print("Hello, world")

; Open a netCDF file and print its contents
  f = addfile("wrfout_d01_2000-01-24_12:00:00.nc","r")
  print(f)

; Read a variable and print its info
  slp = wrf_user_getvar(f,"slp",0)
  printVarSummary(slp)

  wrf_smooth_2d( slp, 3 )                ; Smooth slp

  td2   =  wrf_user_getvar(f,"td2",0)    ; td2 in C
  td_f = 1.8 * td2 + 32.                 ; Convert to F
  td_f@description = "Surface Dew Point Temp"
  td_f@units       = "F"

. .
end
```

Comments begin with ";"
Either on line by itself, or end of line

begin/end are optional

Open the file

This is like doing an "ncdump –h"

Retrieves WRF variable

Use print/printVarSummary for debugging

array arithmetic, like f90

To run this script ("wrf.ncl") on UNIX command line, type:

```
ncl wrf.ncl
```

Scalar variable assignment

```
; Explicit scalar assignment

ndys = 30                          ; integer

x_f  = 2983.599918                 ; float

pi   = 3.14159265358979d           ; double

ll   = 32676l                      ; long

ishort = 10h                       ; short

done = True                        ; logical (False)

long_name = "Water Vapor"          ; string
```

Use "literals" to force type

Mixing types

```
; Mixing types, "largest" type used
i = 7/10            ; integer (i=0)
x = 7/10.           ; float (x=0.7)

y = (22./7)/2d      ; double (1.571428537368774)

z = (i+5) * x       ; float (z=3.5)

; Use "+" for string concatenation
s1 = "hello"
s2 = "world"
s3 = s1 + ", " + s2     ; s3 = "hello, world"

j = 2           ; Can mix strings and numbers
s = "var_" + (j+1) + "_f"     ; s = "var_3_f"
```

Type conversions

```
; Version 6.1.0 and newer: use ':=' to force
ff = 1.5e20    ; float
ff = 1000      ; this is ok, still a float
ff := 1d36     ; now a "double"
ff := (/"one","two","bob"/)  ; array of strings

; Version 6.0.0 and earlier
; Can't change to "higher" type; use delete
ff = 1.5e20    ; float
ff = 1000      ; this is ok, still a float
ff = 1d36      ; not okay, "type mismatch"
delete(ff)
ff = 1d36      ; now this is okay
delete(ff)
ff = (/"one","two","bob"/)
```

Type conversions (cont'd)

```
; Can use conversion functions to
; force "lower" type

dx   = 345.789d                ; double
fx   = tofloat(dx)             ; 345.789
ix   = tointeger(dx)           ; 345
istr = tostring(ix)            ; "345.789000"
```

Arrays

- Row major. . . like C/C++ (*Fortran is column major)*
- Leftmost dimension varies the slowest, rightmost varies fastest (this matters for speed)
- Dimensions are numbered left to right (0,1,…)
- Use "dimsizes" function to get dimension sizes
- Indexes (subscripts) start at 0 (0 to *n*-1)
- Use parentheses to access elements:

```
dx = x(2) – x(1)  ; 3rd value minus 2nd value

; Assume Y is 3D (nx=10,ny=4,nz=2)
y1 = y(0,0,0)     ; first value of a 3D array
yn = y(9,3,1)     ; or y(nx-1,ny-1,nz-1)
                  ; last value of a 3D array
```

Array assignment: (/. . ./)

```
; 1D float array, 3 elements
lat = (/-80,0.,80/)

; string array, 4 elements
MM = (/"March","April","May","June"/)

; string array with appended number ("Mar 01","Apr 01"…)
MMDD = (/"Mar","Apr","May","Jun"/) + " 01"

; 3 x 2 double array
z = ((/1,2d/),(/3,4/),(/9,8/)/)


; Create empty 3D double array, 10 x 64 x 128
x = new((/10,64,128/),double)

; "x" will be filled with default missing value
;       9.96920996838686e+36
```

Array assignment via functions

```
; Generating random numbers
x = random_uniform(-50,1000, (/10,20,30/))

; Do not need "new" first! This is redundant
x = new( (/10,20,30/),float)
x = random_uniform(-50,1000, (/10,20,30/))

; Use "new" only if you have to subscript
x = new( (/10,20,30/),float)

do i=0,9
  x(i,:,:) = some calculation that returns 20x30 array
end do
```

Special functions for arrays

```
; Very useful "where" function
  q = where(z.gt.pi .and. z.lt.pi2, pi*z, 0.5*z)


; "num", "any", "all"

  npos = num (xTemp.gt.0.0) ; Count # values > 0

  if (.not.any(string_array.eq."hello world")) then
    do something
  end if

  if (all(xTemp.lt.0)) then
    do something
  end if

; "ind" function, only on 1D arrays
  ii = ind(pr.lt.500 .and. pr.gt.60)
```

"where" is usually more useful than "ind"

Metadata

- Metadata is information about variables or files.
- In NCL variable model, metadata consists of three things:
  1. Attributes – *describes the file or variable*
     units, history, description, grid type, long name, map projection, missing value
  2. Named dimensions – *describes the dimensions*
     ("time", "north_south", "level", "bottom_top")
  3. Coordinate arrays – *provides coordinate locations of data* (must be one-dimensional)
- Metadata important for calculations, plotting, and general information about data

Metadata (continued)

- WRF-ARW data doesn't have traditional one-dimensional (1D) coordinate arrays.
- WRF lat/lon coordinates are generally 2D or 3D variables on the file and called something like:
  - "XLAT", "XLONG"
  - "XLAT_U", "XLONG_U"
  - "XLAT_V", "XLONG_V"
- WRF variables on "d02" files should have a "coordinates" attribute that tells you which variable on the file represents latitude and longitude

Metadata (continued)

- The "_FillValue" attribute is a special one indicating a variable's missing value.
- "missing_value" attribute not recognized by most of NCL's computational and plotting routines
- Use tools like "ncdump -h" or "ncl_filedump" on a NetCDF file to examine file
  - *Missing values indicated with "-" in ncdump output; NOT the case with "ncl_filedump"*

File and variable attributes

```
; Use the "@" symbol to get at global file attributes.
f = addfile("wrfout_d01_2005-08-27_00:00:00.nc","r")
print(f@TITLE)        ; "OUTPUT FROM WRF V2.1.2 MODEL"
print(f@START_DATE)   ; "2005-08-26_00:00:00"
print(f@MAP_PROJ)     ; 3

; Use the "@" symbol to get at variable attributes too.
uvmet = wrf_user_getvar(f, "uvmet", 0)
print(uvmet@units)          ; "m s-1"
print(uvmet@description)    ; "u,v met velocity"

; Use "isatt" to test for an attribute first.
if(isatt(uvmet,"units")) then
  print("The units of uvmet are '" + uvmet@units + "'")
end if

     The units of uvmet are 'm s-1'
```

Arithmetic operations on arrays, like f90

- May not need to loop over arrays to do calculations
- Arrays need to be same size, but scalars can be used anywhere

```
; Can do arithmetic like Fortran 90
  ch4 = ch4 * 1e6      ; convert to ppm, assign to same var

  A = data_DJF - data_JJA    ; data_DJF/data_JJA must be same size

  zlev = (-7*log(lev/10^3))     ; evaluated as
                                ; (-7)*log(lev/(10^3))
```

Metadata not copied to A or zlev

```
  ; Use "conform" to promote an array
  ; "Twk" is (time,lat,lon,lev), "ptp" is (lat,lon)

    ptropWk = conform(Twk, ptp, (/1,2/)) ; time,lat,lon,lev
```

Array reshape and reverse

```
; Reshaping an array, assume T is 10 x 20 x 30

  t1D = ndtooned(T)                    ; Convert to 1D array
  t2D = onedtond(t1D, (/200,30/))      ; Convert to 200 x 30

  t2D = reshape(T,(/200,30/))          ; Added in V6.1.0


; Reversing dimensions of an array

; Let W(Time, bottom_top_stag, south_north, west_east)
  W = W(:,::-1,:,:)    ; Reverses the level dimension
```

Array Subscripting

- Three kinds of array subscripting
  1. Index (uses ':' and '::')
  2. Coordinate (uses curly braces '{' and '}')
  3. Named dimensions (uses '!')
- Most (all?) WRF-ARW data does not have coordinate arrays, so can't use method #2
- You can mix subscripting types in one variable
- Be aware of dimension reduction
- Index subscripting is 0-based
  (Fortran is 1-based, by default)

Array index subscripting, : and ::

```
; Consider T(ntime x nlat x nlon)
  t = T                  ; copies metadata, don't use T(:,:,:)
  t = (/T/)              ; doesn't copy metadata
                         ; (_FillValue is retained)

; The following creates 2D array "t"
  t = T(0,:,::5)         ; 1st time index, all lat, every 5th lon
                         ; (nlat x nlon/5)

  t = T(0,::-1,:50)      ; 1st time index, reverse lat,
                         ; first 51 lons (nlat x 51)

  t = T(:1,45,10:20)     ; 1st two time indices, 46th index of lat,
                         ; 11th-21st indices of lon (2 x 11)

; To prevent dimension reduction
  t = T(0,:,::5)         ; nlat x nlon/5
  t = T(0:0,:,::5)       ; 1 x nlat x nlon/5
```

Topics

- Overview
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Tips, where to get help
- What's new

Opening and examining a WRF output file

```
f = addfile("wrfout_d01_2005-08-27_00:00:00.nc","r")
print(f)
```

WRF files don't have ".nc" suffix; must add here.

print(f) results

```
Variable: f (file variable)

filename:       wrfout_d01_2005-08-27_00:00:00
path:   wrfout_d01_2005-08-27_00:00:00
   file global attributes:
      TITLE :  OUTPUT FROM WRF V2.1.2 MODEL
      START_DATE : 2005-08-26_00:00:00
      SIMULATION_START_DATE : 2005-08-26_00:00:00
      WEST-EAST_GRID_DIMENSION : 400
      SOUTH-NORTH_GRID_DIMENSION : 301
      BOTTOM-TOP_GRID_DIMENSION : 35
      DX : 12000
      DY : 12000
      GRIDTYPE : C
      DYN_OPT : 2
      DIFF_OPT : 1 KM_OPT : 4
      DAMP_OPT : 0
```

global attributes

print(f) results (continued)

more global attrs

```
KHDIF :  0
KVDIF :  0
MP_PHYSICS : 3
RA_LW_PHYSICS : 1
RA_SW_PHYSICS : 1
SF_SFCLAY_PHYSICS : 1
SF_SURFACE_PHYSICS : 1
BL_PBL_PHYSICS : 1
CU_PHYSICS : 1
WEST-EAST_PATCH_START_UNSTAG : 1
WEST-EAST_PATCH_END_UNSTAG : 399
WEST-EAST_PATCH_START_STAG : 1
WEST-EAST_PATCH_END_STAG : 400
SOUTH-NORTH_PATCH_START_UNSTAG : 1
SOUTH-NORTH_PATCH_END_UNSTAG : 300
SOUTH-NORTH_PATCH_START_STAG : 1
SOUTH-NORTH_PATCH_END_STAG : 301
BOTTOM-TOP_PATCH_START_UNSTAG : 1
BOTTOM-TOP_PATCH_END_UNSTAG : 34
BOTTOM-TOP_PATCH_START_STAG : 1
BOTTOM-TOP_PATCH_END_STAG : 35
GRID_ID : 1
PARENT_ID : 0
I_PARENT_START : 0
J_PARENT_START : 0
PARENT_GRID_RATIO : 1
DT : 60
```

print(f) results (continued)

variable dimension names

variables

```
. . .
dimensions:
    Time = 1  // unlimited
    DateStrLen = 19
    west_east = 399
    south_north = 300
    west_east_stag = 400
    bottom_top = 34
    south_north_stag = 301
    bottom_top_stag = 35
    ext_scalar = 1
    soil_layers_stag = 5
variables:
    character Times ( Time, DateStrLen )
    float LU_INDEX ( Time, south_north, west_east )
        FieldType :     104
        MemoryOrder :   XY
        description :   LAND USE CATEGORY
        units :
        stagger :
    float U ( Time, bottom_top, south_north, west_east_stag )
        FieldType :     104
        MemoryOrder :   XYZ
        description :   x-wind component
        units :         m s-1
        stagger :       X
```

```
float V ( Time, bottom_top, south_north_stag, west_east )
   FieldType :     104
   MemoryOrder :   XYZ
   description :   y-wind component
   units :         m s-1
   stagger :       Y

float W ( Time, bottom_top_stag, south_north, west_east )
   FieldType :     104
   MemoryOrder :   XYZ
   description :   z-wind component
   units :         m s-1
   stagger :       Z

float PH ( Time, bottom_top_stag, south_north, west_east )
   FieldType :     104
   MemoryOrder :   XYZ
   description :   perturbation geopotential
   units :         m2 s-2
   stagger :       Z

float PHB ( Time, bottom_top_stag, south_north, west_east )
   FieldType :     104
   MemoryOrder :   XYZ
   description :   base-state geopotential
   units :         m2 s-2
   stagger :       Z
```

print(f) results (continued)

more variables

Example of wrfout_d03 file

filename: wrfout_d03_2012-04-22_23_00_00
file global attributes:
TITLE : OUTPUT FROM WRF V3.3.1 MODEL
START_DATE : 2012-04-20_00:00:00
SIMULATION_START_DATE : 2012-04-20_00:00:00
...
dimensions:
Time = 1 // unlimited
south_north = 546
bottom_top = 31
...
variables:
...
float MAPFAC_UY (Time, south_north, west_east_stag)
FieldType : 104
MemoryOrder : XY
stagger :X
coordinates : XLONG_U XLAT_U

float XLAT_U (Time, south_north, west_east_stag)
FieldType : 104
MemoryOrder : XY
description : LATITUDE, SOUTH IS NEGATIVE
units : degree_north
coordinates : XLONG_U XLAT_U

float F (Time, south_north, west_east)
FieldType : 104
MemoryOrder : XY
description : Coriolis sine latitude term
units : s-1
stagger :
coordinates : XLONG XLAT

float XLAT (Time, south_north, west_east)
FieldType : 104
MemoryOrder : XY
description : LATITUDE, SOUTH IS NEGATIVE
units : degree_north
coordinates : XLONG XLAT

Using “ncl_filedump” on UNIX command line

Don’t need to write a script to quickly look at a WRF file. On the UNIX command line, type:

```
ncl_filedump –h
ncl_filedump wrfout_d01_2005-08-27_00:00:00.nc | less
ncl_filedump –v RAINC wrfout_d01_2005-08-27_00:00:00.nc
```

Can use ncl_filedump on other files that NCL’s “addfile” supports: GRIB 1 and 2, HDF4, HDF-EOS2, etc

```
ncl_filedump TES-Aura_L3-ATM-TEMP_r0000003459_F01_05.he5
ncl_filedump z_tigge_c_rjtd_20061119120000_0072_sl_glob_prod.grb2
ncl_filedump states.shp
```

Two ways to read a variable off a file

1. Use "->" syntax to directly read variables

2. Use "wrf_user_getvar" function
   - Developed to make it easier to get derived variables
   - Must load "WRFUserARW.ncl" script
   - You can modify this script (more later)
   - Can only use with WRF-ARW data

Reading (and examining) a variable off a file (method 1)

```
f = addfile("wrfout_d01_2005-08-27_00:00:00.nc","r")
u = f->U
printVarSummary(u)
; print(u)          ; Same as printVarSummary, but includes values
```

printVarSummary(u) results

named dimensions

no coordinate arrays

variable attributes

```
Variable: u
Type: float
Total Size: 16320000 bytes
            4080000 values
Number of Dimensions: 4
Dimensions and sizes:   [Time | 1] x [bottom_top | 34] x [south_north
| 300] x [west_east_stag | 400]
Coordinates:
Number Of Attributes: 5
  FieldType :   104
  MemoryOrder : XYZ
  description : x-wind component
  units :       m s-1
  stagger :     X
```

Reading (and examining) a variable off a file (method 2)

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

f = addfile("wrfout_d01_2005-08-27_00:00:00.nc","r")
slp = wrf_user_getvar(f,"slp",0)
printVarSummary(slp)
```

printVarSummary(slp) results

```
Variable: slp
Type: float
Total Size: 478800 bytes
            119700 values
Number of Dimensions: 2
Dimensions and sizes:   [south_north | 300] x [west_east | 399]
Coordinates:
Number Of Attributes: 5
  description : Sea Level Pressure
  units :       hPa
  FieldType :   104
  MemoryOrder : XYZ
  stagger :
```

Further querying a variable

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

f = addfile("wrfout_d01_2005-08-27_00:00:00.nc","r")
slp = wrf_user_getvar(f,"slp",0)

print(dimsizes(slp))    ; Print dimension sizes of slp
print(min(slp))         ; Print minimum of slp
print(typeof(slp))      ; Print type of slp
print(getvaratts(slp)) ; Print attributes of slp
```

```
300
399
973.2794
float
description
units
FieldType
MemoryOrder
stagger
```

Demo time?

Topics

- Overview
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Tips, where to get help
- What's new

WRF-NCL Functions

- Two kinds:
  - ***Built-in*** - mainly functions to calculate diagnostics. *Seldom need to use these directly.*

```
slp = wrf_slp(z, tk, P, QVAPOR)
```

  - ***"WRFUserARW.ncl"*** - developed to make it easier to calculate derived variables and generate plots, calls some built-in functions

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
slp = wrf_user_getvar(f,"slp",time)  ; internally calls
                                     ; wrf_slp
```

WRF-NCL built-in functions

Can use NCL built-in functions, in place of wrf_user_getvar, not always recommended!

```
T       = f->T(time,:,:,:)
P       = f->P(time,:,:,:)
PB      = f->PB(time,:,:,:)
QVAPOR  = f->QVAPOR(time,:,:,:)
PH      = f->PH(time,:,:,:)
PHB     = f->PHB(time,:,:,:)
T = T + 300.
P = P + PB
QVAPOR = QVAPOR > 0.0 ; Set anything <= 0 to msg
PH     = ( PH + PHB ) / 9.81

z    = wrf_user_unstagger(PH,PH@stagger)
tk   = wrf_tk( P , T )
slp = wrf_slp( z, tk, P, QVAPOR )
```

Replace with single call

```
slp = wrf_user_getvar(f,"slp",time)
```

WRF-NCL "*WRFUserARW.ncl*" functions

wrf_user_getvar - Get fields from input file

```
ter = wrf_user_getvar(a,"HGT",0)
t2  = wrf_user_getvar(a,"T2",-1)
slp = wrf_user_getvar(a,"slp",1)
```

wrf_user_getvar is user-modifiable! (more later)

Diagnostics

avo/pvo	Absolute/Potential Vorticity
cape_2d	2D mcape/mcin/lcl/lfc
cape_3d	3D cape/cin
dbz/mdbz	Reflectivity
. . .	
omg	**Omega [C]**
slp	Sea level pressure
. . .	
tv	***Virtual temperature [K]***
tw	***Wet bulb temperature [C]***
updraft_helicity	Updraft helicity [m-2/s-2]
ua/va/wa	Wind on mass points
uvmet/uvmet10	U/V components of wind rotated to earth coords
z/height	Height

New in NCL V6.2.0

Other WRF-NCL "*WRFUserARW.ncl*" functions

- wrf_user_list_times
  Get list of times available in input file

```
times = wrf_user_list_times (f)
```

- wrf_user_unstagger
  Unstaggers an array

```
ua = wrf_user_unstagger (U, "X")
ua = wrf_user_getvar(f,"ua",time)
```

- wrf_map_overlays
  Draws plots over a map background

```
map = wrf_map_overlays(a, wks, \
   (/contour,vector/), pltres, mpres)
```

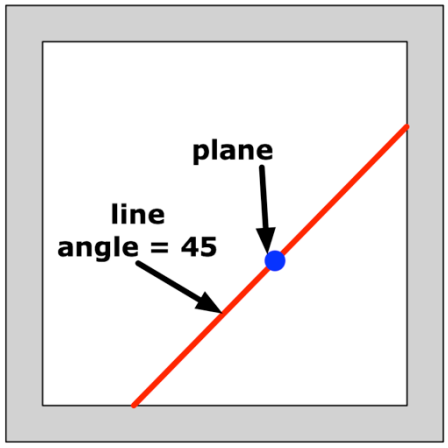
Other WRF-NCL "*WRFUserARW.ncl*" functions

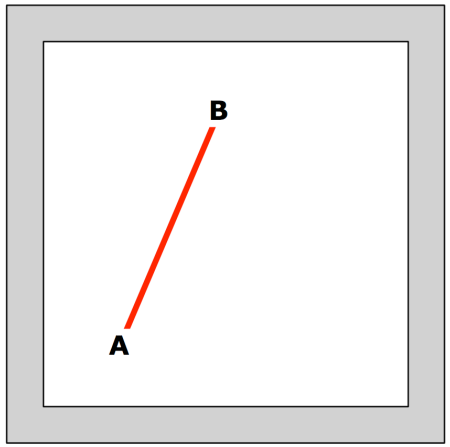
- wrf_user_intrp3d
  Interpolate horizontally to a given pressure or height level
  Interpolate vertically *(pressure/height)*, along a given line

```
tc_plane = wrf_user_intrp3d( tc, p, "v", (/30,25/), \
                             45., False )
```

- wrf_user_intrp2d
  Interpolate along a given line

```
t2_plane = wrf_user_intrp2d(t2, (/12,10,25,45/), \
                            0., True)
```





Other WRF-NCL "*WRFUserARW.ncl*" functions

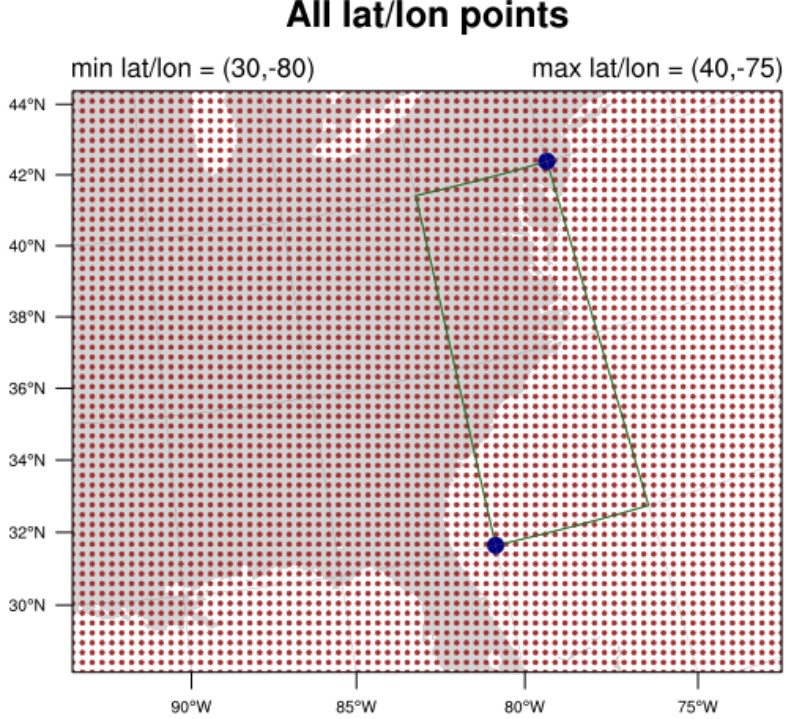
- wrf_user_ll_to_ij / wrf_user_ij_to_ll

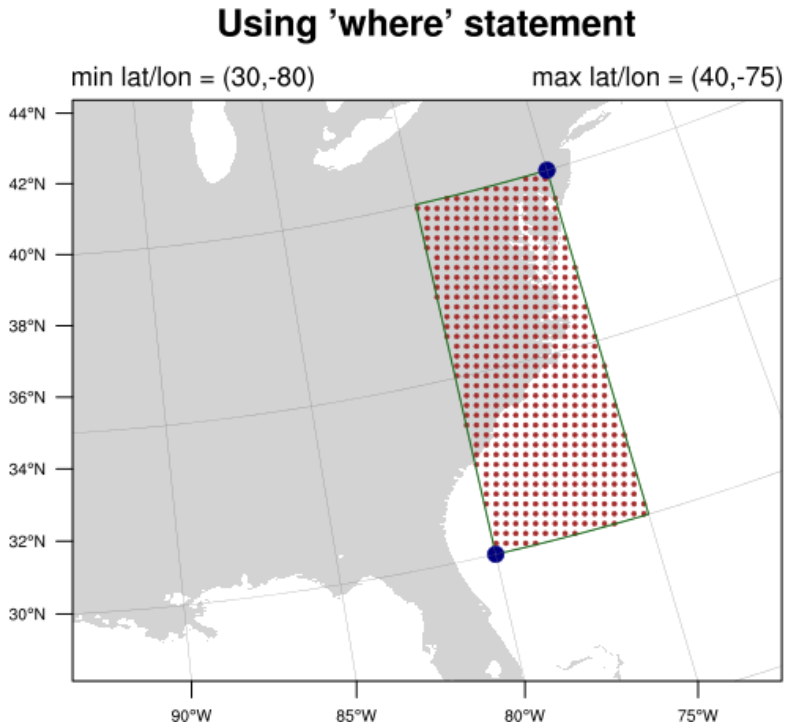
  Convert: lat/lon values to i/j index locations

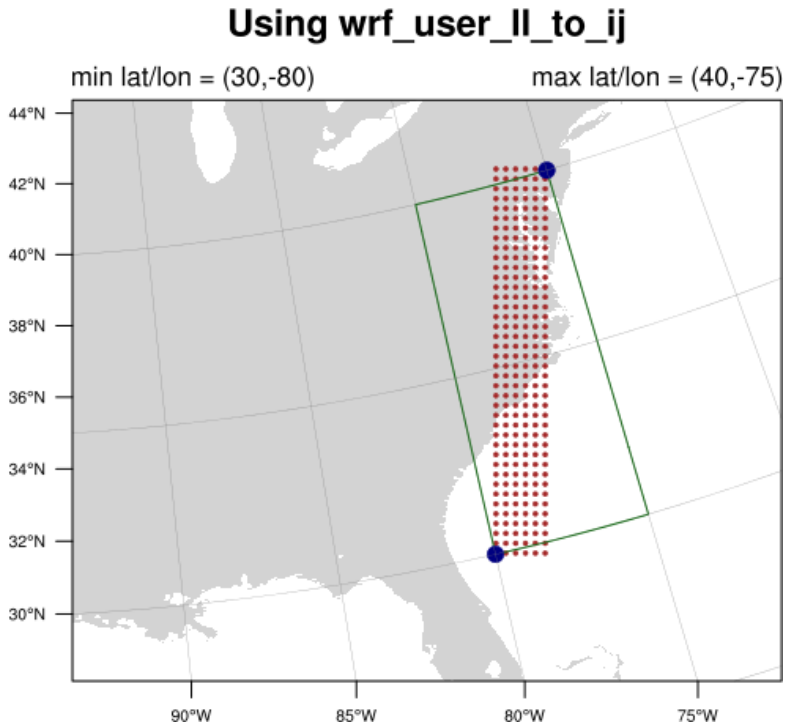
```
locij = wrf_user_ll_to_ij (f, 100., 40., res)
locll = wrf_user_ij_to_ll (f, (/10, 12/), \
                              (/40, 50/), res)
```

- *res@useTime* - Default is 0
  Set to a time index value if you want the reference longitude/latitudes to come from a different time index - only use this for moving nest output which has been stored in a single file.

Using wrf_user_ll_to_ij and "where" statement to sub-select area







Modifying wrf_user_getvar function

- Copy the following file to your own directory:
  "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

- Edit your copy and look for line that starts with:

```
function wrf_user_getvar
```

- Before the lines:

```
  return(var)
end
```

Add these lines, replacing "*newvar*" as appropriate:

```
if( variable .eq. "newvar" ) then
  . . .fill in code here. . .
  return(newvar)
end if
```

Modifying wrf_user_getvar function (cont'd)

- To use the new version of this function, you can do one of two things:

1. Load your modified script instead of the system one:

```
load "./WRFUserARW.ncl"
xxx = wrf_user_getvar(f,"XXX",0)
```

2. Remove all but the modified "wrf_user_getvar" function from your copy, rename the function ("wrf_user_getvar2"), and rename the file ("my_new_script.ncl"). To use the new function, you need to load the above script and your new script:

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
load "./my_new_script.ncl"

xxx = wrf_user_getvar2(f,"XXX",0)
```

Topics

- Overview
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Tips, where to get help
- What's new

Links for visualization scripts

- WRF-ARW online tutorial
  http://www.mmm.ucar.edu/wrf/OnLineTutorial/index.htm

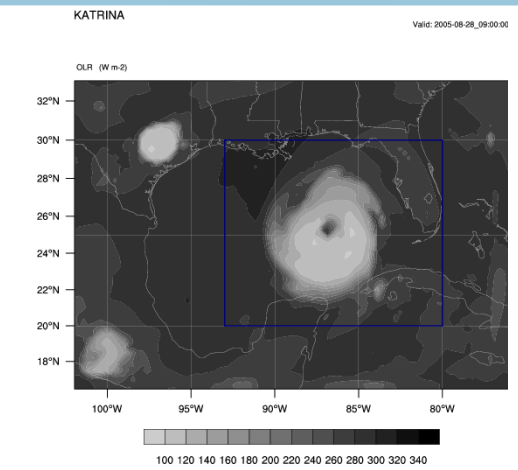
- NCL/WRF examples page
  http://www.ncl.ucar.edu/Applications/wrf.shtml

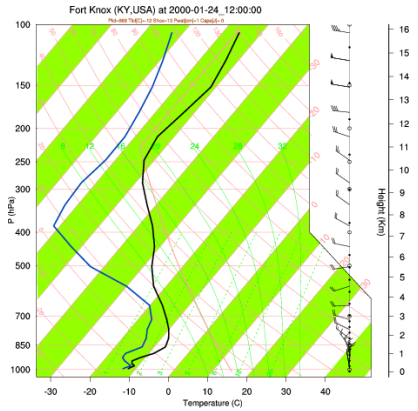
  NCL Home Page -> Examples -> WRF

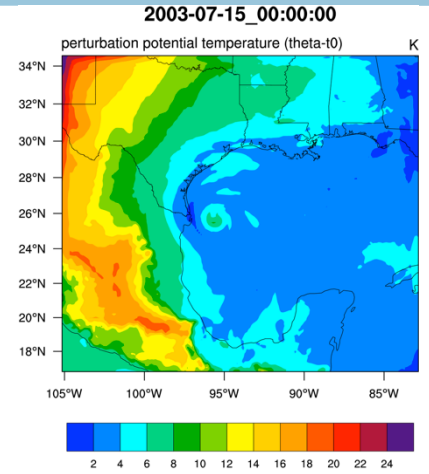
- Description of WRF-NCL functions
  http://www.ncl.ucar.edu/Document/Functions/wrf.shtml

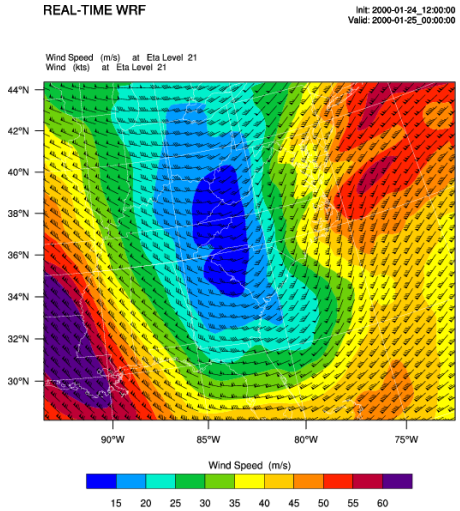
  NCL Home Page -> Functions -> Category -> WRF

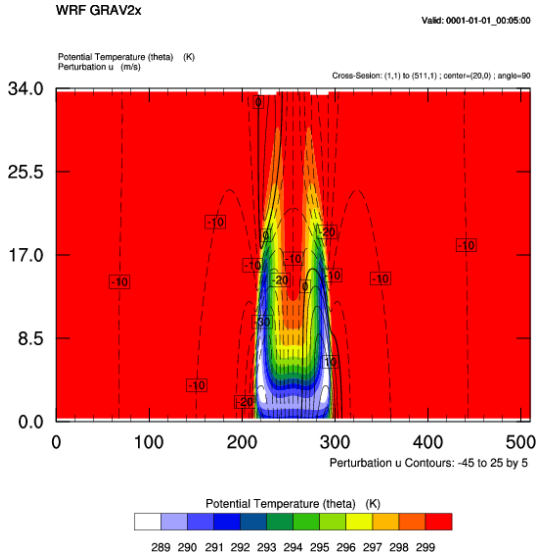
Step-by-step WRF-ARW visualizations

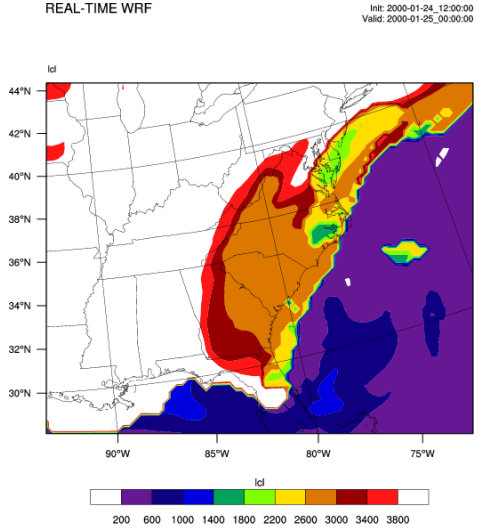












OUTPUT FROM WRF V3.0.1.1 MODEL
WE = 74 ; SN = 61 ; Levels = 28 ; Dis = 30km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

OUTPUT FROM WRF V3.0.1.1 MODEL
WE = 74 ; SN = 61 ; Levels = 28 ; Dis = 30km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/

Step-by-step: filled contours using wrf_xxxx

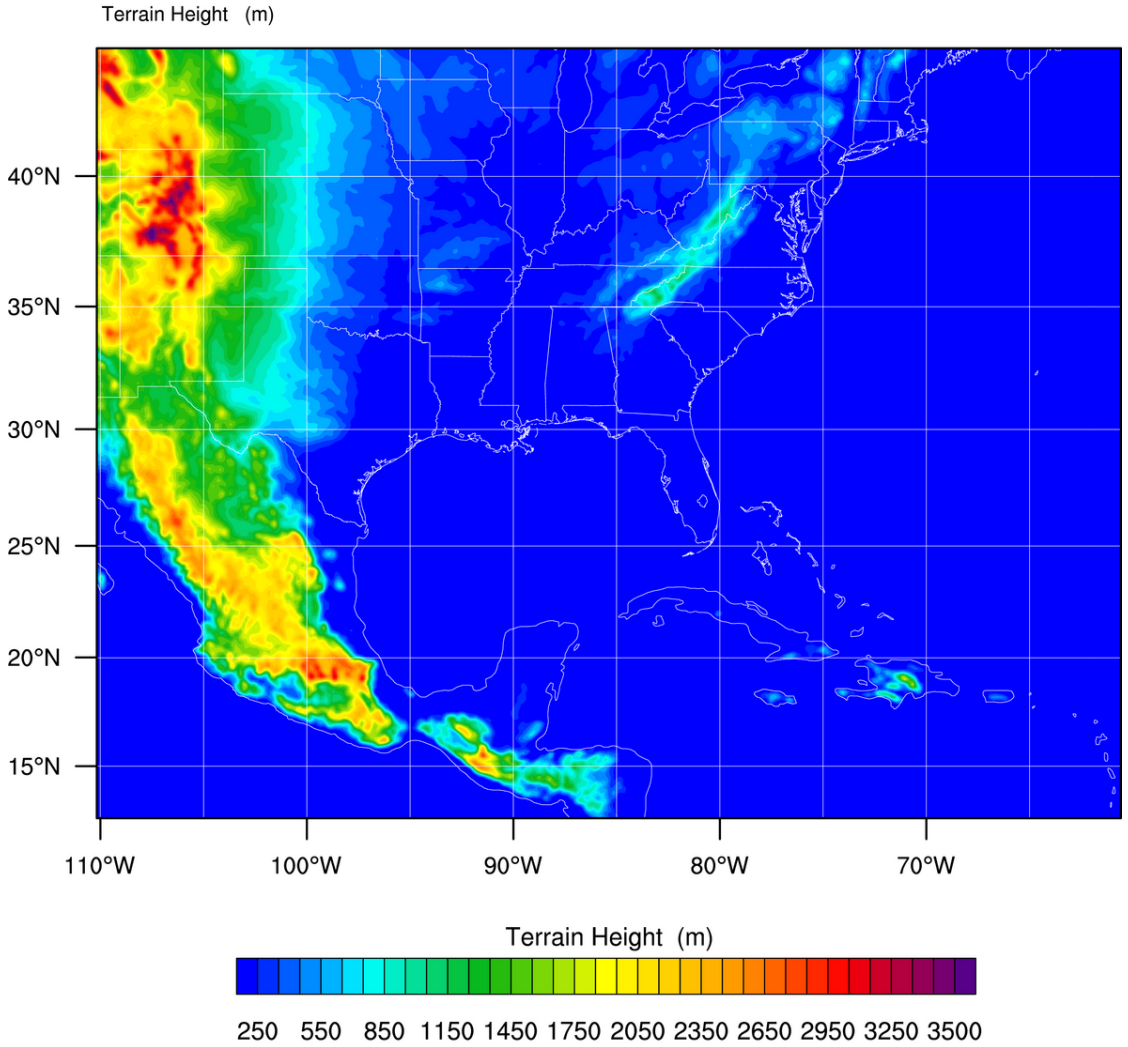
```
; Load the necessary scripts
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

; Open a file and read a variable
f = addfile("wrfout_d01_2005-08-27_00:00:00.nc","r")
hgt = wrf_user_getvar(f,"HGT",0)

wks = gsn_open_wks("png","hgt")  ; "hgt.png"

; Set some plotting resources
res = True
res@cnFillOn = True

; These are special wrf_xxxx resources
res@MainTitle          = "GEOGRID FIELDS"
res@ContourParameters = (/ 250., 3500., 100. /)
contour = wrf_contour(f,wks,hgt,res)

pltres = True
mpres  = True
plot   = wrf_map_overlays(f,wks,contour,pltres,mpres)
```

These are plot options, also known as "resources"

wrf_map_overlays looks at file to determine map projection

GEOGRID FIELDS

Init: 2005-08-26_00:00:00



OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

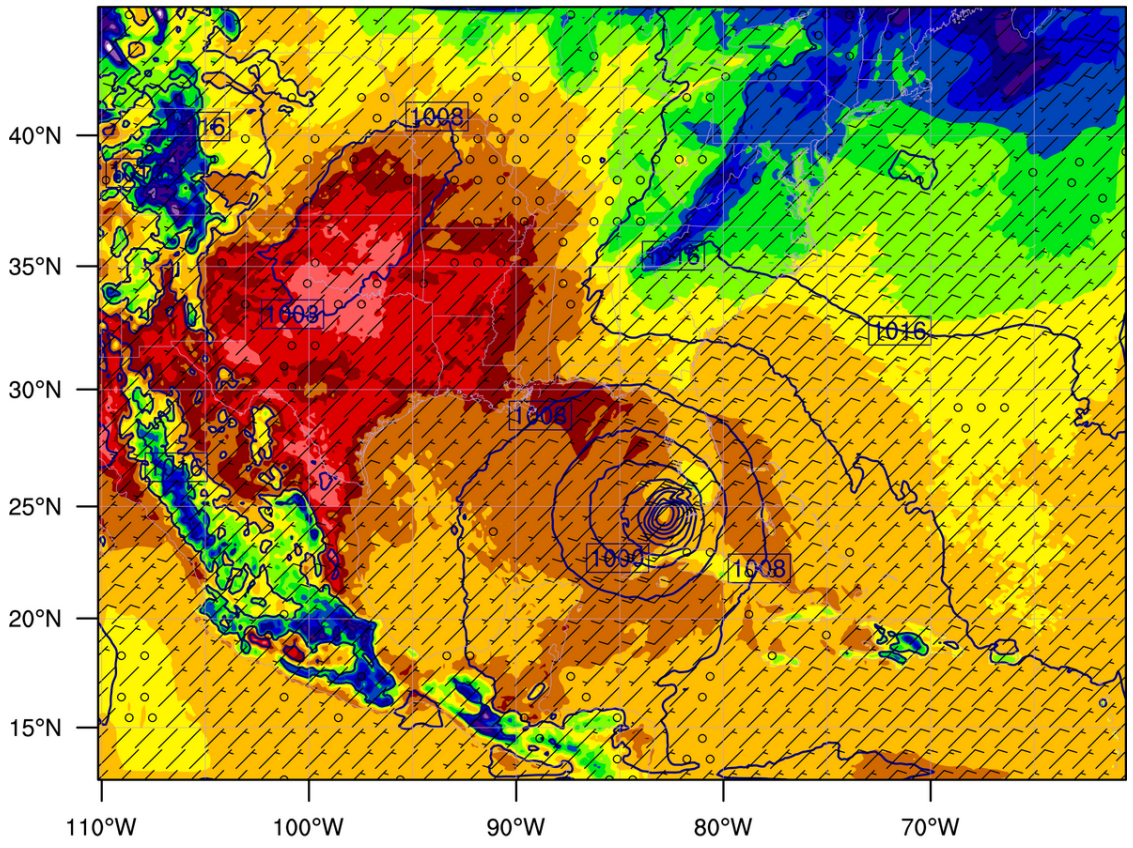
Step-by-step: line/fill contours, vectors

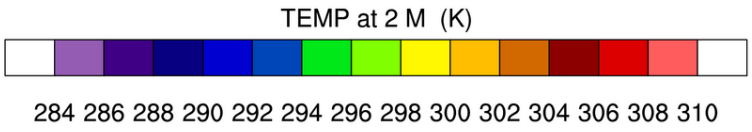
```
; Load the necessary scripts
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

; Open a file and get several diagnostics
f   = addfile("wrfout_d01_2005-08-27_00:00:00.nc","r")

slp = wrf_user_getvar(f,"slp",0)
t2  = wrf_user_getvar(f, "T2",0)
u10 = wrf_user_getvar(f, "U10",0)
v10 = wrf_user_getvar(f, "V10",0)
```

```
wks = gsn_open_wks("ps","wrf")    ; "wrf.ps" file for output

; Line contours
os                    = True
os@cnLineColor        = "NavyBlue"
os@cnLineThicknessF   = 2.0
c_slp                 = wrf_contour(f,wks,slp,os)

; Filled contours
ot                = True
ot@cnFillOn       = True
c_tc              = wrf_contour(f,wks,t2,ot)

; Vectors
ov                = True
ov@NumVectors     = 47
vec               = wrf_vector(f,wks,u10,v10,ov)

; Overlay everything on a map
mpres  = True
pltres = True
plot = wrf_map_overlays(f,wks,(/c_tc,c_slp,vec/),pltres, mpres)
```

Init: 2005-08-26_00:00:00

TEMP at 2 M (K)
Sea Level Pressure (hPa)
U at 10 M (m s-1)





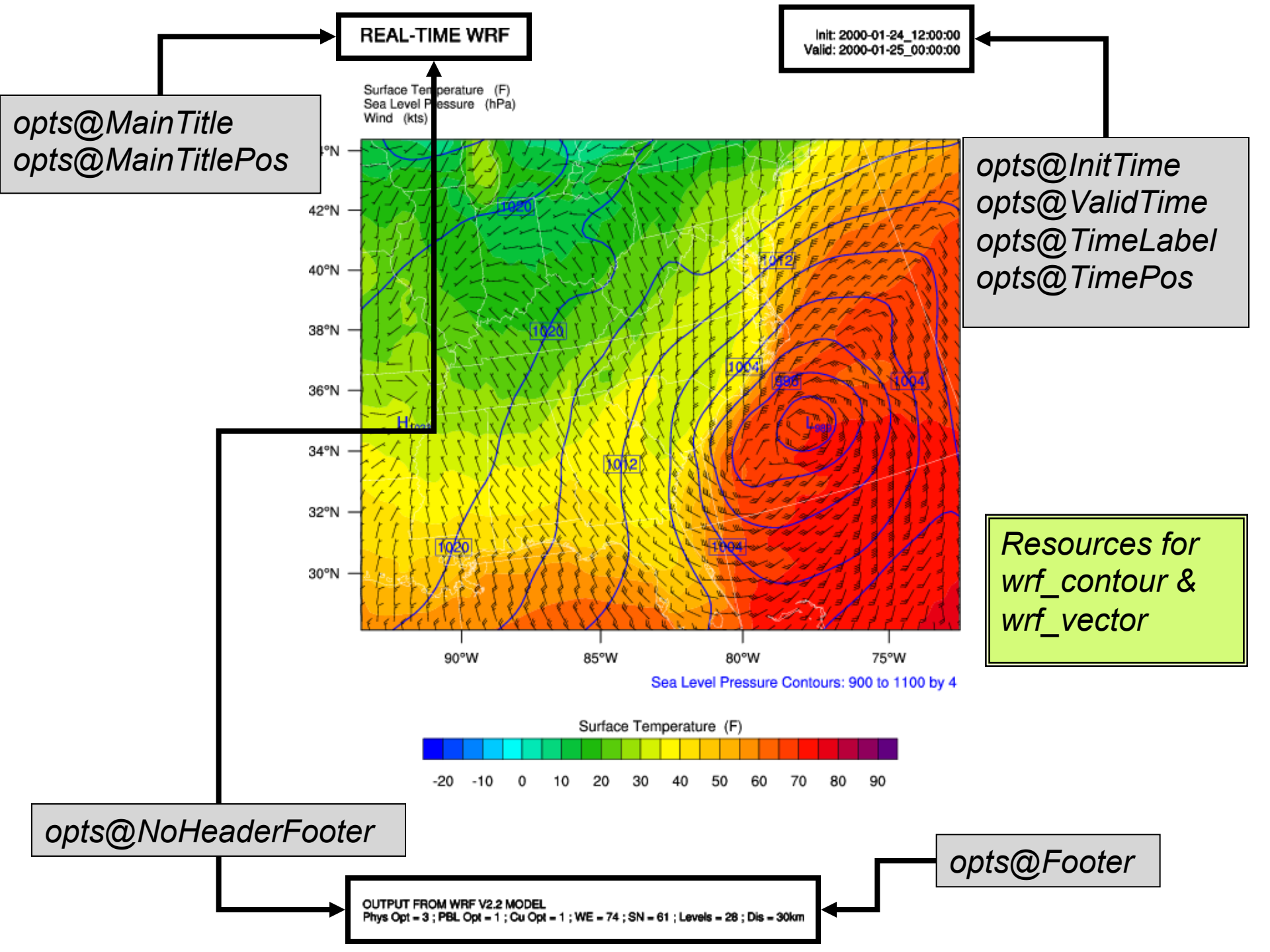
OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

wrf_contour/wrf_vector

Create line/shaded/filled contours and vectors

```
contour = wrf_contour(f, wks, ter, opts)
vector = wrf_vector(f, wks, u, v, opts)
```

opts@MainTitle	Main title on the plot
opts@MainTitlePos	Main title position (default=left)
opts@NoHeaderFooter	Turn off headers & footers (default=False)
opts@Footer	Add model information as a footer (default=True)
opts@InitTime	Plot initial time on graphic (default=True)
opts@ValidTime	Plot valid time on graphic (default=True)
opts@TimeLabel	Label to use for valid time
opts@TimePos	Time position (default=right)
opts@ContourParameters	Contour parameters
opts@FieldTitle	Overwrite the field title
opts@UnitLabel	Overwrite the field units
opts@PlotLevelID	Add level information to field title
opts@NumVectors	Density of wind vector *(wrf_vector) (default=25)*


REAL-TIME WRF
Init: 2000-01-24_12:00:00
Valid: 2000-01-25_00:00:00
opts@MainTitle
opts@MainTitlePos
opts@InitTime
opts@ValidTime
opts@TimeLabel
opts@TimePos
Surface Temperature (F)
Sea Level Pressure (hPa)
Wind (kts)
42°N
40°N
38°N
36°N
34°N
32°N
30°N
90°W
85°W
80°W
75°W
1020
1012
1004
996
Sea Level Pressure Contours: 900 to 1100 by 4
Surface Temperature (F)
-20 -10 0 10 20 30 40 50 60 70 80 90
Resources for wrf_contour & wrf_vector
opts@NoHeaderFooter
opts@Footer
OUTPUT FROM WRF V2.2 MODEL
Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1 ; WE = 74 ; SN = 61 ; Levels = 28 ; Dis = 30km

wrf_map_overlays/wrf_overlays

Overlay plots created with wrf_contour and wrf_vector

```
plot = wrf_map_overlays (f, wks, (/contour,vector/), \
                                 pltres, mpres)
plot = wrf_overlays (f, wks, (/contour,vector/), \
                            pltres)
```

To zoom in, set:

mpres@ZoomIn = True

and

mpres@Xstart
mpres@Xend
mpres@Ystart
mpres@Yend

to the corner x/y positions of the zoomed plot. You can use wrf_user_ll_to_ij to get the values for X/Ystart/end

http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/Examples/SPECIAL/wrf_Zoom.htm

wrf_map_overlays/wrf_overlays (cont'd)

Overlay plots created with wrf_contour and wrf_vector

```
  mpres          = True
  opts           = True
  opts@cnFillOn = True
  contour = wrf_contour(a,wks,ter,opts)
  plot    = wrf_map_overlays(a,wks,(/contour/),pltres,mpres)
```

```
; As an example, look at the lower right 1/4 of the domain
  dims    = dimsizes(ter)
  x_start = dims(1)/2
  x_end   = dims(1)-1
  y_start = 0
  y_end   = dims(0)/2
  ter_zoom = ter(y_start:y_end,x_start:x_end)
  mpres          = True
  opts           = True
  opts@cnFillOn = True
  mpres@ZoomIn   = True
  mpres@Xstart   = x_start
  mpres@Ystart   = y_start
  mpres@Xend     = x_end
  mpres@Yend     = y_end
  contour = wrf_contour(a,wks,ter_zoom,opts)
  plot    = wrf_map_overlays(a,wks,(/contour/),pltres,mpres)
```

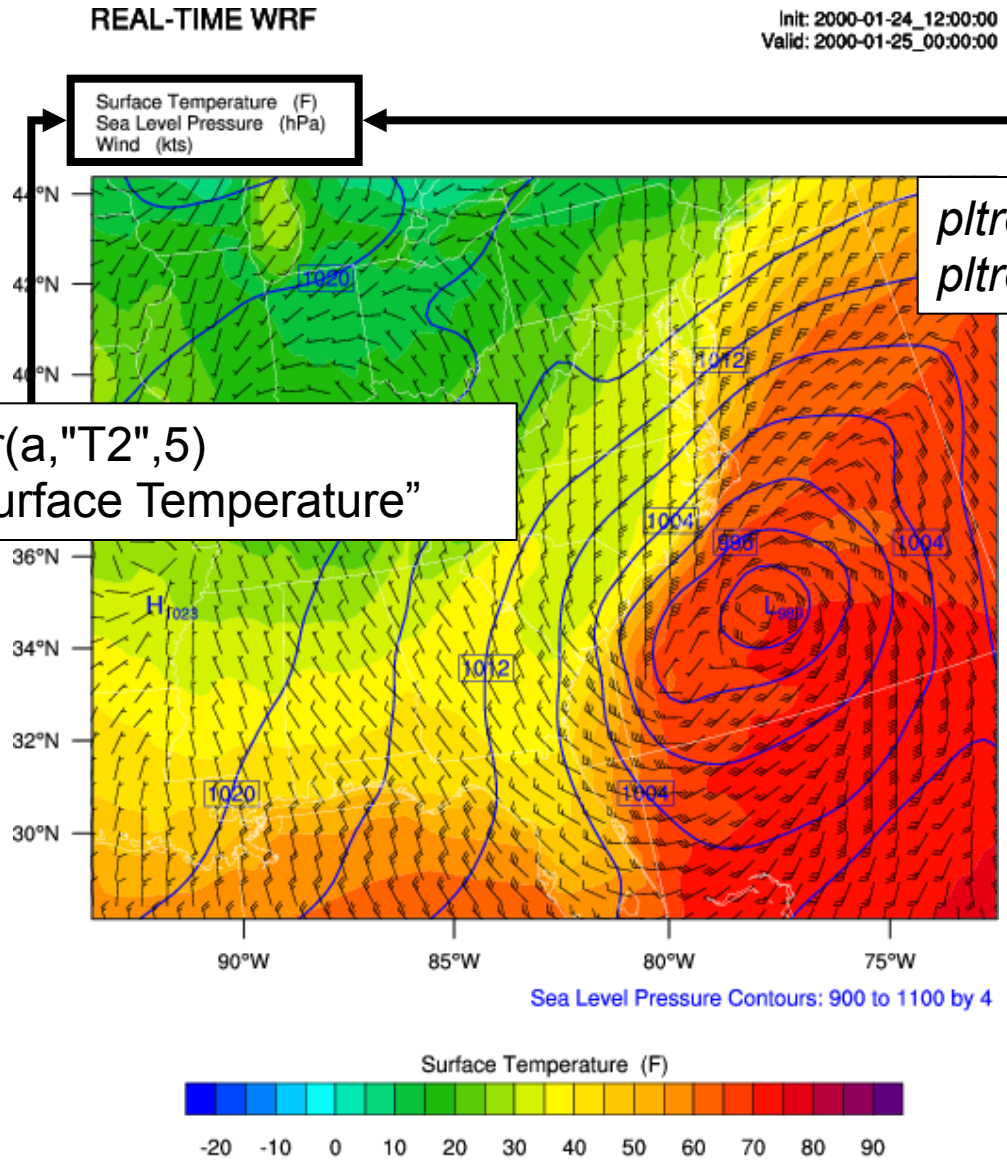
wrf_map_overlays/wrf_overlays

Overlay plots created with wrf_contour and wrf_vector

```
plot = wrf_map_overlays (f, wks, (/contour,vector/), \
                         pltres, mpres)
plot = wrf_overlays (f, wks, (/contour,vector/), \
                     pltres)
```

pltres@NoTitles	Turn off all titles
pltres@CommonTitle	Common title
pltres@PlotTitle	Plot title
pltres@PanelPlot	Whether a panel plot is to be drawn
pltres@FramePlot	Whether to advance the frame

Resources for wrf_overlays and wrf_map_overlays



pltres@NoTitles
pltres@CommonTitle

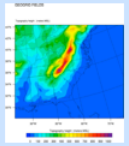
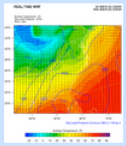
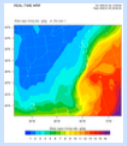
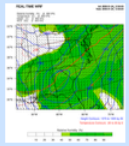
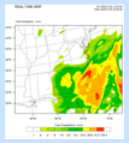
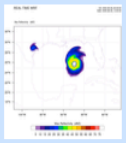
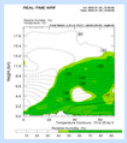
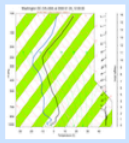
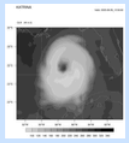
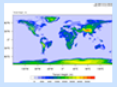
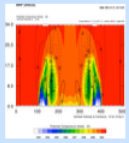
t2 = wrf_user_getvar(a,"T2",5)
t2@description = "Surface Temperature"

http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/

Latest version of WRFUserARW.ncl file usually available here.

Scripts and full-sized images available.

Google
"WRF ARW NCL"

Basic Plots  Basic Plot Setup *(This series of examples takes users though same basic steps in generating plotting scripts.)* Get and plot a single field Multiple input files	**Basic Surface Plots**  Surface 1 Surface 3 Surface 2	**Plots on Model Levels**  Clouds Levels from wrfout files Levels from metgrid files	**Plots on Interpolated Levels**  Height Levels Pressure Levels
Plotting Precipitation  Precipitation	**Diagnostics**  CAPE dBZ Vorticity *(More diagnotics are available, shown are only some newer/special diagnostics)*	**Cross-section Plots**  Height - Through a Pivot Point Height - Point A to Point B Pressure Limited Vertical Extent For 2D fields	**Skew_T Plots**  Skew_T
Speciality Plots  Overlay Zoom Overlay & Zoom Panel 1 Panel 2 Meteograms WRF Time Series data All fields in a file	**Preview Domain**  *This functionality, although available in NCL version 5.0.1, is still experiential.* Preview	**Global WRF**  gWRF_merc	**Idealized cases**  wrf_Grav2x wrf_Hill2d wrf_Squall_2d_x wrf_Squall_2d_y wrf_Seabreeze2x wrf_BWave wrf_QSS

Topics

- Overview
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Tips, where to get help
- What's new

Calling Fortran codes from NCL

- Easier to use F77 code, but works with F90 code
- Need to isolate definition of input variables and wrap with special comment statements:

```
C NCLFORTSTART
C NCLEND
```

- Use a tool called WRAPIT to create a *.so file
- Load *.so file in NCL script with "external" statement
- Call Fortran function with special "::" syntax
- Must preallocate arrays! (using NCL's "new" statement)

http://www.ncl.ucar.edu/Document/Tools/WRAPIT.shtml

Example F77 code: *myTK.f*

```
C NCLFORTSTART
      subroutine compute_tk(tk,pressure,theta,nx,ny,nz)
      implicit none
      integer nx, ny, nz
      real    tk(nx, ny, nz)
      real    pressure(nx, ny, nz), theta(nx, ny, nz)
C NCLEND
      integer i, j, k
      real    pi

      do k=1,nz
        do j=1,ny
          do i=1,nx
            pi = (pressure(i,j,k)/1000.)**(287./1004.)
            tk(i,j,k) = pi*theta(i,j,k)
          end do
        end do
      end do
      end
```

Create "myTK.so" file and use in script

```
% WRAPIT myTK.f
```

This will create a "myTK.so" file

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
external myTK "./myTK.so"

begin
      t = wrf_user_getvar(a,"T",5)
      t = t + 300
      p = wrf_user_getvar(a,"pressure",5)

; Must preallocate space for output arrays
      dim = dimsizes(t)
      tk  = new( dimsizes(t), typeof(t) )

; Remember, Fortran/NCL arrays are ordered differently
      myTK :: compute_tk (tk,p,t,dim(2),dim(1),dim(0))
end
```

Calling Fortran 90 codes from NCL

- Can use simple Fortran 90 code
- Your F90 program cannot contain any of the following features:
  - pointers or structures as arguments
  - missing or optional arguments
  - keyword arguments
  - recursive procedures
- The input arguments must be reproduced in a separate F77-like “stub” file
- “WRAPIT” is a modifiable script

Example F90 code: *myTK.f90*

myTK.f90

```
subroutine compute_tk (tk, pres, theta, nx, ny, nz)
  implicit none
  integer :: nx,ny,nz
  real, dimension (nx,ny,nz) :: tk, pres, theta, pi

  pi = (pres/1000.)**(287./1004.)
  tk = pi * theta

end subroutine compute_tk
```

Example F90 code: *myTK.f90* + *stub*

myTK.f90

```
subroutine compute_tk (tk, pres, theta, nx, ny, nz)
  implicit none
  integer :: nx,ny,nz
  real, dimension (nx,ny,nz) :: tk, pres, theta, pi

  pi = (pres/1000.)**(287./1004.)
  tk = pi * theta

end subroutine compute_tk
```

myTK.stub

```
C NCLFORTSTART
  subroutine compute_tk (tk, pres, theta, nx, ny, nz)
  implicit none
  integer nx,ny,nz
  real    tk(nx,ny,nz)
  real    pres(nx,ny,nz), theta(nx,ny,nz)
C NCLEND
```

Create "myTK.so" file and use in script

```
% WRAPIT myTK.stub myTK.f90
```

Should create a "myTK.so" file. Script will be exactly the same.

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
external myTK "./myTK.so"

begin
      t = wrf_user_getvar(a,"T",5)
      t = t + 300
      p = wrf_user_getvar(a,"pressure",5)

; Must preallocate space for output arrays
      dim = dimsizes(t)
      tk  = new( dimsizes(t), typeof(t) )

      myTK :: compute_tk (tk,p,t,dim(2),dim(1),dim(0))
end
```

Topics

- Overview
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Tips, where to get help
- What's new

Debugging tips

- Start with an existing script, if possible
- Use editor enhancements for coloring of syntax, functions, etc
- Use indentation (even though not needed)
- Use “ncl_filedump” to look at file quickly
- Use “printVarSummary” to examine variables
  - Check for no “_FillValue” or wrong “_FillValue” value

Debugging tips (cont'd)

- To further examine data, use:
  - print(min(x)) and print(max(x)) ; Min/Max of data
  - print(num(ismissing(x))) ; Count # of msg vals
  - print(dimsizes(x)) ; Print dimension sizes
- Read documentation for functions
- For graphics, make sure spelling the resource name correctly
- Read errors and warnings carefully

Inefficient code

- Nested do loops, unnecessary code in do loops
  - Try to use f90-style arithmetic where possible
  - If code doesn't need to be in do loop (like initializing a variable), move it outside the loop
- Copying metadata unnecessarily. Use (/ and /) to avoid this:

```
ch4_tmp = (/ch4/)
```

- Creating lots of big arrays and not deleting them when no longer needed. Use NCL's "delete" procedure to clean up.
- Reordering the same array multiple times
  - Do once and store to local variable

Improving memory efficiency in NCL using array arithmetic

Nested do loop: 9.6 CPU seconds

```
; nx = ny = nz = 100
tk = new((/nx,ny,nz/),float)
do k=0,nz-1
  do j=0,ny-1
    do i=0,nx-1
      pi = (p(i,j,k)/1000.)^(287./1004.)
      tk(i,j,k) = pi*theta(i,j,k)
    end do
  end do
end do
```

Using NCL's array arithmetic: 0.12 CPU seconds

```
; nx = ny = nz = 100
pi = (p/1000.)^(287./1004.)
tk = pi*theta
```

80x faster!

Problems installing or running NCL?

- Send email to ncl-install@ucar.edu or ncl-talk@ucar.edu (must subscribe first):

  http://mailman.ucar.edu/mailman/listinfo/ncl-install
  http://mailman.ucar.edu/mailman/listinfo/ncl-talk

- Be specific about problem:
  - What kind of machine ("uname –a")
  - Which version of NCL, or which file did you download? ("ncl –V")
  - What exactly is the problem? Include what you are trying to do, and exactly what error message you got.

Customizing your NCL graphics environment

~/.hluresfile

- Download ".hluresfile" file, put in home directory!!
  - ✪ Changes your background, foreground colors to white/black
  - ✪ Changes font from **times-roman** to **helvetica**
  - ✪ Changes "function code" from ':' to '~'
  - ***WRF-NCL users:*** use to change the default color map

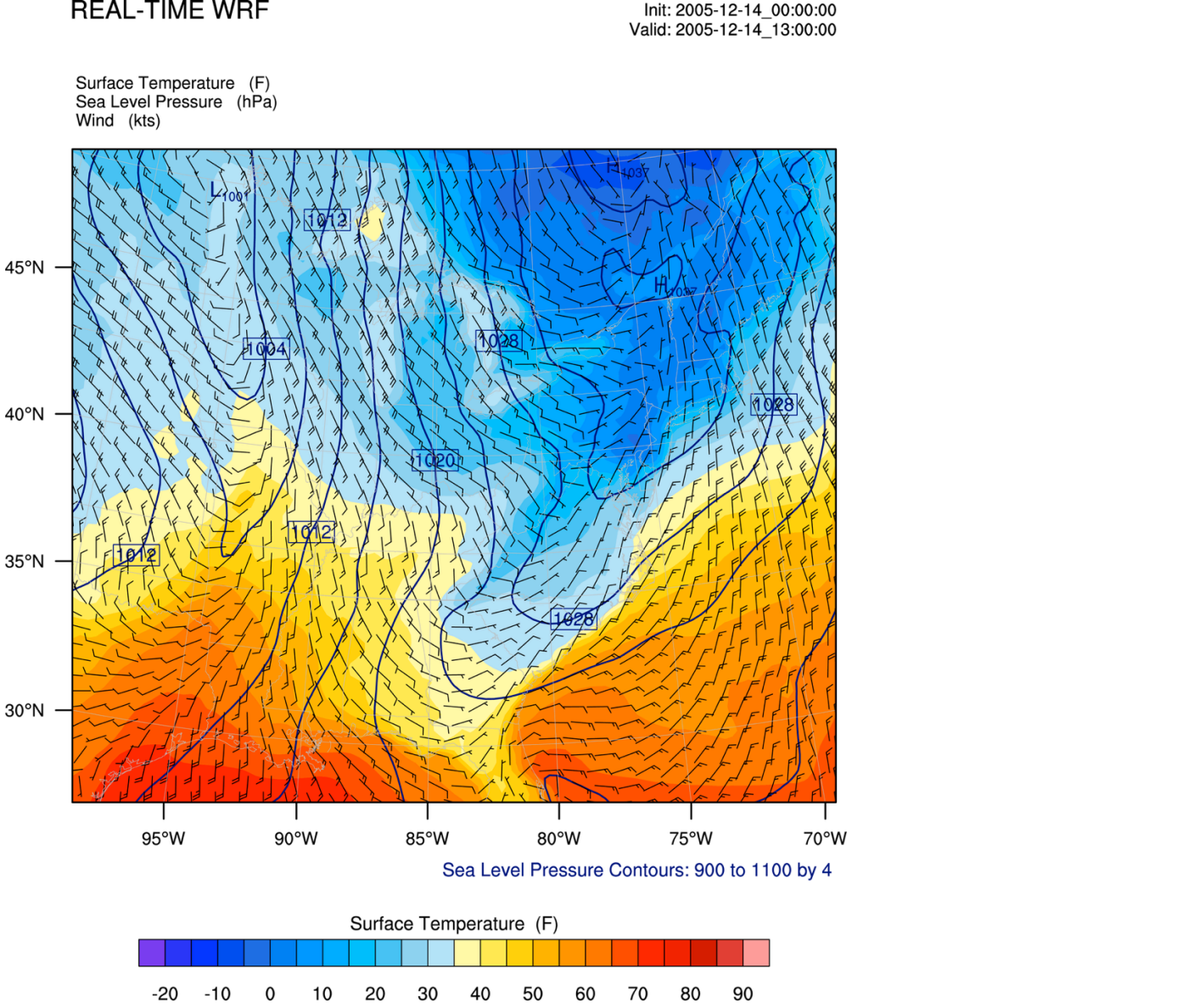
✪ *These are the defaults in V6.1.0 and later*

http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml

Sample ".hluresfile"

```
*wkColorMap          : BlAqGrYeOrReVi200

*wkWidth             : 1500

*wkHeight            : 1500
```

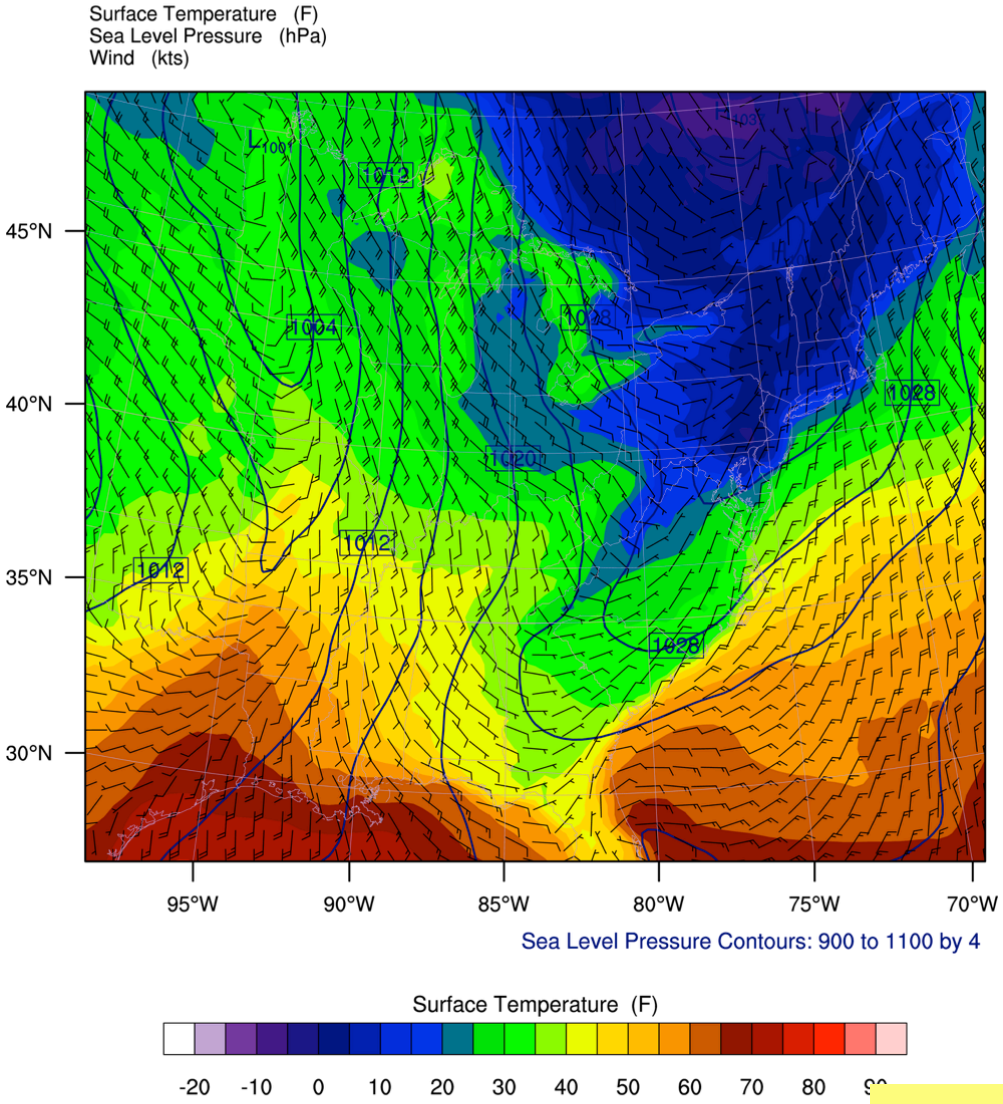
wkWidth and wkHeight affect size of PNG and X11



NCL default color table

REAL-TIME WRF

Init: 2005-12-14_00:00:00
Valid: 2005-12-14_13:00:00



Changed default color map to BlAqGrYeOrReVi200

OUTPUT FROM WRF V2.1.1 MODEL
WE = 98 ; SN = 84 ; Levels = 37 ; Dis = 30km ; Phys Opt = 2 ; PBL Opt = 1 ; Cu Opt = 1

Useful URLS

- Online WRF-NCL Graphics Tutorial
  http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/
- Editor Enhancements
  http://www.ncl.ucar.edu/Applications/editor.shtml
- Mini NCL reference manual
  http://www.ncl.ucar.edu/Document/Manuals/language_man.pdf
- WRF-NCL functions (built-in and “WRFUserARW.ncl”)
  http://www.ncl.ucar.edu/Document/Functions/wrf.shtml
- Download NCL
  http://www.ncl.ucar.edu/Download/
- Application examples (includes WRF and shapefile examples)
  http://www.ncl.ucar.edu/Applications/
- NCL Workshops
  http://www.ncl.ucar.edu/Training/Workshops/
- NCL email lists to join
  http://www.ncl.ucar.edu/Support/email_lists.shtml

NCA

Topics

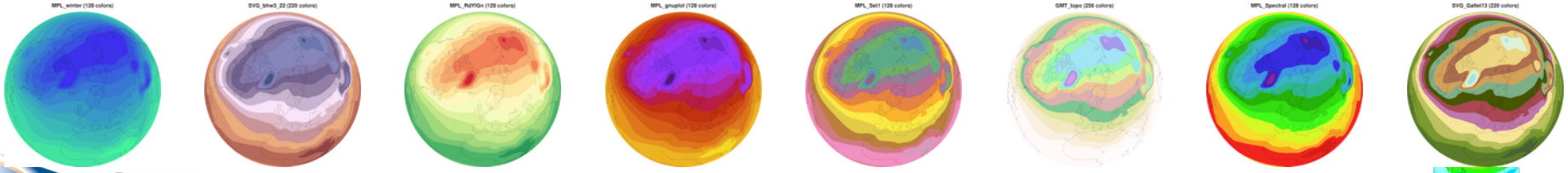
- Overview
- NCL language basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Tips, where to get help
- What's new

What's new in NCL V6.2.0

Released April 2, 2014

http://www.ncl.ucar.edu/current_release.shtml

- Significant speed-up for raster contouring and primitives
- New WRF diagnostics in wrf_user_getvar: omega ("omg"), virtual temperature ("tv"), wet bulb temperature ("tw")
- Significant file I/O improvements and bug fixes
- New "svg" graphical output format
- Library for generating KML output
- Over 120 new color tables



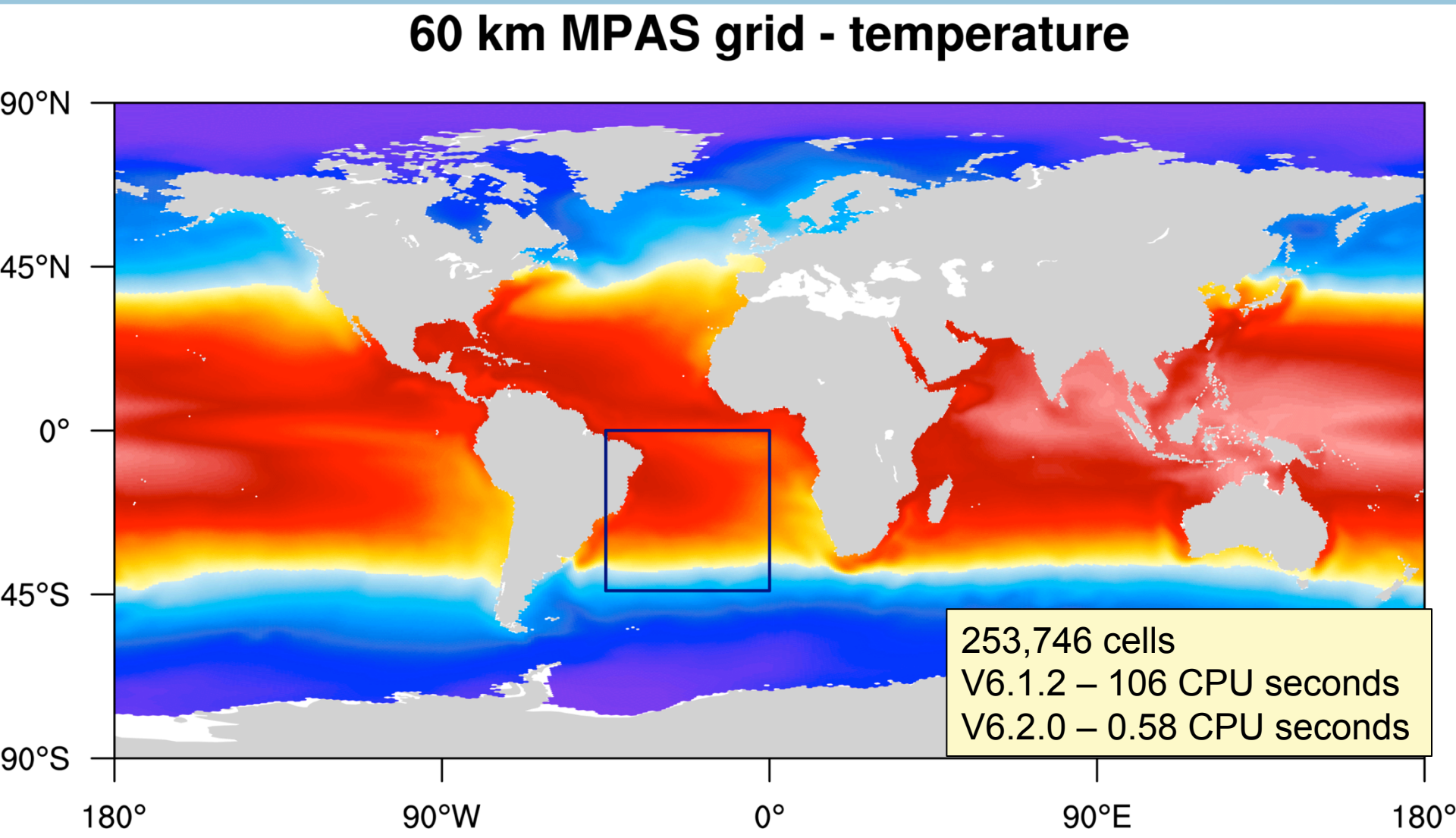
What's new in NCL V6.2.0 (cont'd)

Released April 2, 2014

http://www.ncl.ucar.edu/current_release.shtml

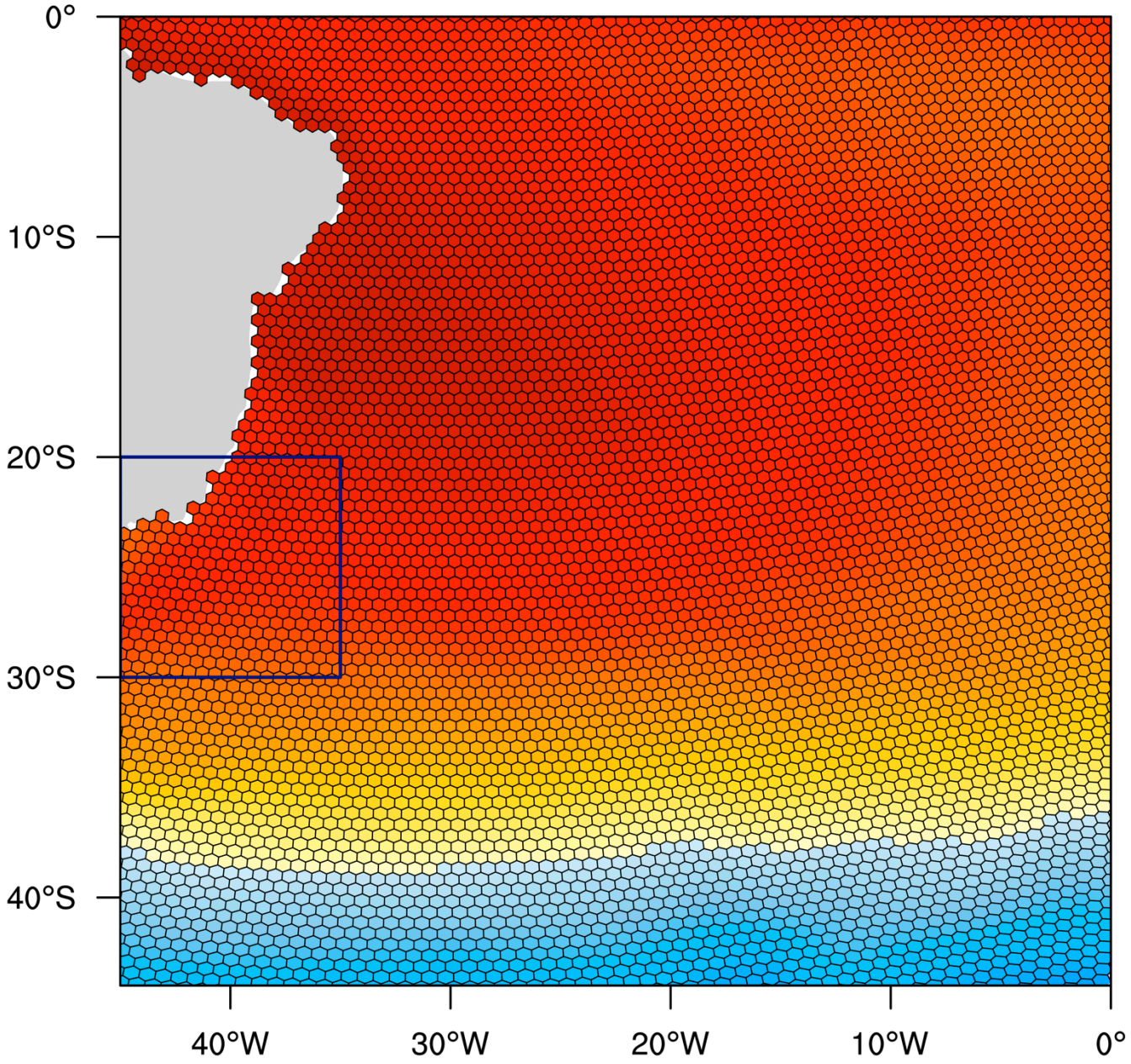
- New functions
  - Simple and multiple linear regression analyses
  - Kolmogorov-Smirnov two sample test
  - Area of a polygon
  - Several more
- Bug fixes to existing functions
  - Plotting rotated WRF lat/lon grids
- Several new graphical resources

Faster viewing of high-res MPAS grid



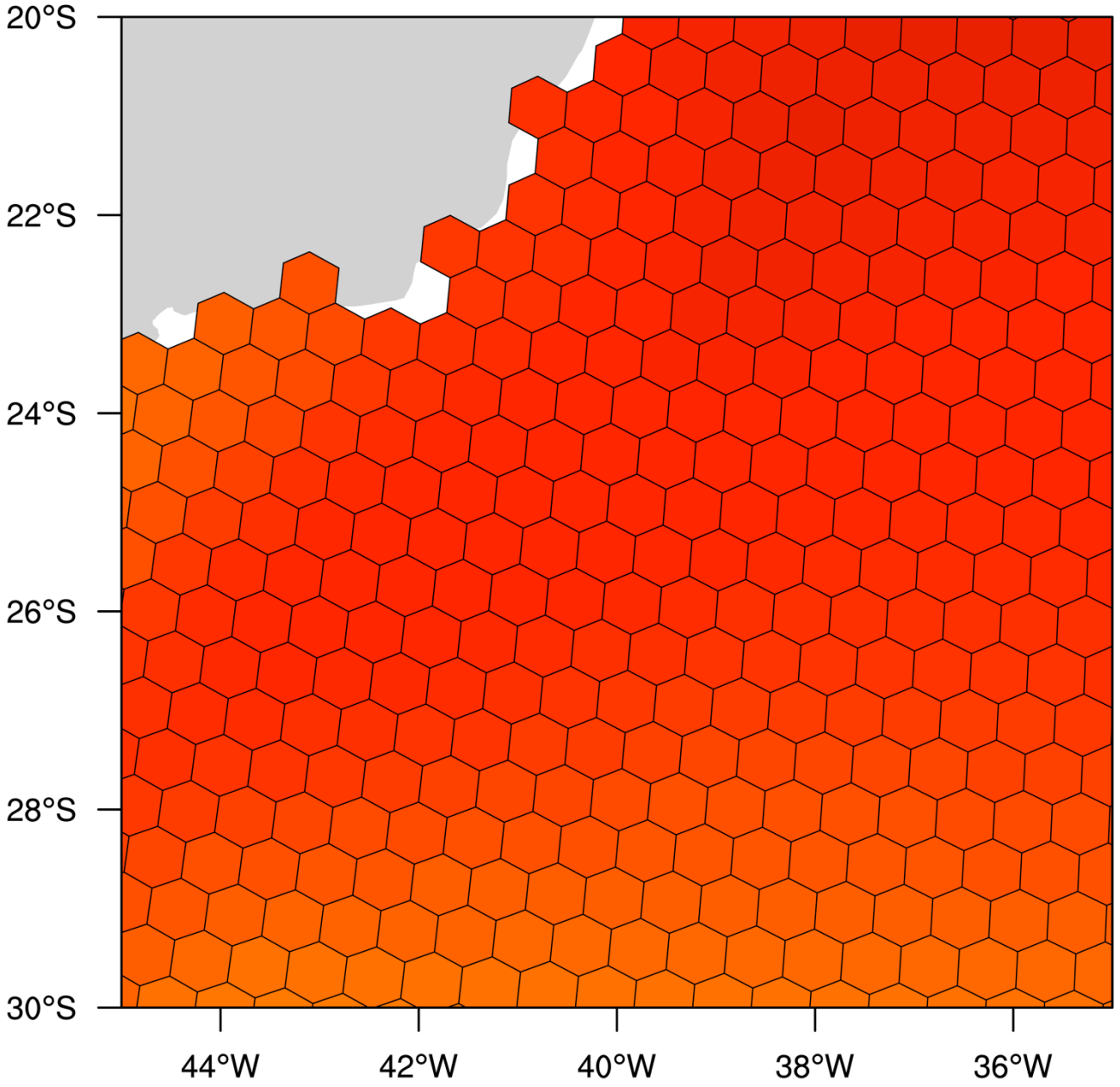
60 km MPAS grid - temperature

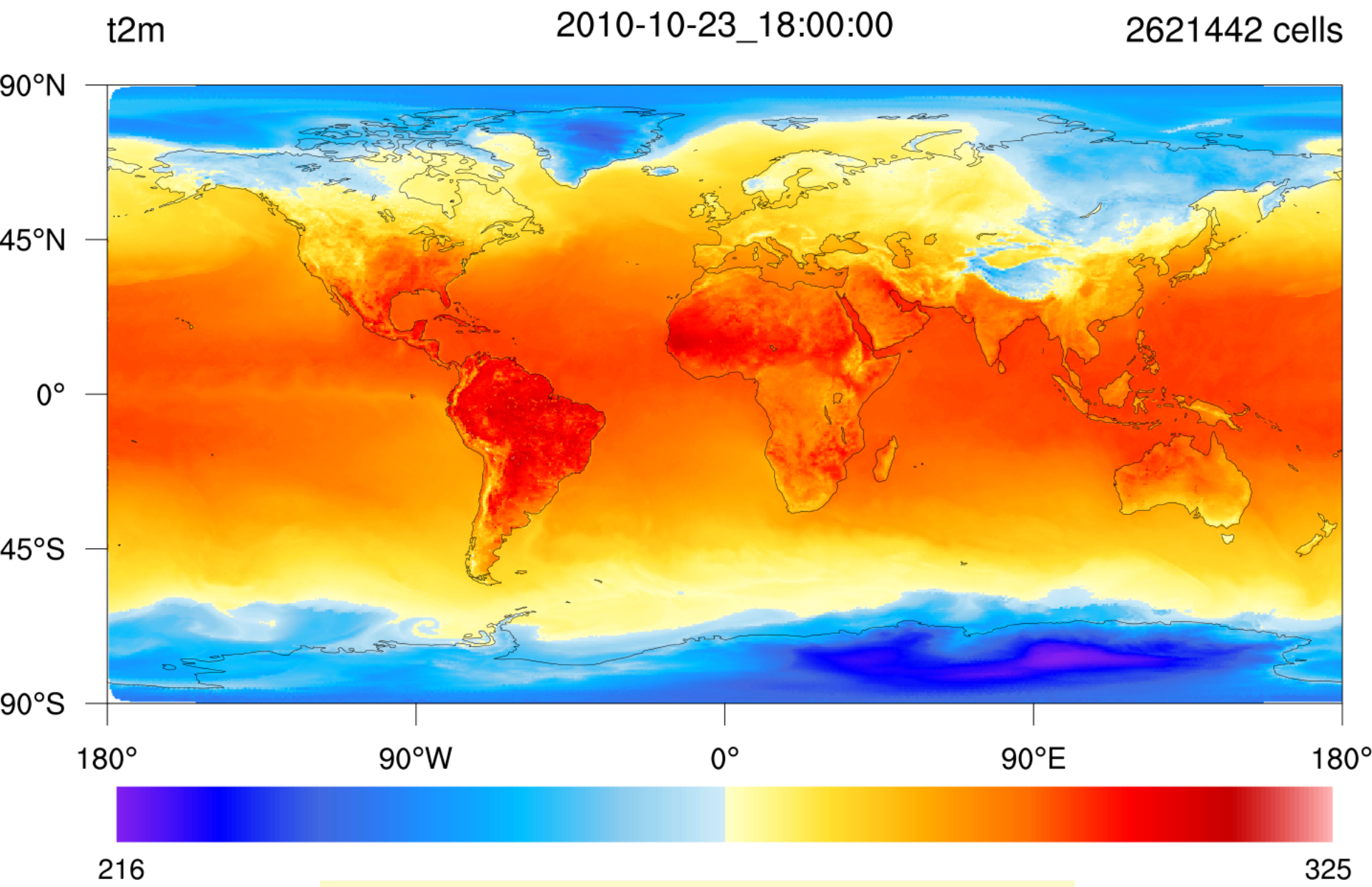
Fast drawing of the MPAS edges



60 km MPAS grid - temperature

Zoomed in view of MPAS edges





2,621,442 cells
V6.1.2 – 83.5 CPU seconds
V6.2.0 – 14.5 CPU seconds

Data from Michael Duda (MMM)

What's coming in V6.2.1/V6.3.0

- Version 6.2.1 – late summer 2014
  - A bug-fix release
- Version 6.3.0 – Winter 2015
  - Priorities still being fleshed out
  - Further speed-up for graphics
  - Focus on parallelism
  - Better support for shapefiles (write capability)
  - Quick-look tool in GoogleEarth-like environment
  - Drawing of vectors on unstructured grids
  - Code modernization
- Python releases – WRF functions added to PyNGL, major PyNIO release coming

Questions?

Mary Haley (haley@ucar.edu)

wrfhelp@ucar.edu

Questions specific to WRF-NCL

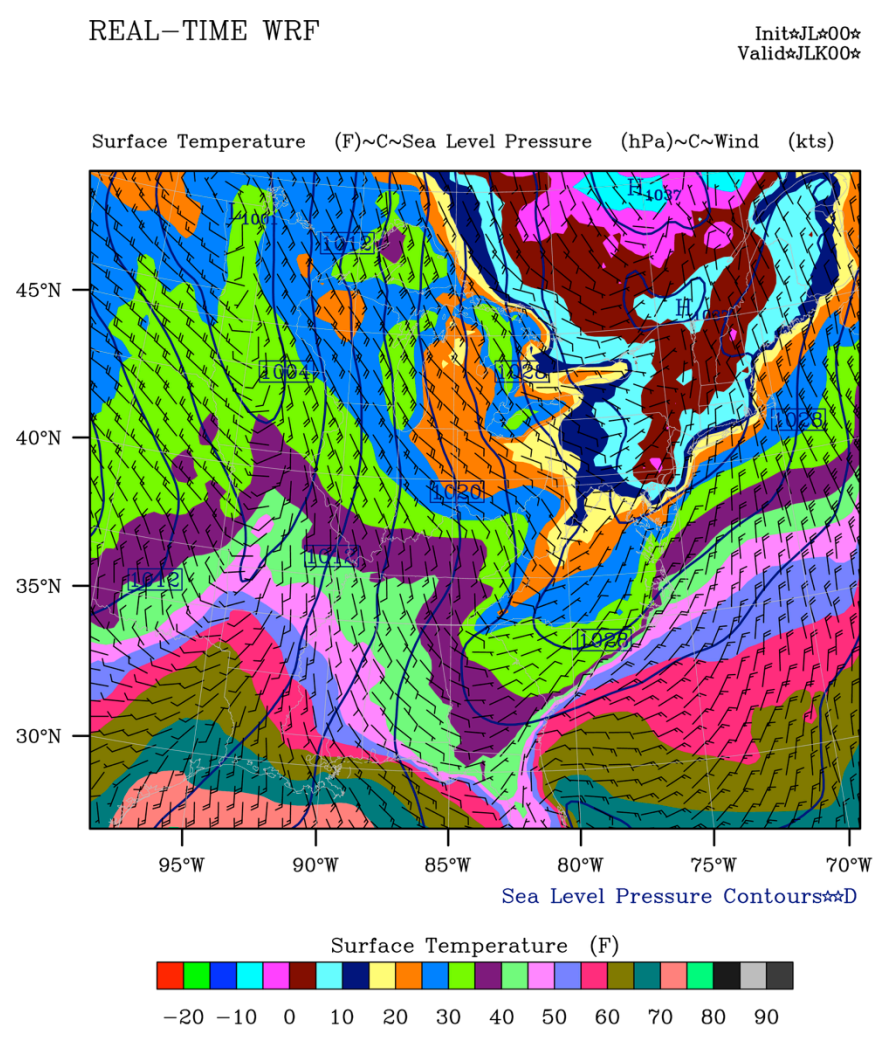
ncl-talk@ucar.edu

Issues with NCL
(must subscribe first)

http://mailman.ucar.edu/mailman/admin/ncl-talk

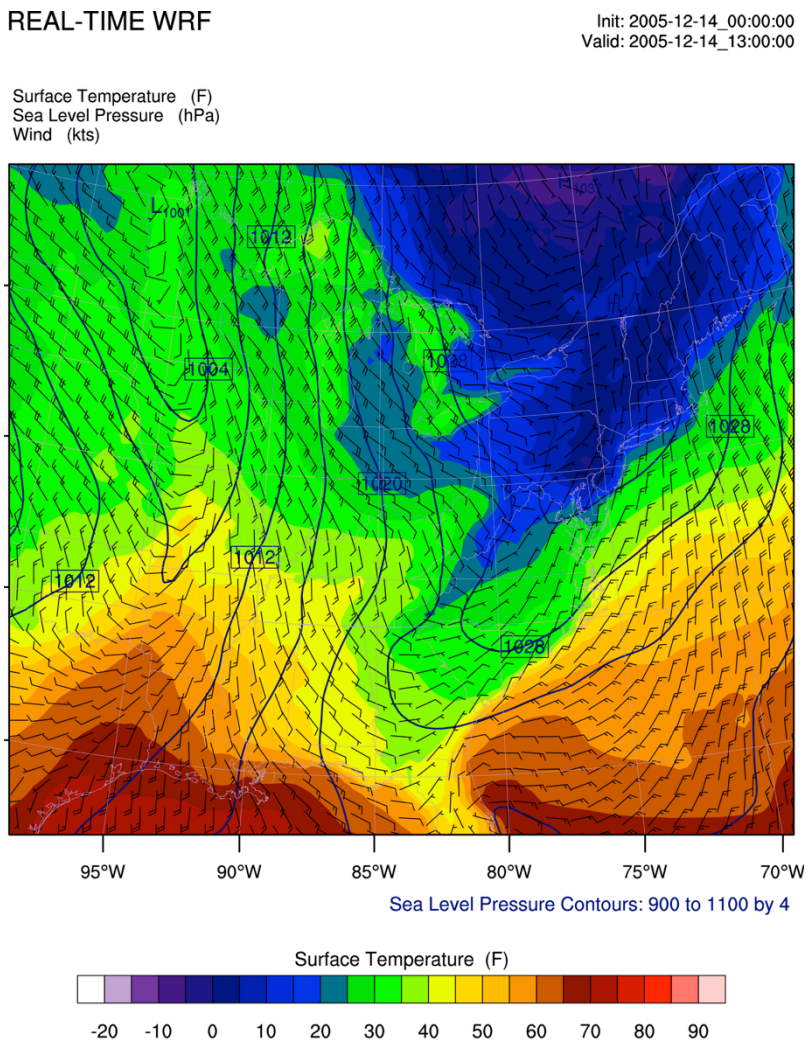
In **NCL V6.0.0** and earlier…

without ~/.hluresfile



OUTPUT FROM WRF V2.1.1 MODEL~C~ WE = 98 ; SN = 84 ; Levels = 37 ; Dis = 30km ; Phys Opt = 2 ; PBL Opt

with ~/.hluresfile



OUTPUT FROM WRF V2.1.1 MODEL
WE = 98 ; SN = 84 ; Levels = 37 ; Dis = 30km ; Phys Opt = 2 ; PBL Opt = 1 ; Cu Opt = 1

Installing NCL and setting up environment

- www.earthsystemgrid.org (login/password)
- Download appropriate precompiled binary
- Run “tar –zxvf” on the *.tar.gz file
- setenv NCARG_ROOT to parent directory
- Add $NCARG_ROOT/bin to search path
- Copy “.hluresfile” to home directory

http://www.ncl.ucar.edu/Download/

http://www.ncl.ucar.edu/Download/install.shtml

NCL and GoogleEarth™

- Summer 2013 intern project in CISL
- Mohammad Abouali, intern; Alan Norton and Rick Brownrigg: mentors
- A library of NCL routines has been developed to enable earth scientists to easily convert geo-referenced model output and other data to KML for display in Google Earth.