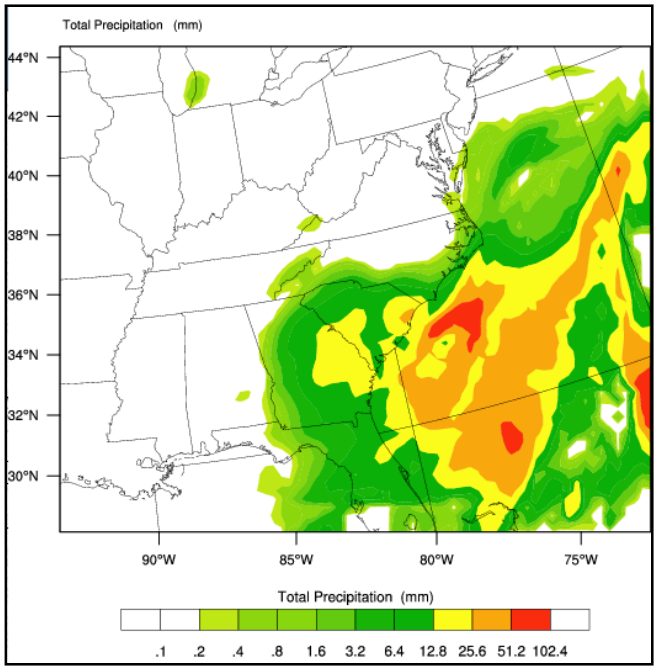
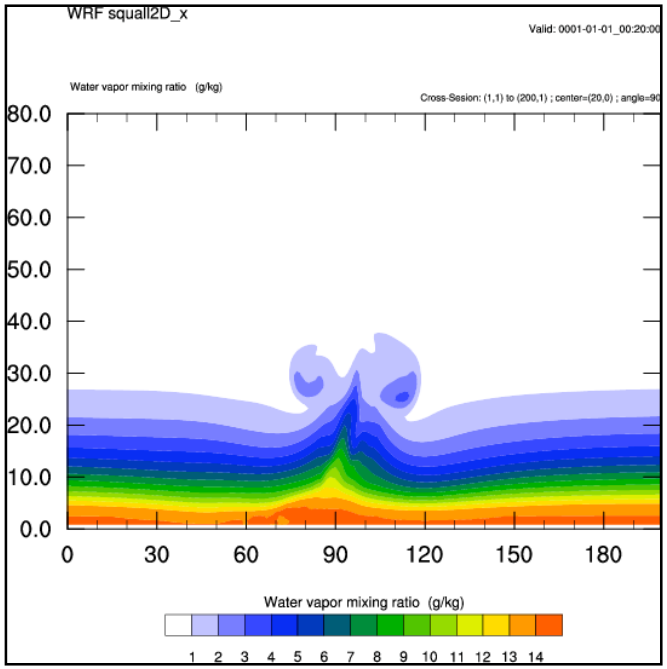
Post-processing WRF-ARW Data with the NCAR Command Language

Mary Haley (with tips from Cindy Bruyère)

11th Annual WRF Users' Event, June 22-25, 2010



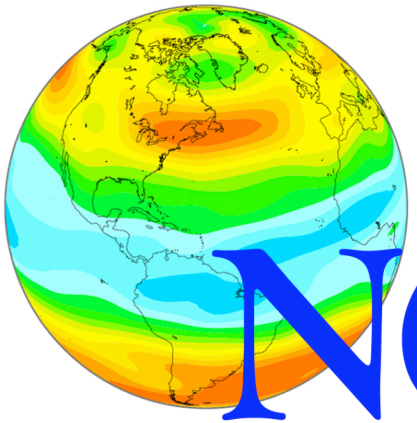


The National Center for Atmospheric Research is sponsored by the National Science Foundation.

Topics

- ✓ Overview of NCL and WRF-NCL
- ✓ NCL basics
- ✓ *Get you familiar with WRF-NCL scripts*
  - ✓ *Opening and examining a data file*
  - ✓ *Reading and querying variables*
  - ✓ *Plotting variables*
- ✓ Calling Fortran codes from NCL
- ✓ Debugging tips and common mistakes
- ✓ Installation and environment set-up
- ✓ Useful URLs

- Overview
- NCL basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs



NCL

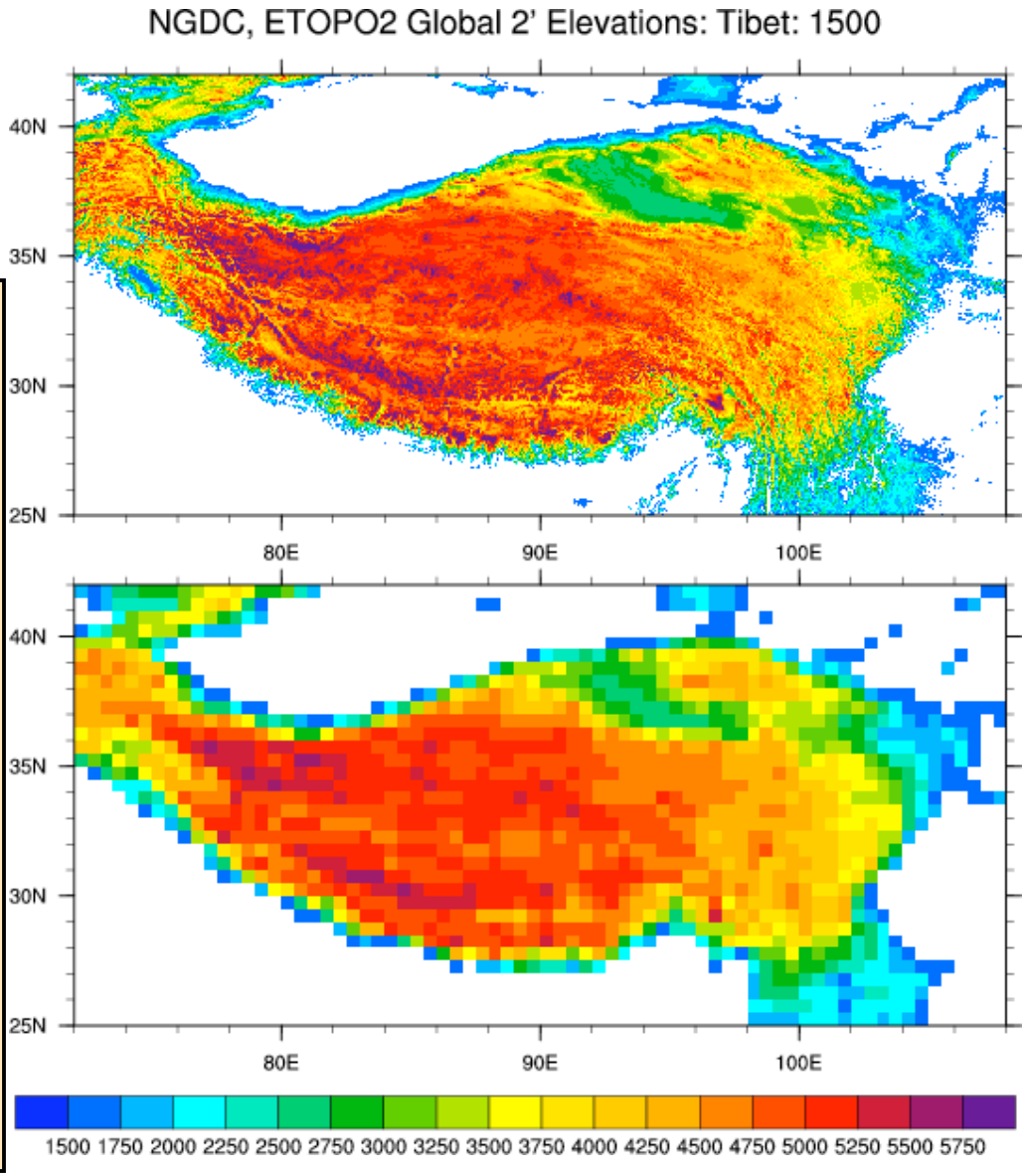
NCAR Command Language

A scripting (interpreted) language tailored for the analysis and visualization of geoscientific data

- Developed in NCAR/CISL in close collaboration with CGD scientists
- UNIX binaries and source available, **free**
- Extensive NCL website, hundreds of examples
- Hands-on workshops
- Email lists for consulting

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

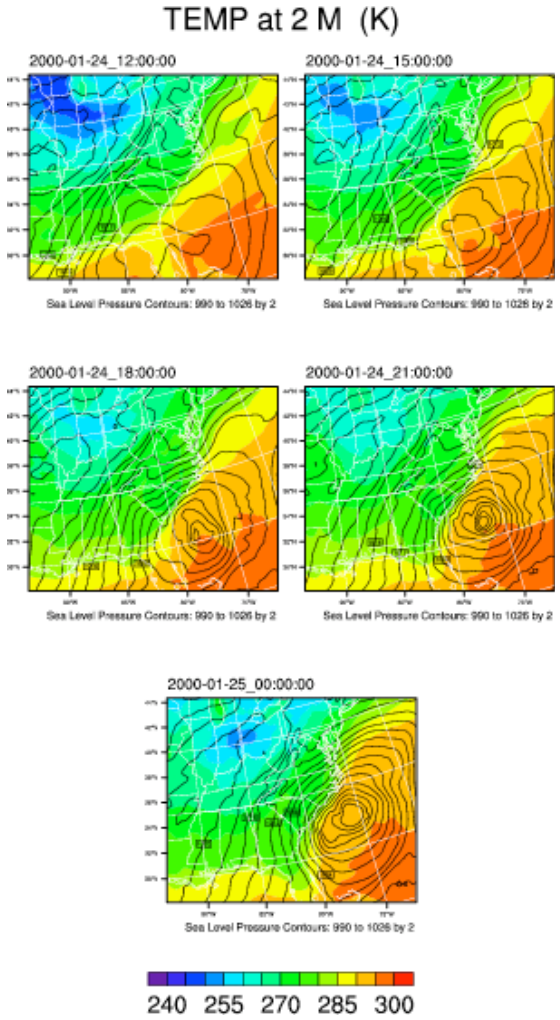
CGD CISL



WRF-NCL

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

- **Included with NCL**
- Developed by scientists in MMM
- Maintained by Cindy Bruyère/MMM and myself
- Functions for calculating basic diagnostics
- Functions for specialized visualizations – precipitation, surface, vorticity, meteograms, helicity, squall, dBZ, etc.
- Website with lots of analysis and visualization examples
- Workshops and tutorials (like this users' event, usually for WRF in general)
- Email list for consulting, wrfhelp@ucar.edu



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

More about NCL

- Similar to Python or IDL
- Tailored to climate and atmospheric sciences
- Has variable types, “if-then-endif”, “do” loops, arithmetic operators
- F90-like array arithmetic that will ignore missing values
- Can call your own Fortran 77/90 or C routines
- Simple, robust file input/output
- Hundreds of data analysis routines
- Publication-quality graphics that are highly customizable

NCL: File input and output

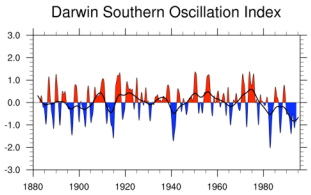
- Data model based on netCDF model (metadata describes data)
- One function reads all supported data formats:
  - NetCDF, GRIB 1 and 2, HDF4, HDF-EOS2, *shapefiles*, (*new: HDF-EOS5*)
    (next release: HDF5)
  - Writes NetCDF and HDF4 (compressed NetCDF too)
- OPeNDAP-enabled client available
- ASCII, binary (read and write)

http://www.ncl.ucar.edu/Applications/list_io.shtml

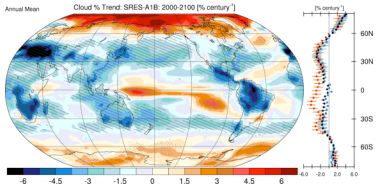
NCL: Data analysis

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

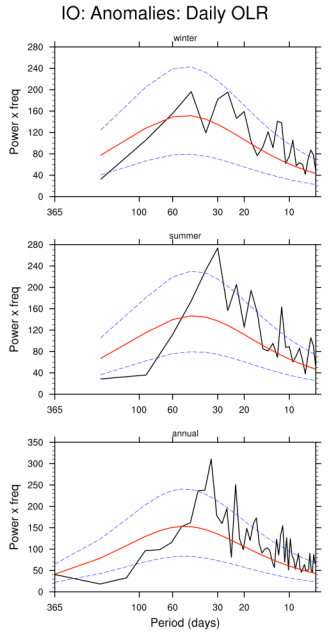
http://www.ncl.ucar.edu/Applications/list_dataP.shtml

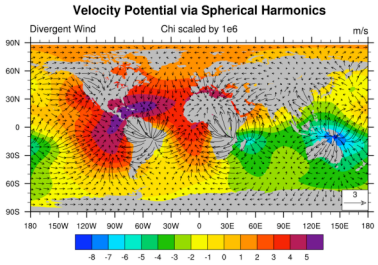


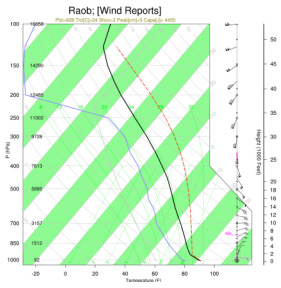
NCL: Visualization

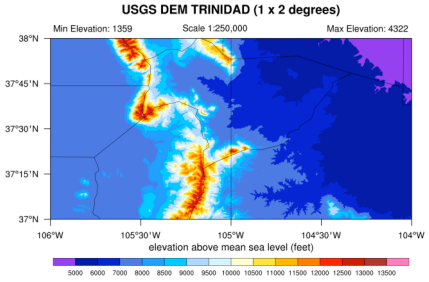


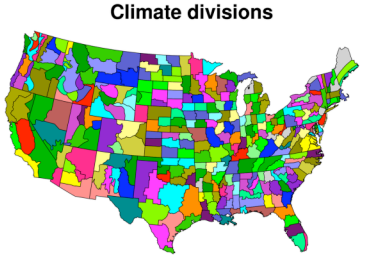
- High-quality and customizable visualizations
- Contours, XY, vectors, streamlines
- Maps with common map projections
- Handles data on regular and irregular grids, triangular meshes
- Specialized scripts for WRF-ARW data, meteograms, skew-T, wind roses, histograms, cross section, panels
- wrf_xxxx interfaces: simplifies visualization
- Over 1,400 visualization “options”







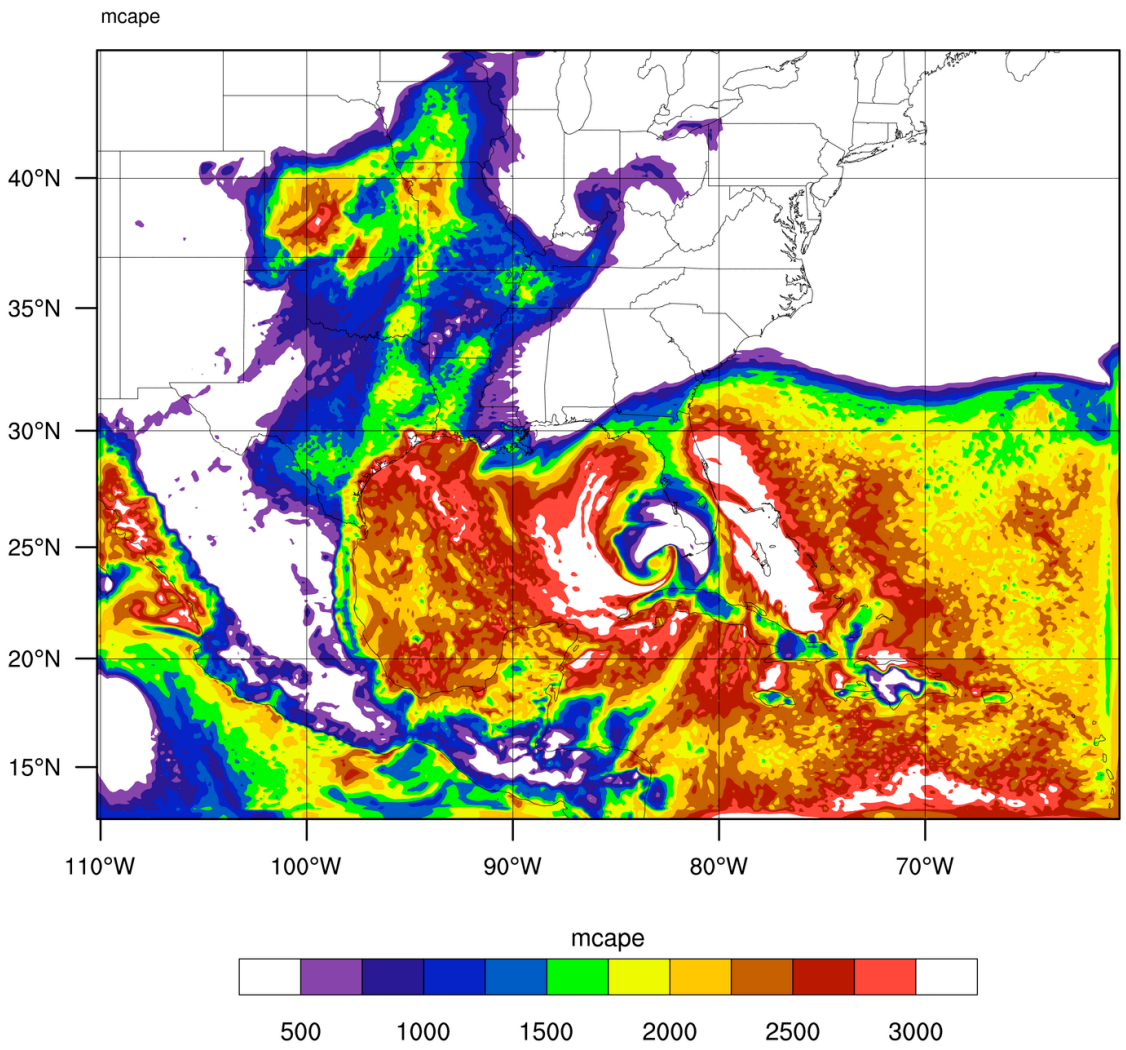




http://www.ncl.ucar.edu/gallery.shtml

REAL-TIME WRF

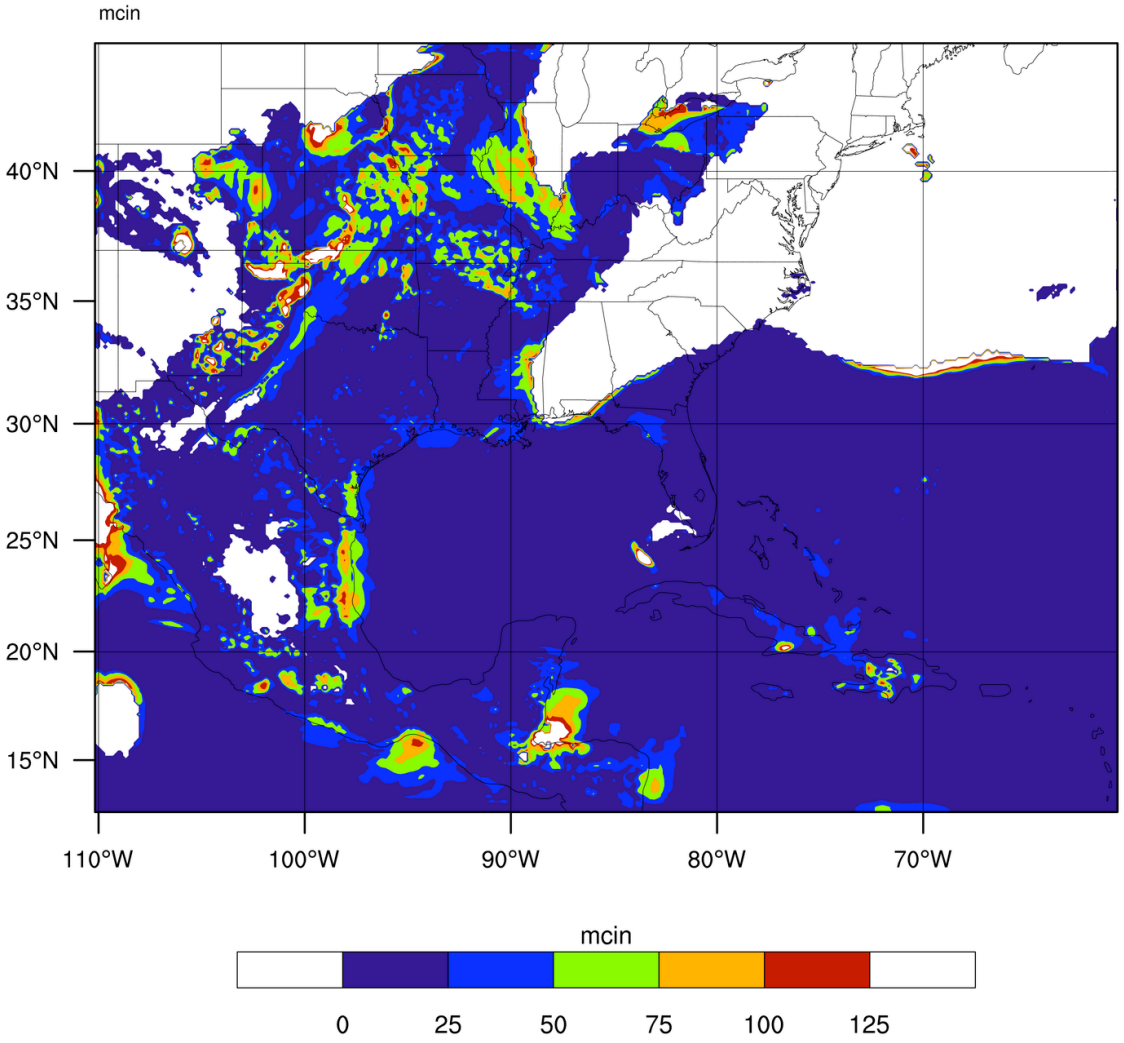
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



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

REAL-TIME WRF

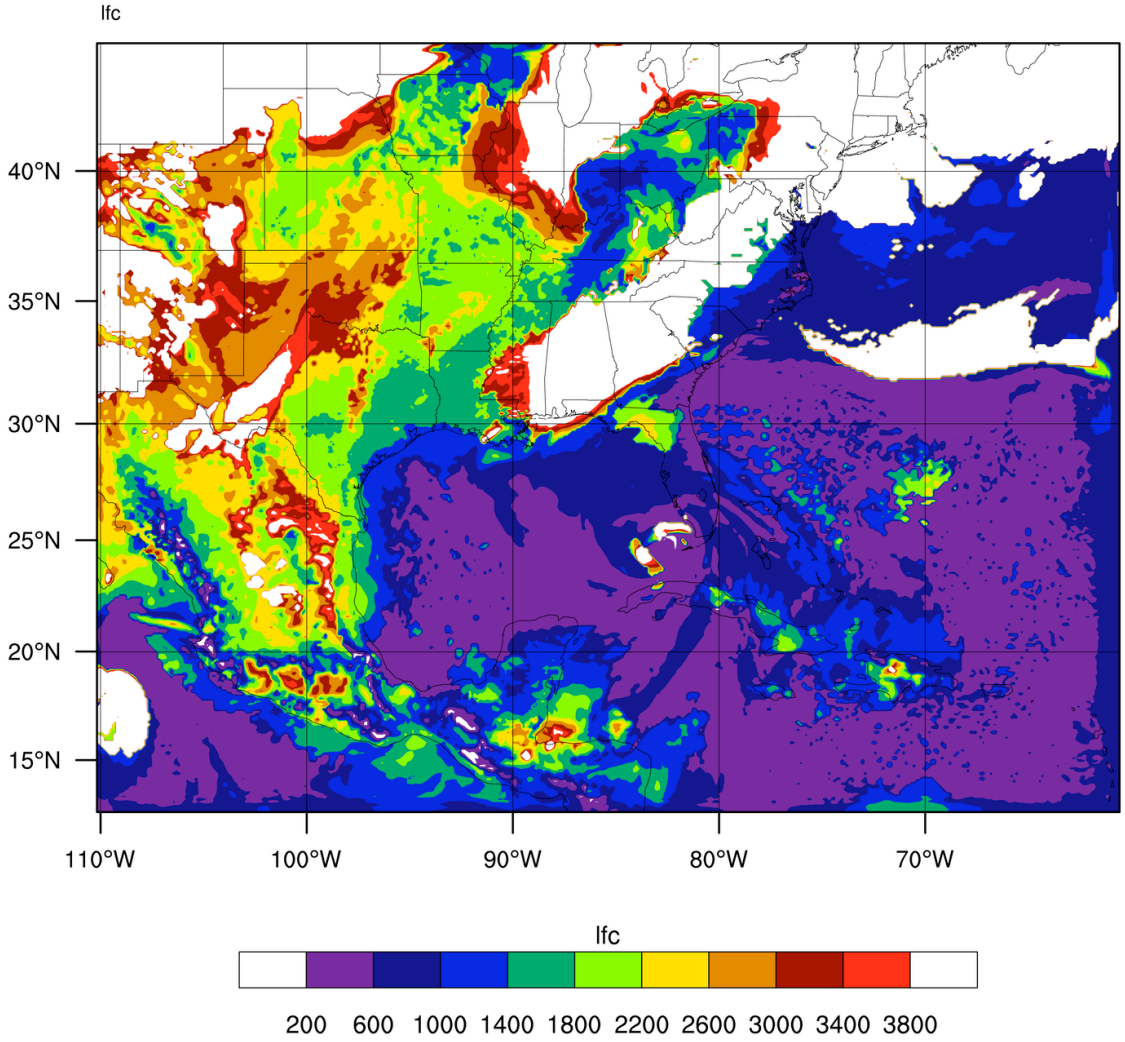
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



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

REAL-TIME WRF

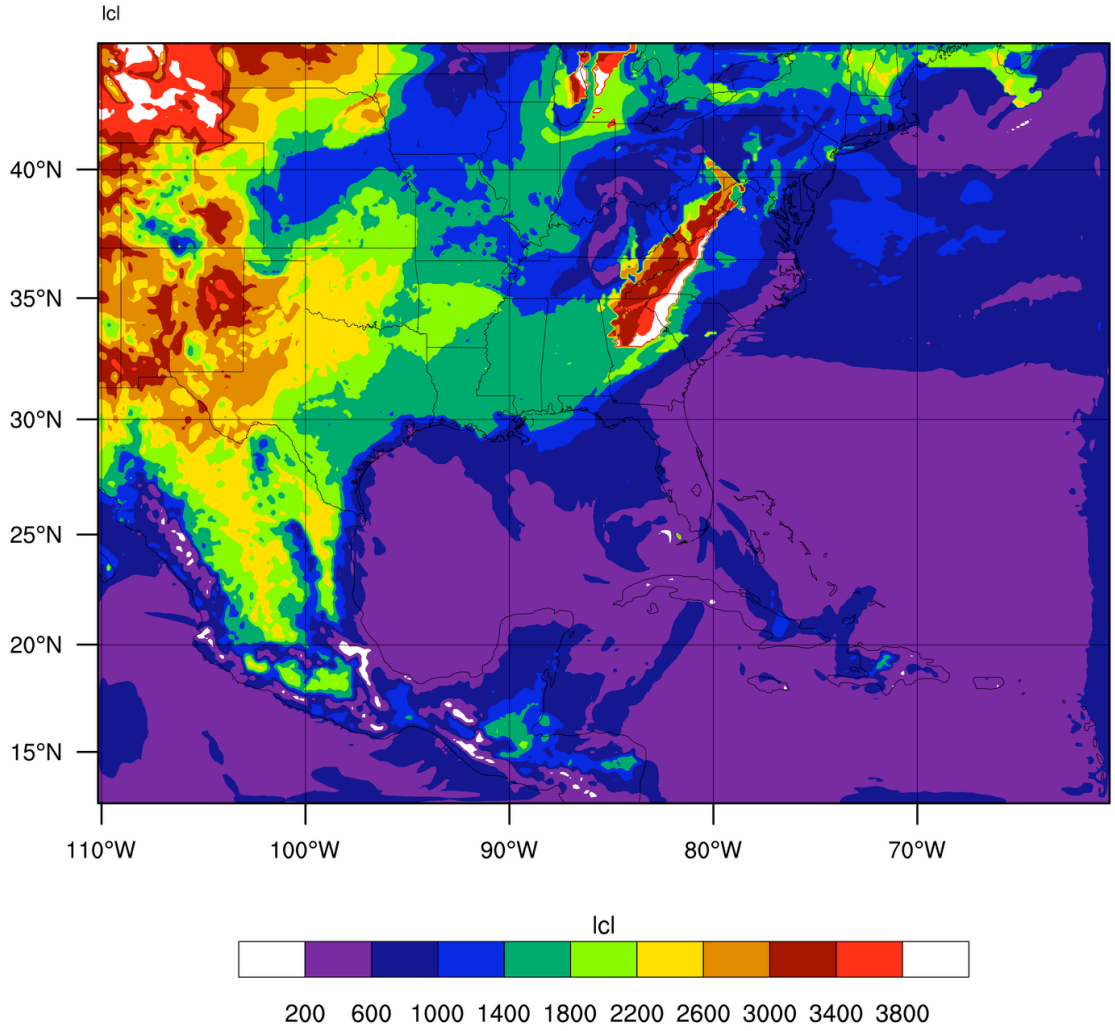
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



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

REAL-TIME WRF

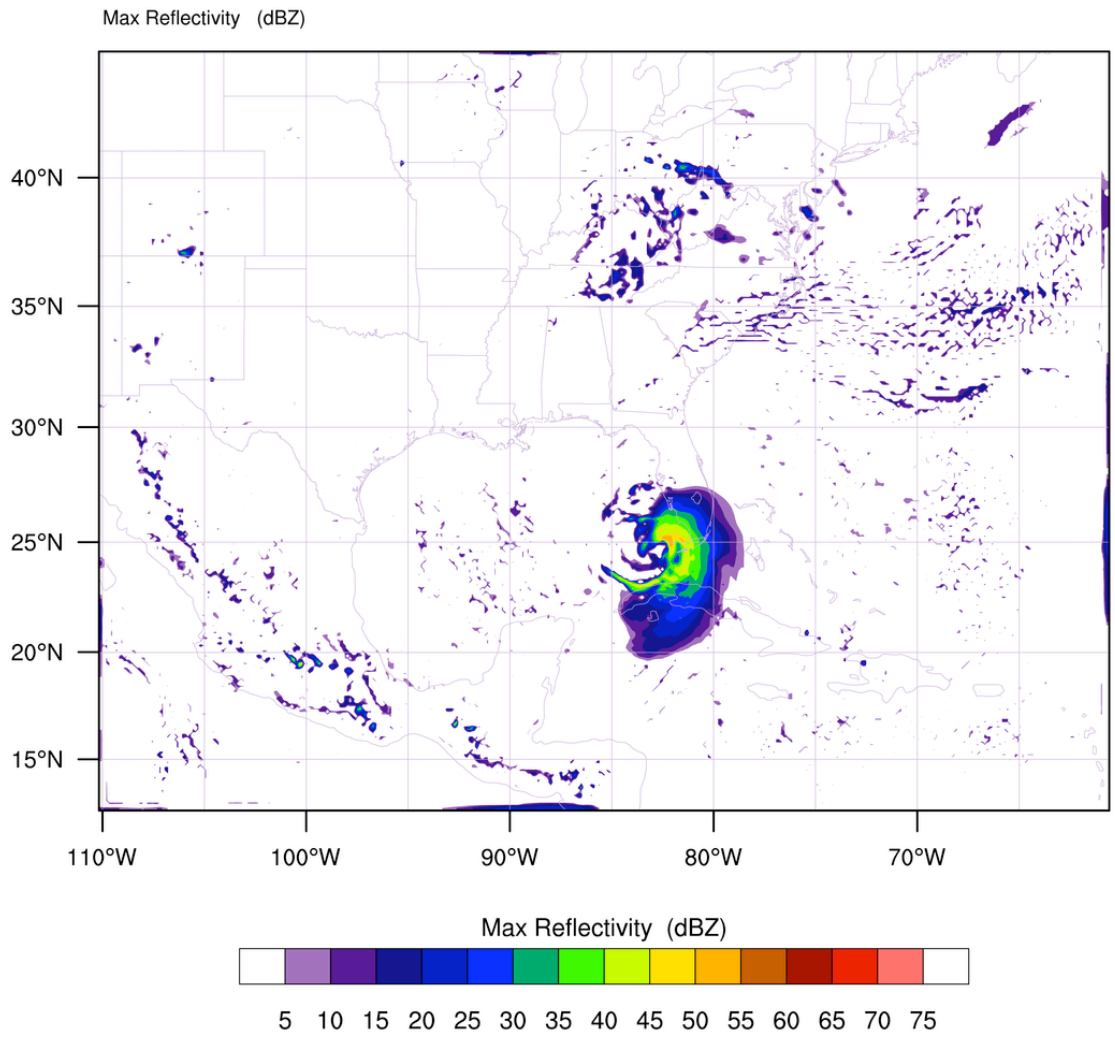
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



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

REAL-TIME WRF

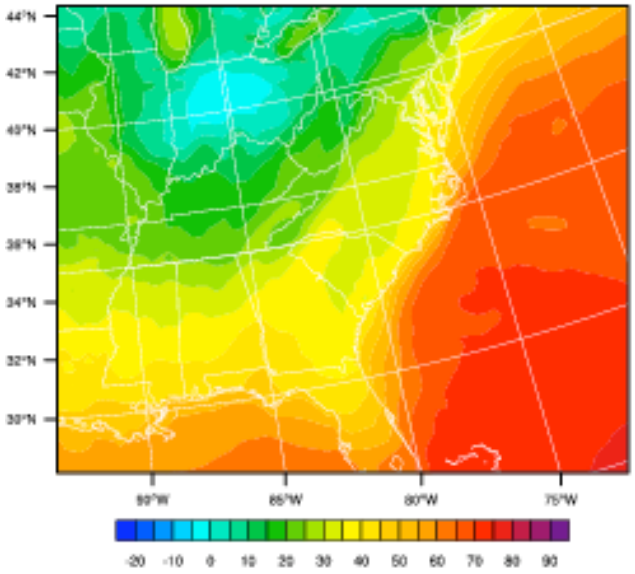
Init: 2005-08-26_00:00:00
Valid: 2005-08-27_00:00:00



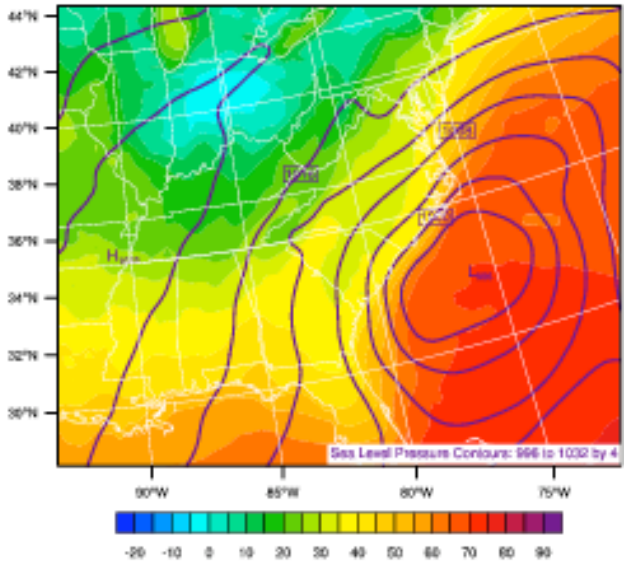
OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

PLOTS for : 2000-01-25_00:00:00

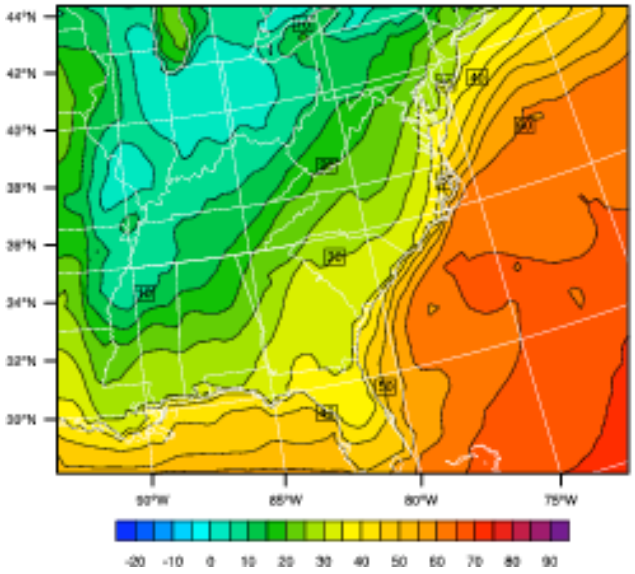
Surface Temperature (F)



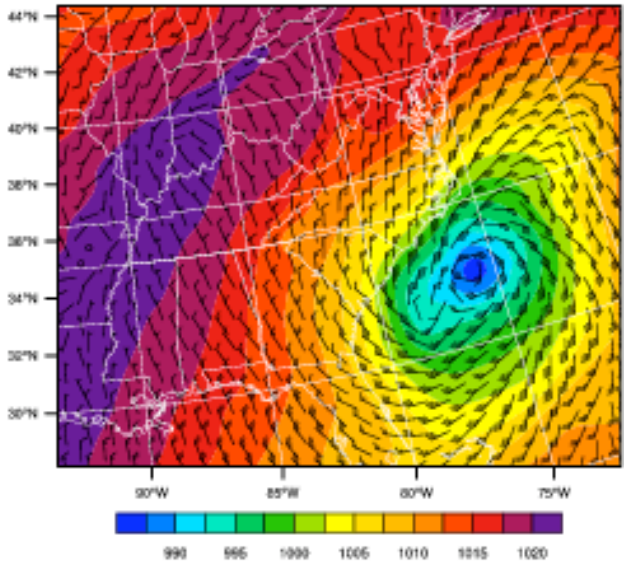
Surface Temperature (F)
Sea Level Pressure (hPa)

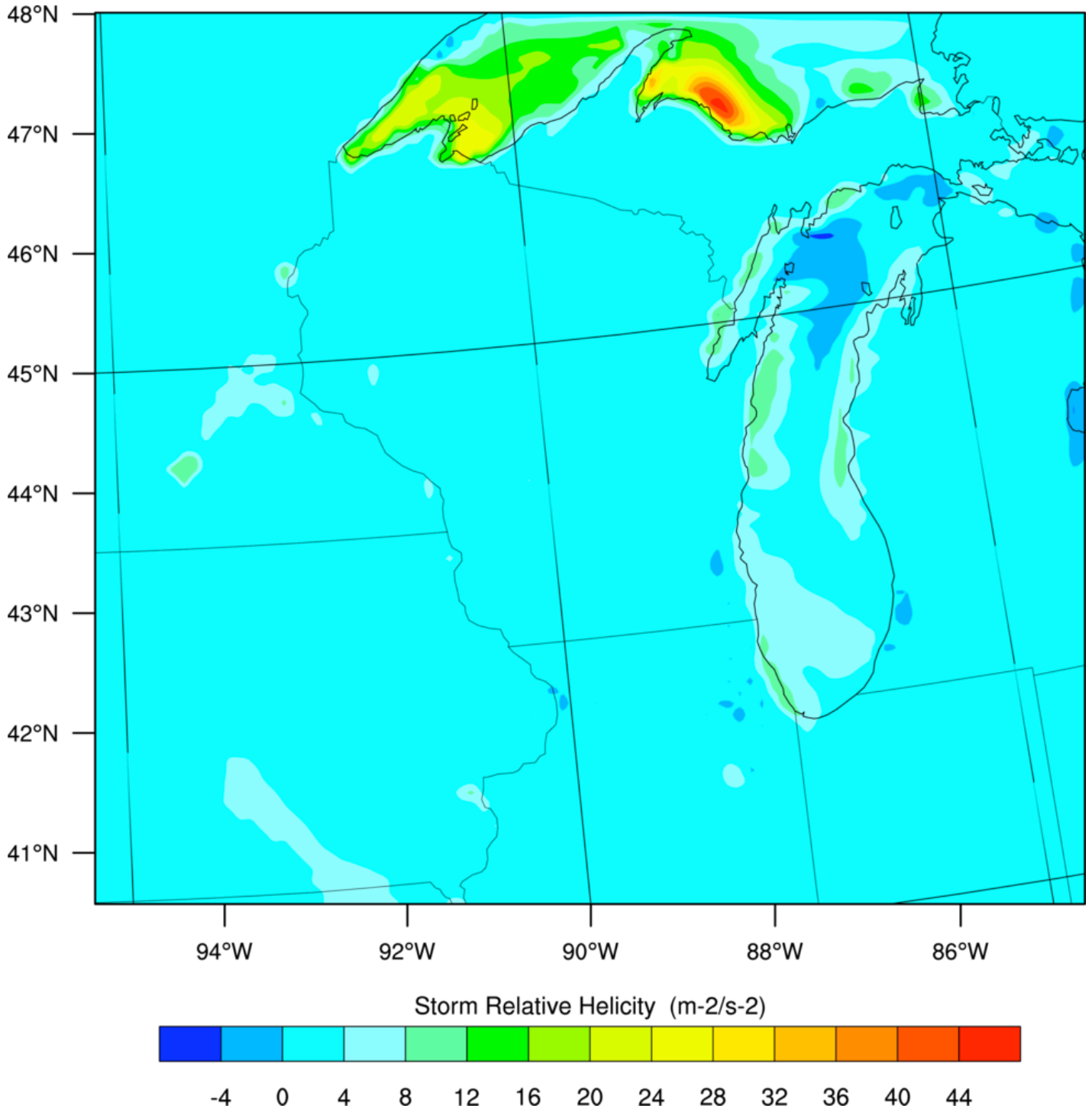


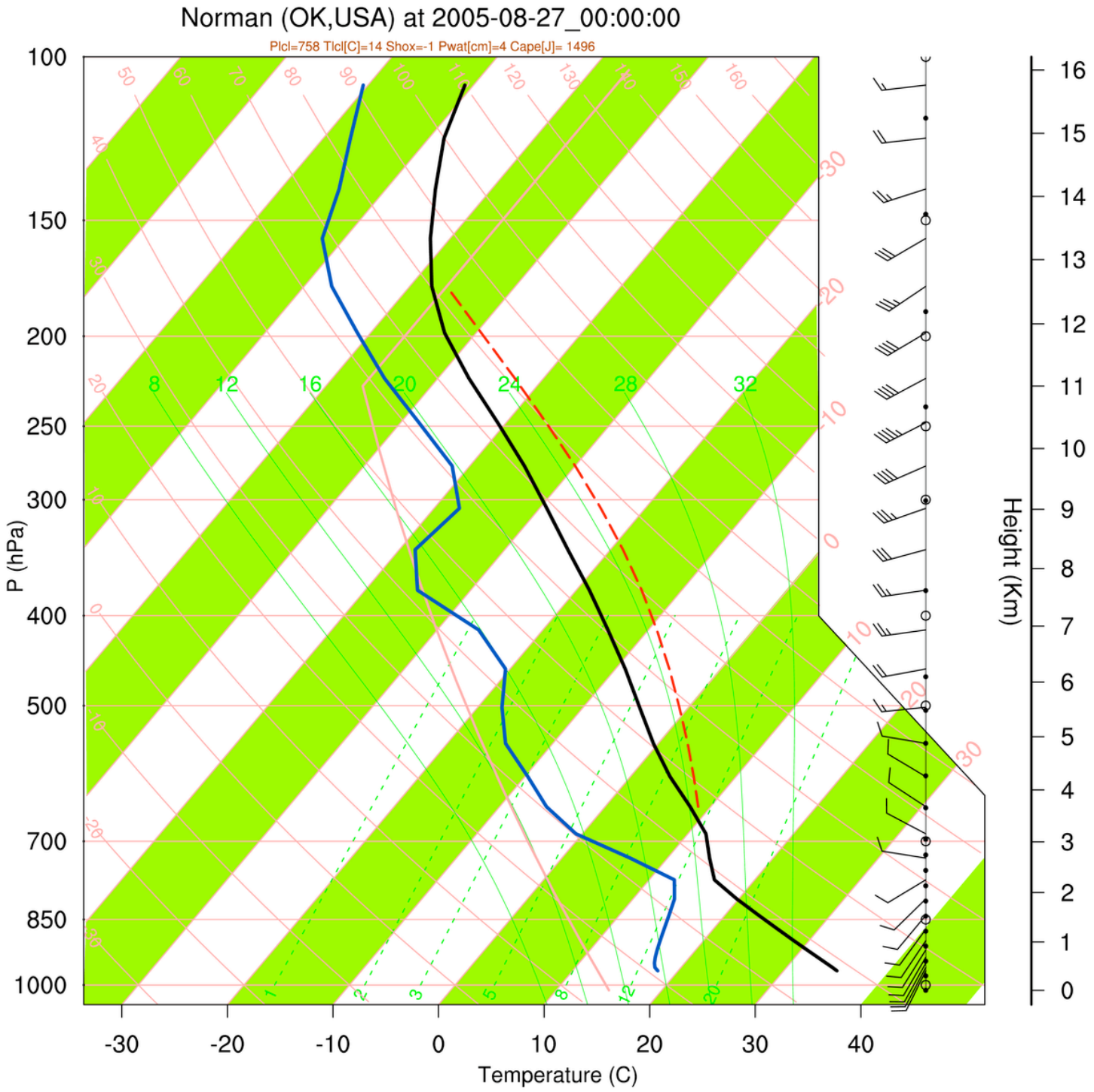
Surface Dew Point Temp (F)

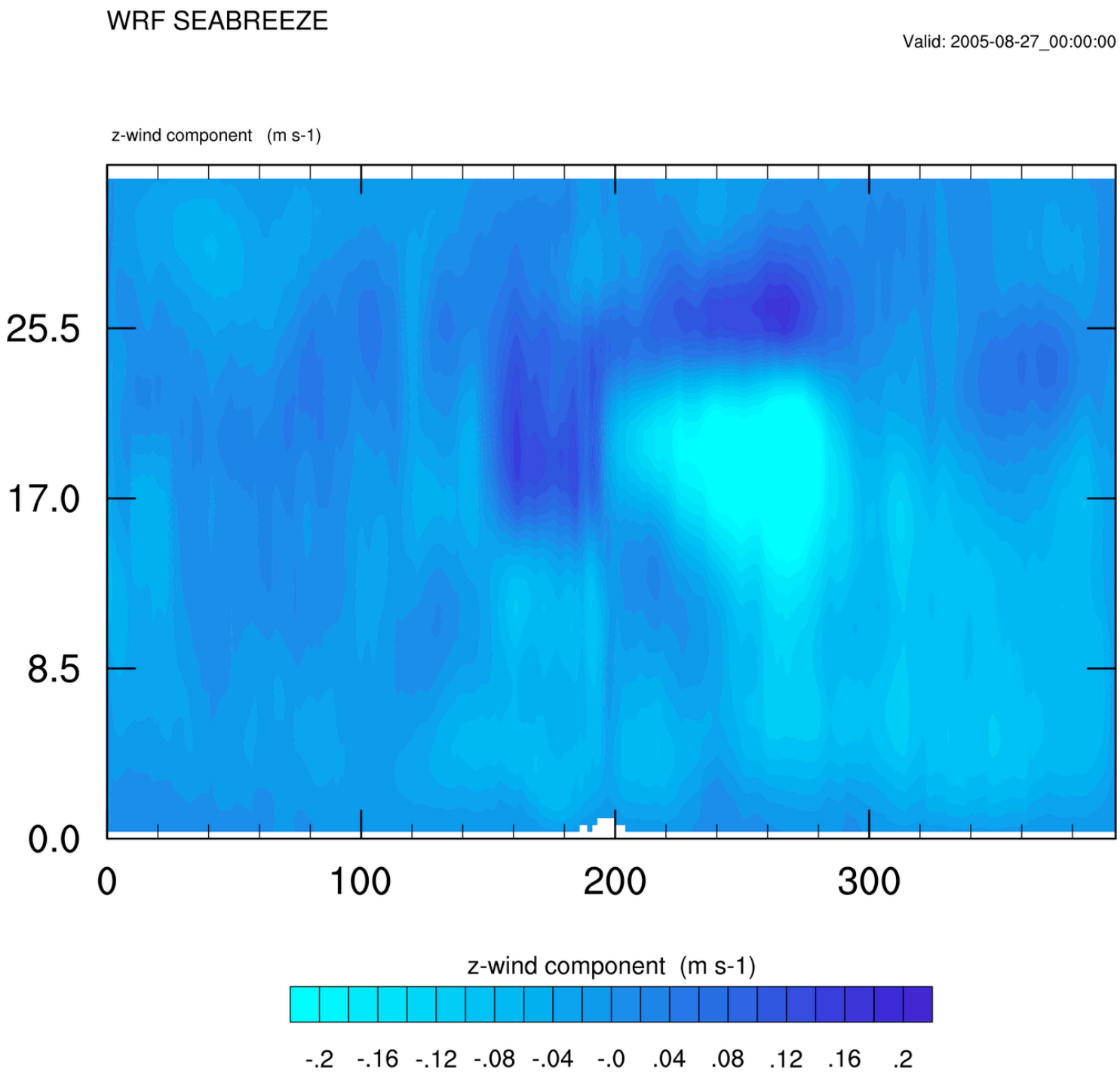


Sea Level Pressure (hPa)
Wind (kts)

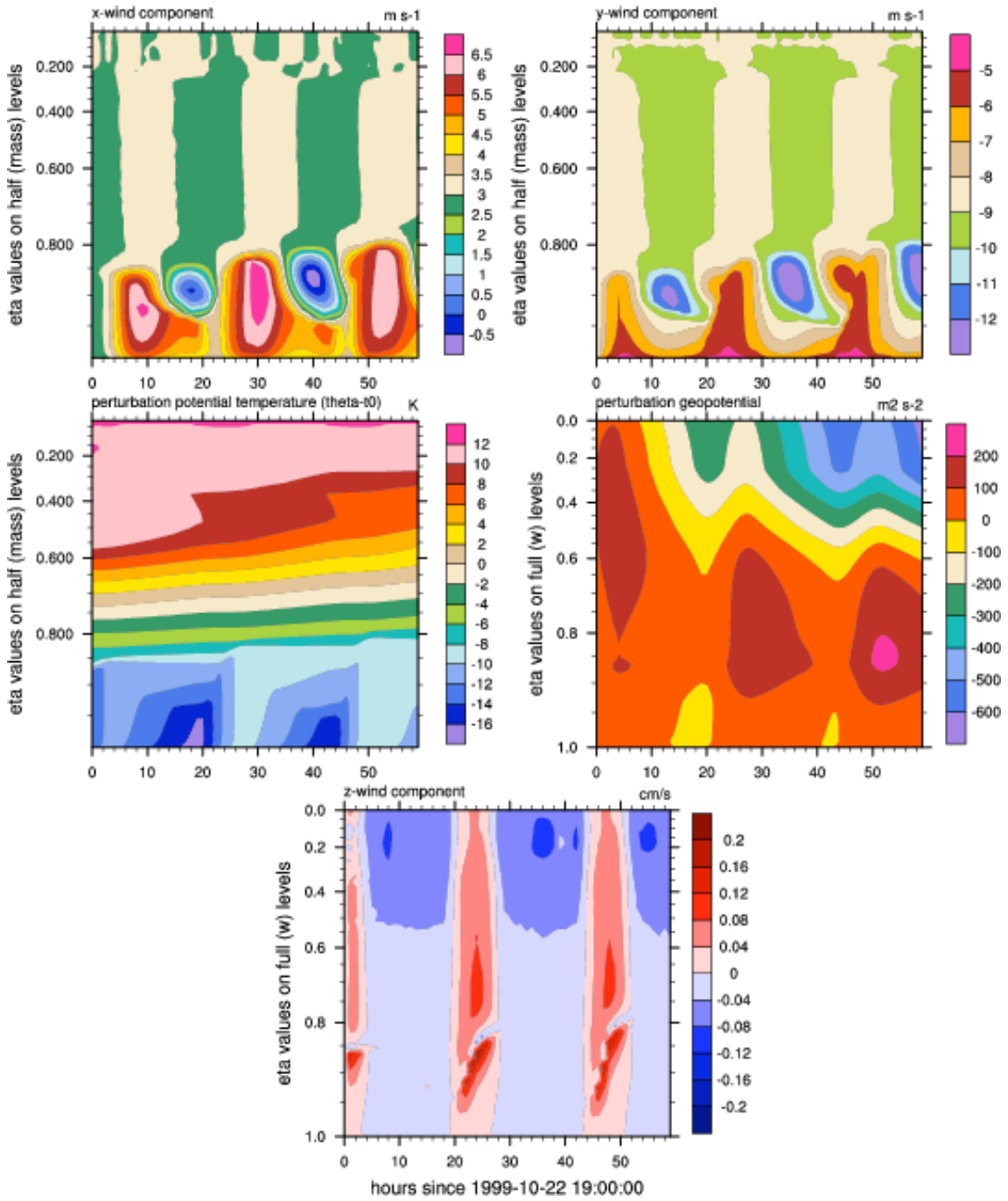



48°N
47°N
46°N
45°N
44°N
43°N
42°N
41°N
94°W
92°W
90°W
88°W
86°W
Storm Relative Helicity (m-2/s-2)
-4
0
4
8
12
16
20
24
28
32
36
40
44


Norman (OK,USA) at 2005-08-27_00:00:00
Plcl=758 Tlcl[C]=14 Shox=-1 Pwat[cm]=4 Cape[J]= 1496
P (hPa)
100
150
200
250
300
400
500
700
850
1000
-30
-20
-10
0
10
20
30
40
Temperature (C)
Height (Km)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


WRF SEABREEZE
Valid: 2005-08-27_00:00:00
z-wind component (m s-1)
25.5
17.0
8.5
0.0
0
100
200
300
z-wind component (m s-1)
-.2
-.16
-.12
-.08
-.04
-.0
.04
.08
.12
.16
.2

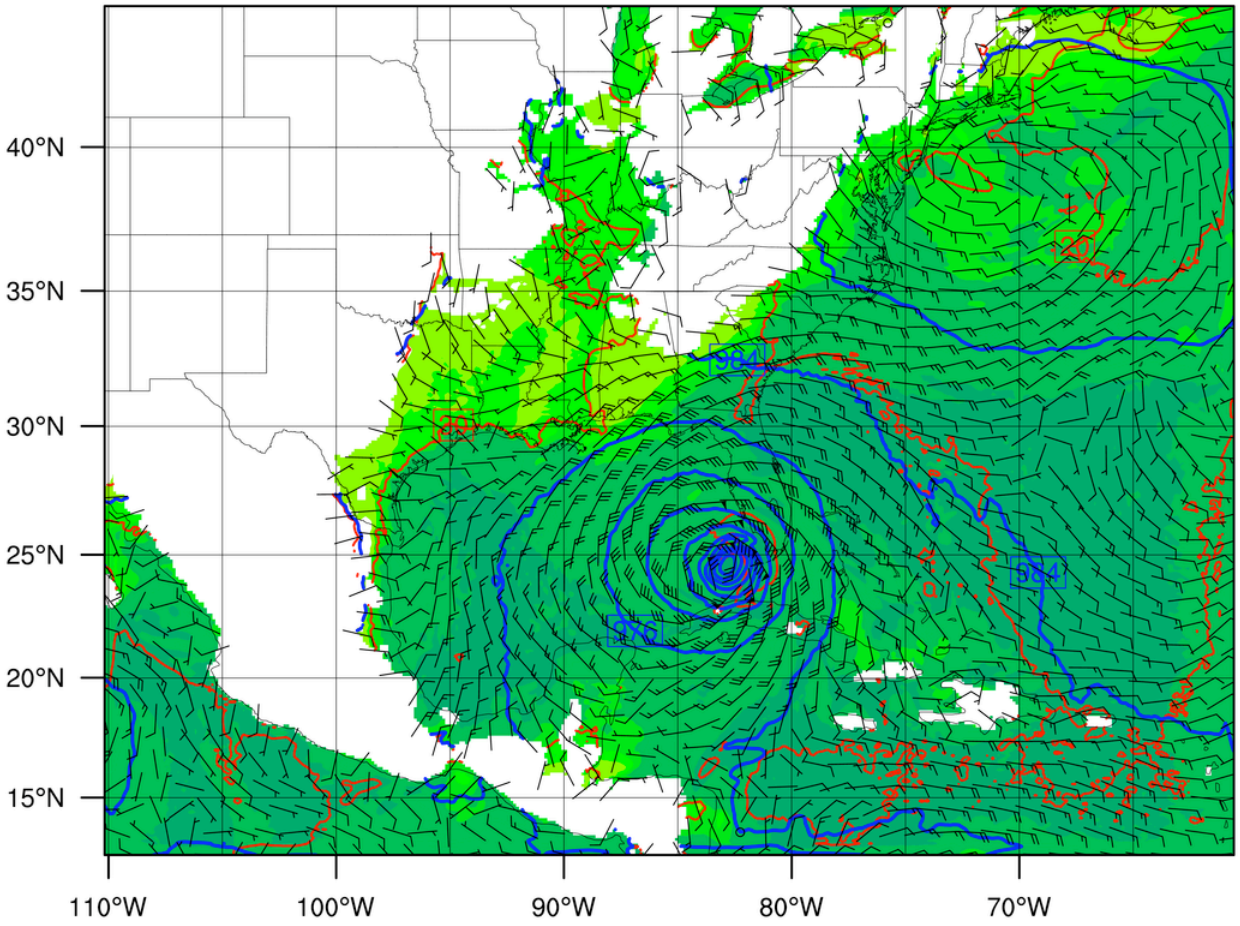
WRF-SCM: 37.60N 96.70W



REAL-TIME WRF

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

Relative Humidity (%) at 0.25 km
Temperature (C) at 0.25 km
Pressure (hPa) at 0.25 km
Wind (kts) at 0.25 km



Pressure Contours: 948 to 988 by 4

Temperature Contours: 10 to 45 by 5

Relative Humidity (%)

Other NCL visualizations

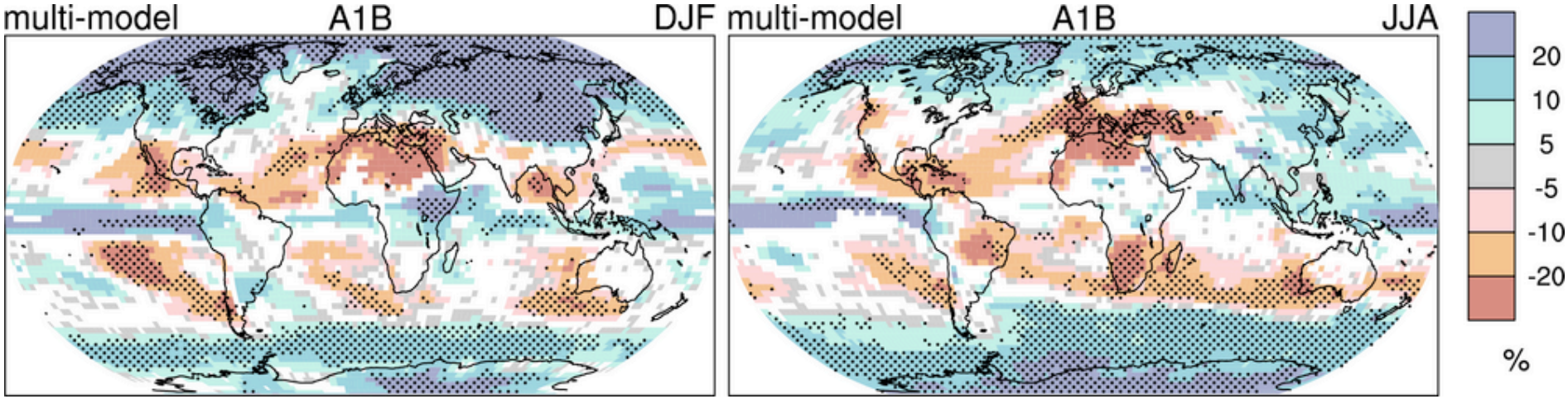
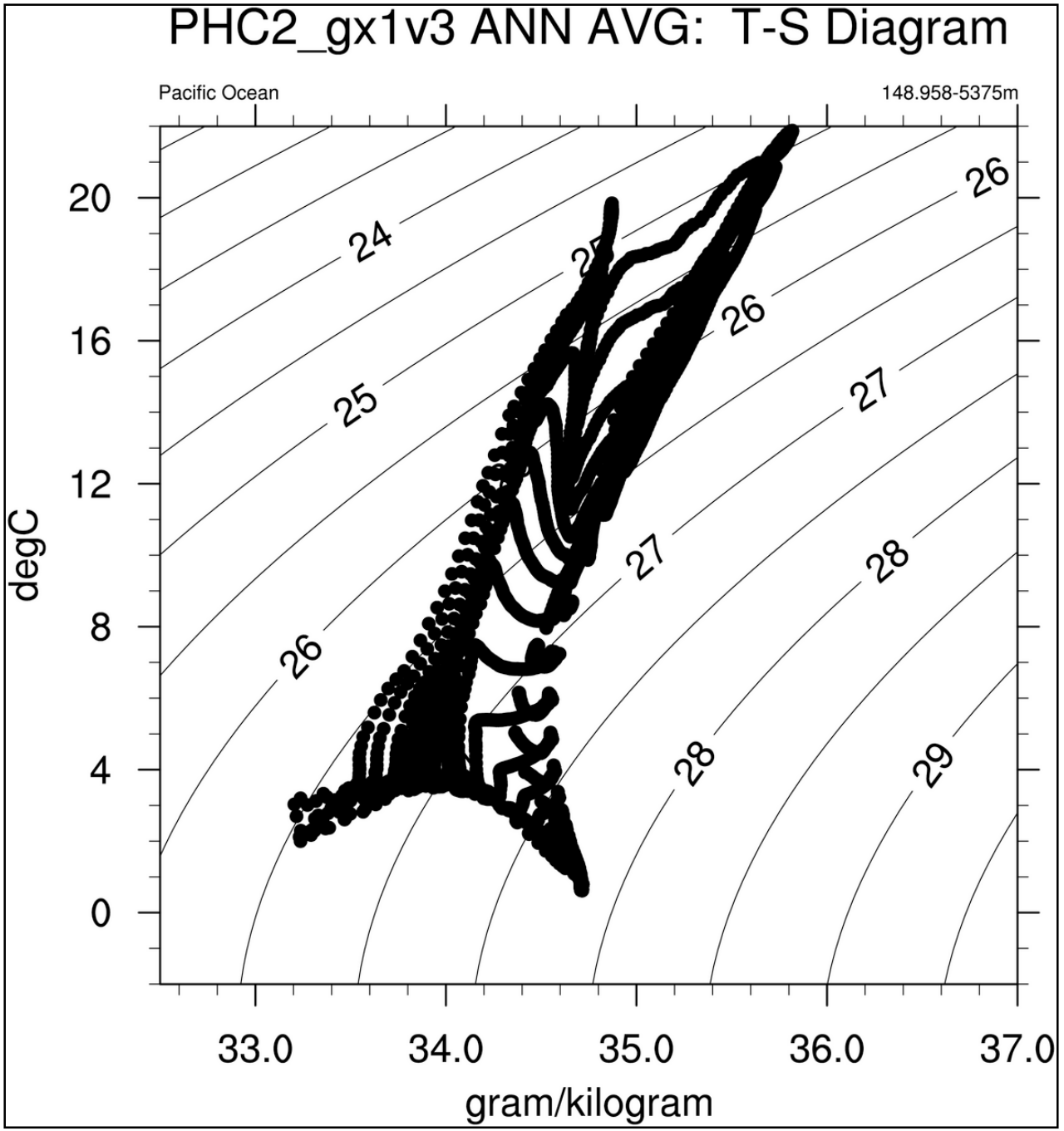
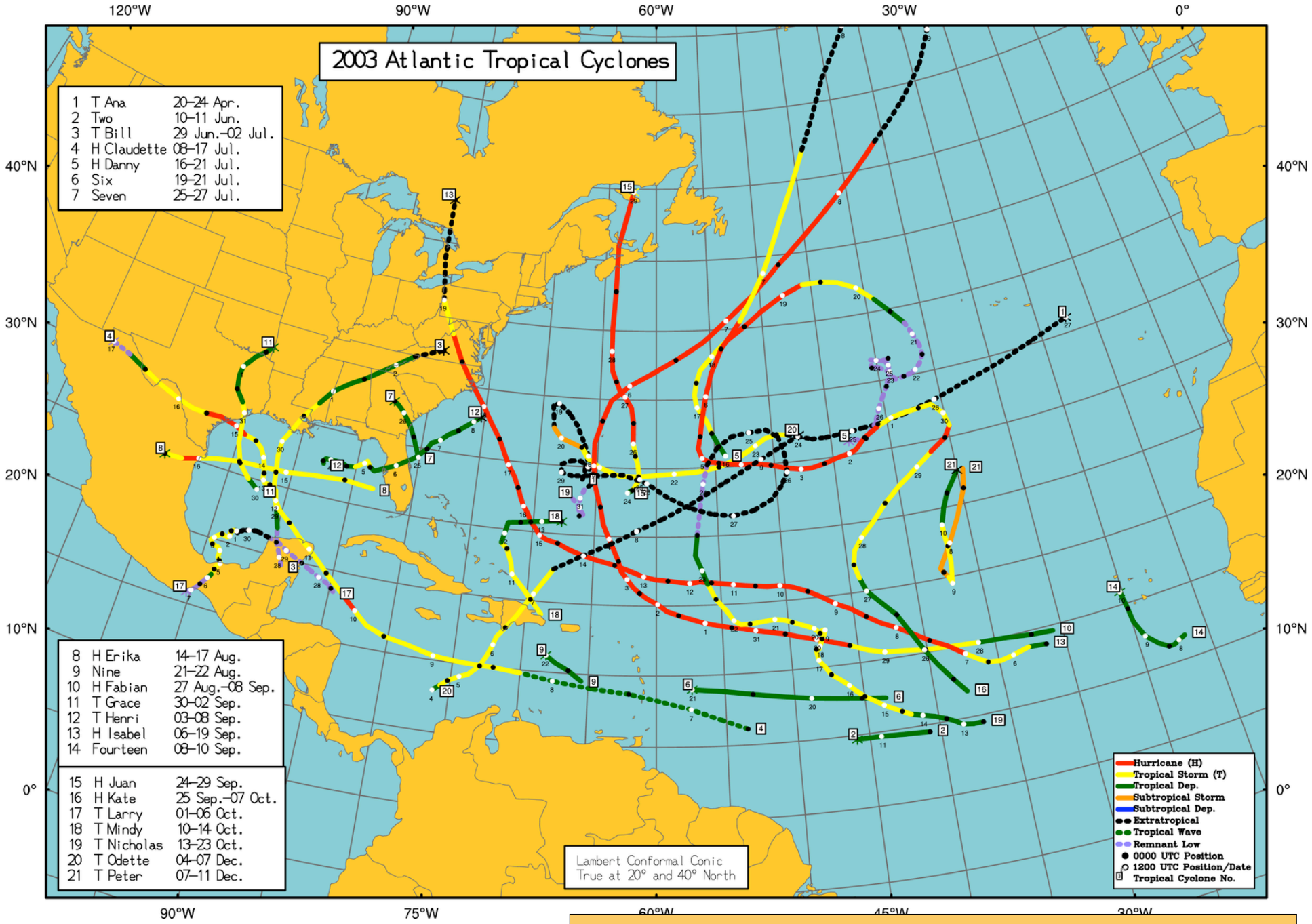


Image courtesy of Julie Arblaster
Bureau of Meteorology, University of Melbourne

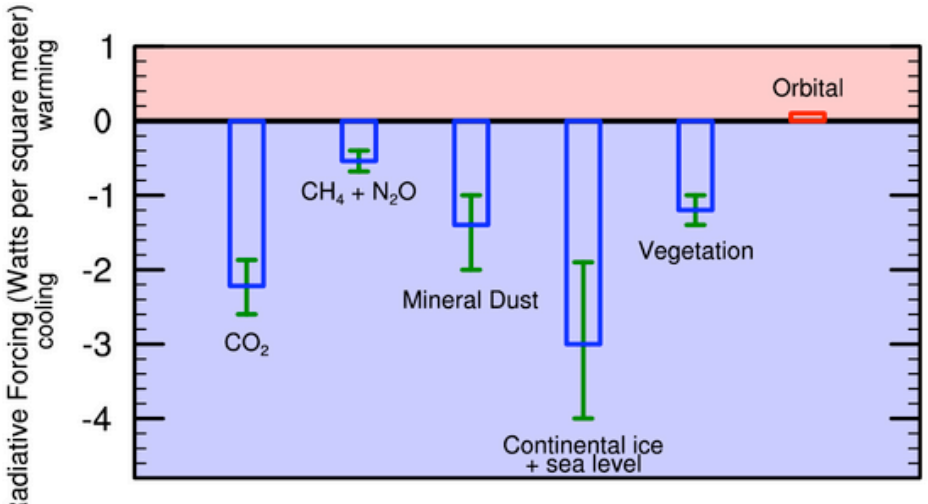


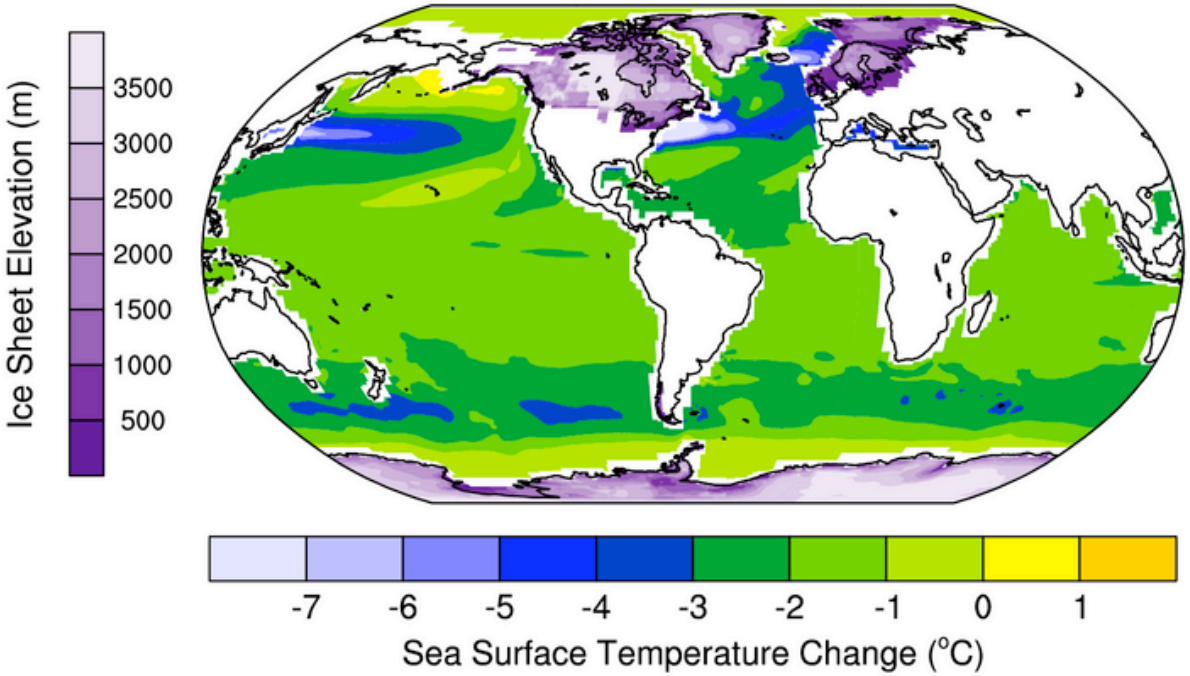
A **T-S diagram** is a graph showing the relationship between temperature and salinity as observed together at, for example, specified depths in a water column. Isopleths of constant density are often also drawn on the same diagram as a useful additional interpretation aid. In the ocean, certain T-S combinations are preferred, leading to the procedure of identification via the definition of water types and water masses and their distributions.

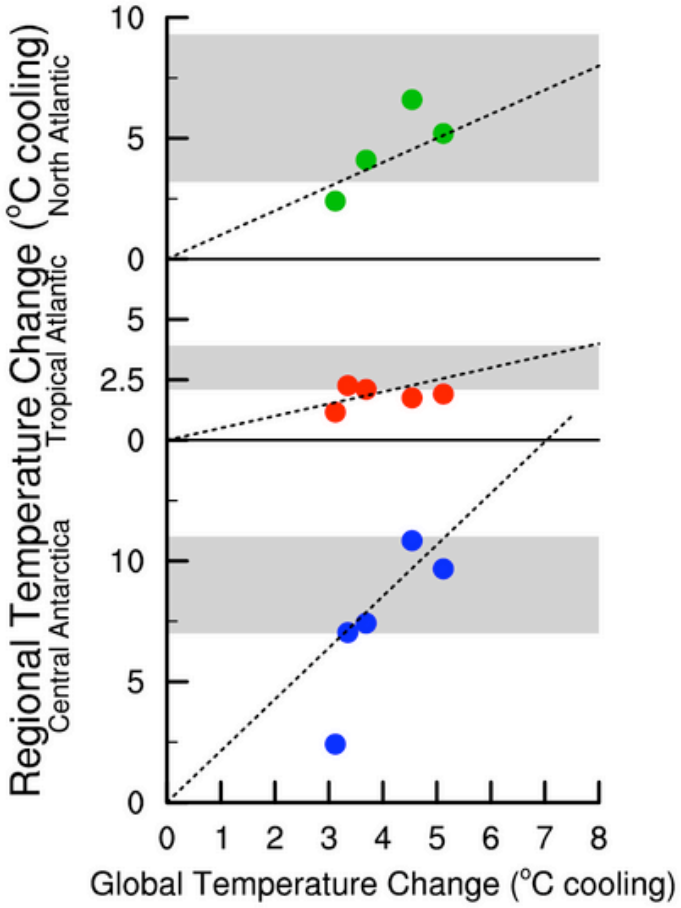
Image contributed by Christine Shields, NCAR/CGD.



Graphic by Jonathan Vigh, Colorado State University



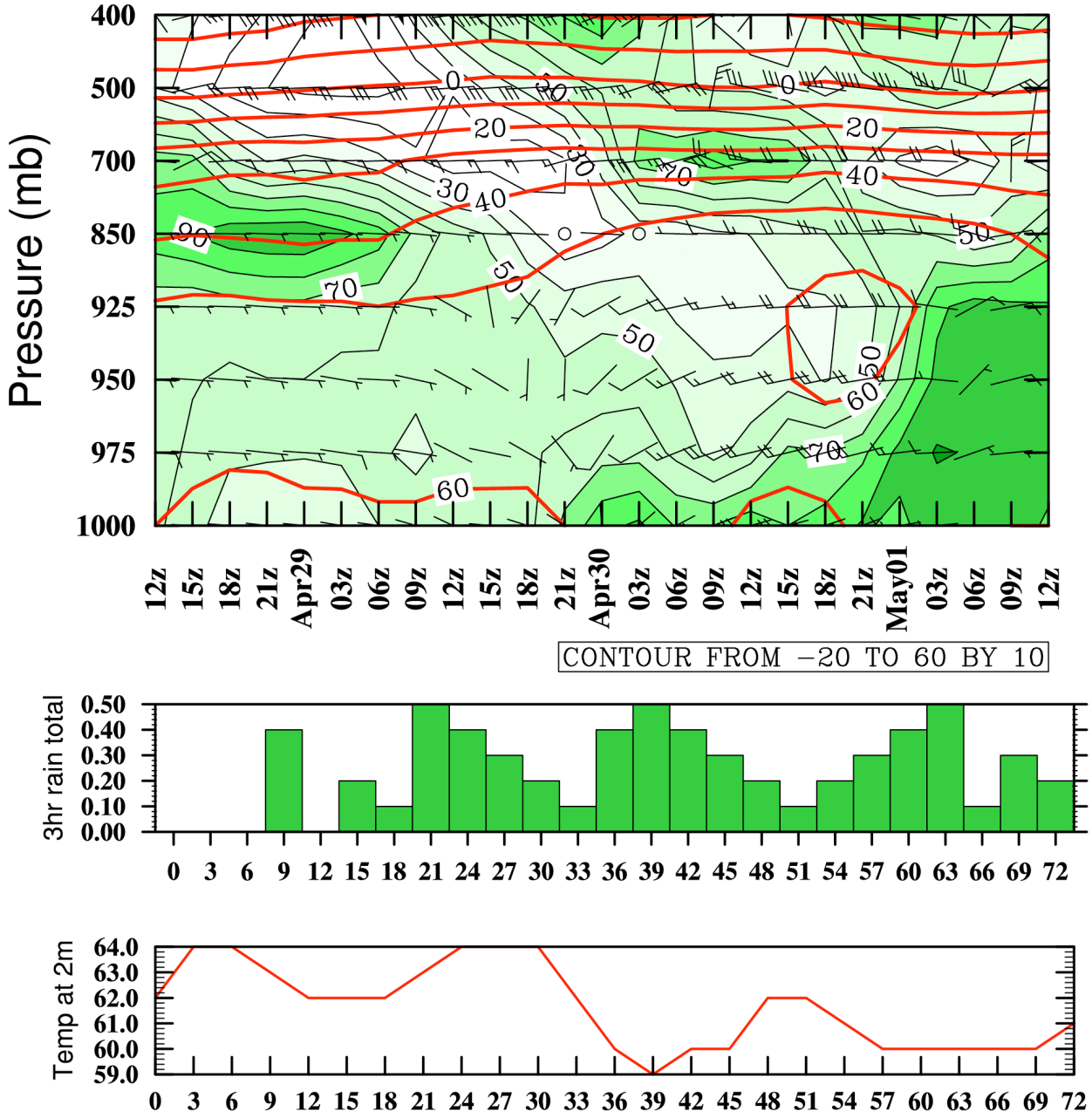


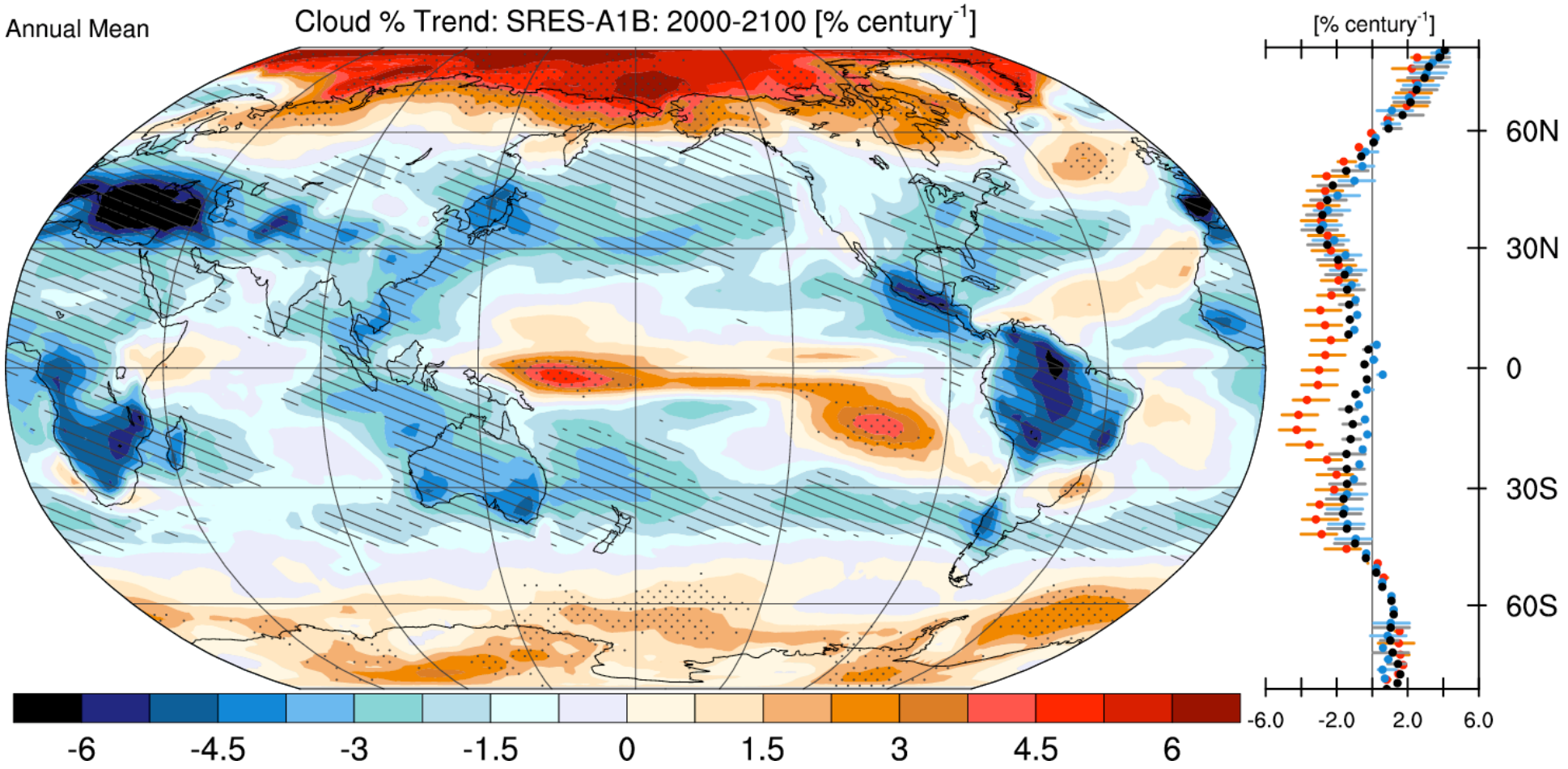


Based on a visualization of Adam Phillips

From John Ertl, FNMOC

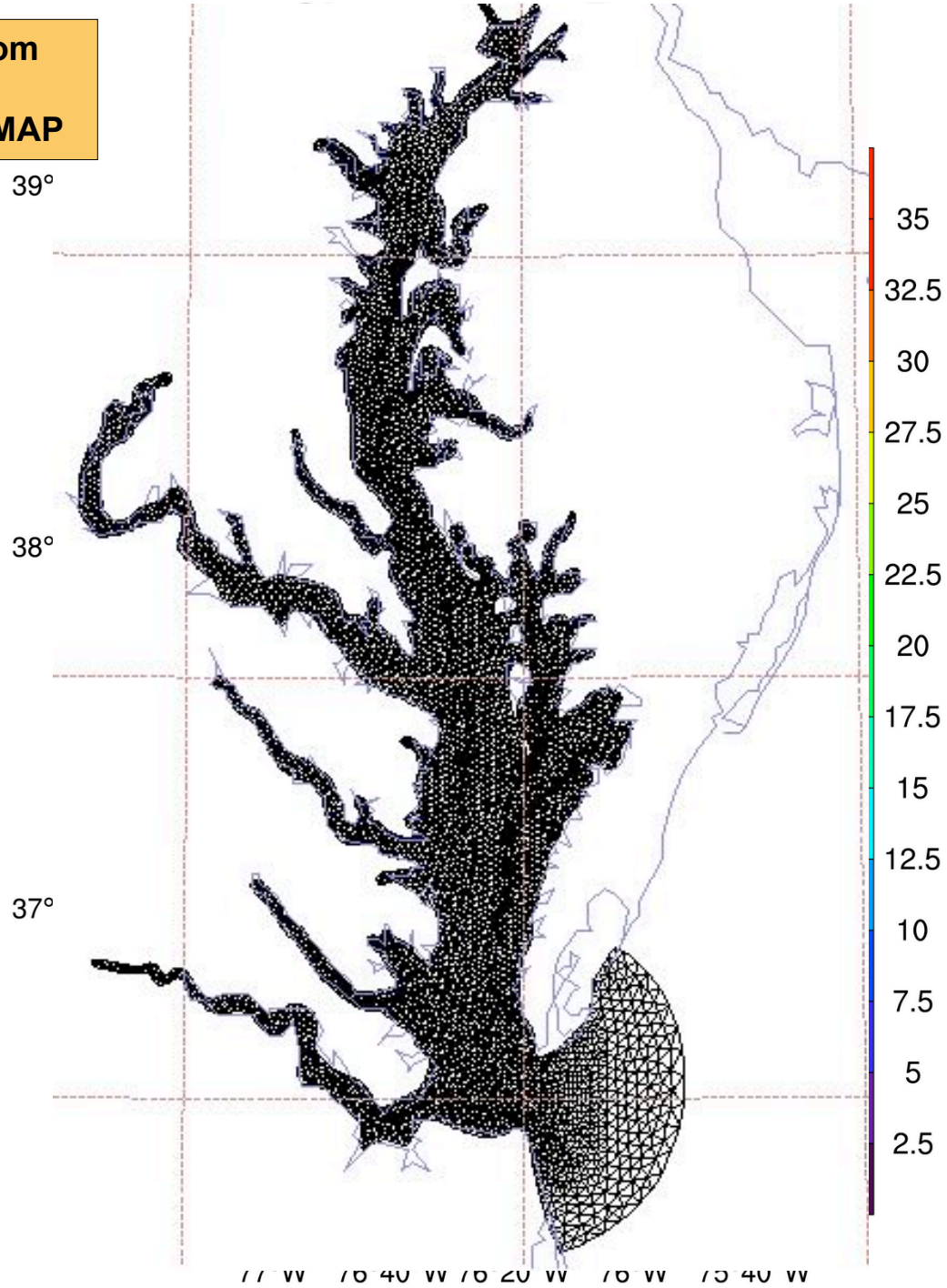
Meteogram for LGSA, 28/12Z



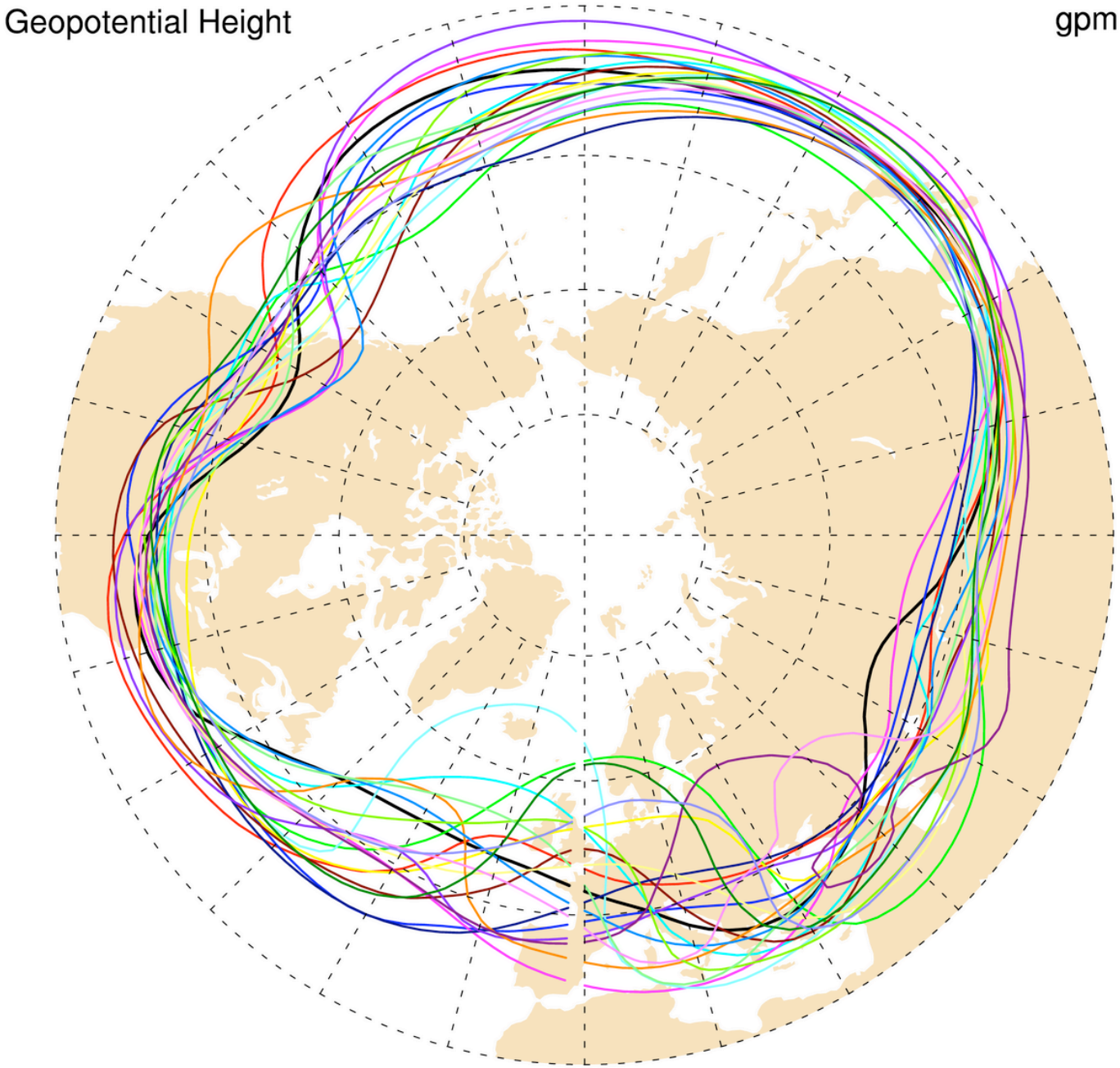


John Fasullo, NCAR/CGD

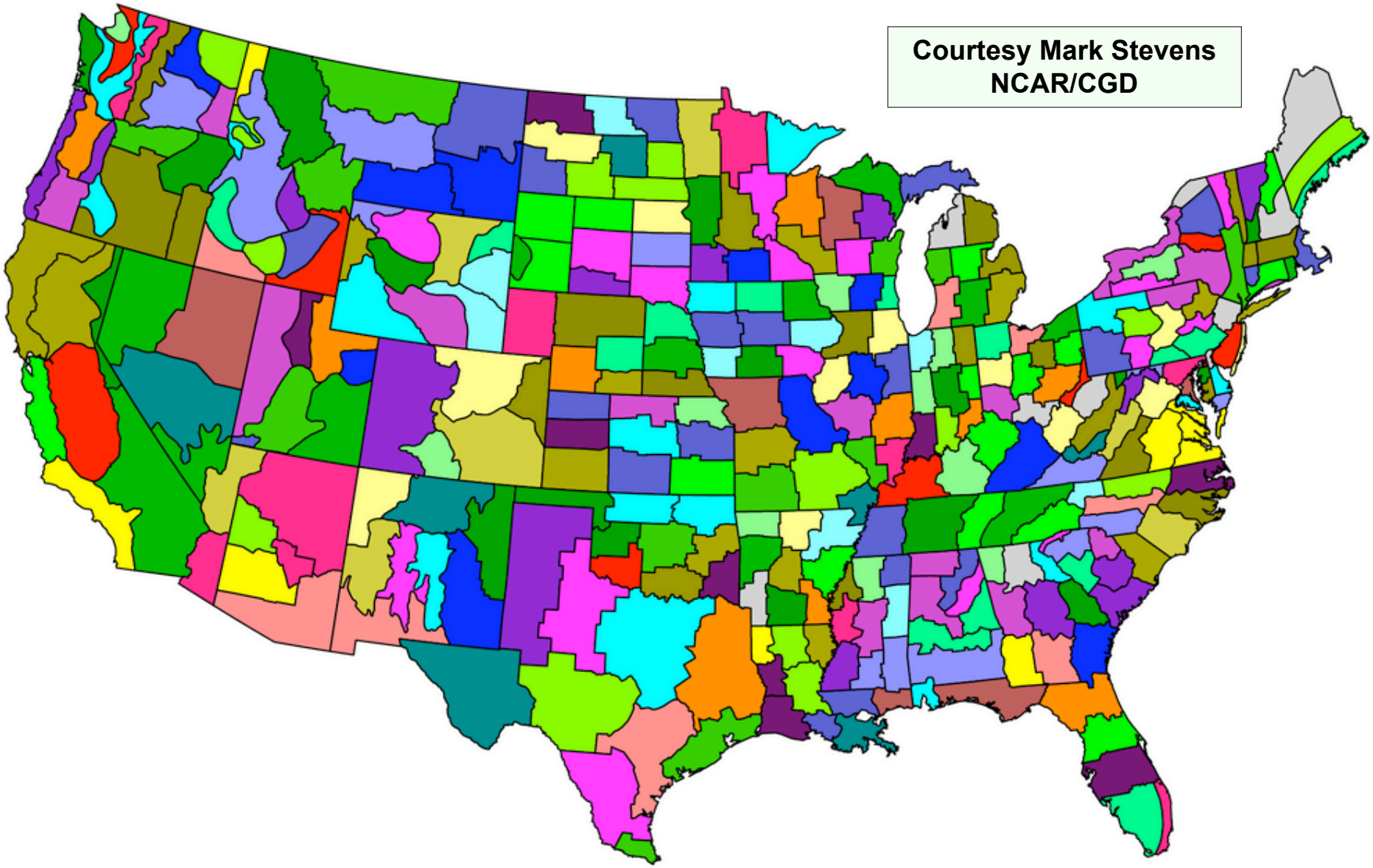
Triangular mesh from
Tom Gross
NOAA/NOS/CSDL/MMAP



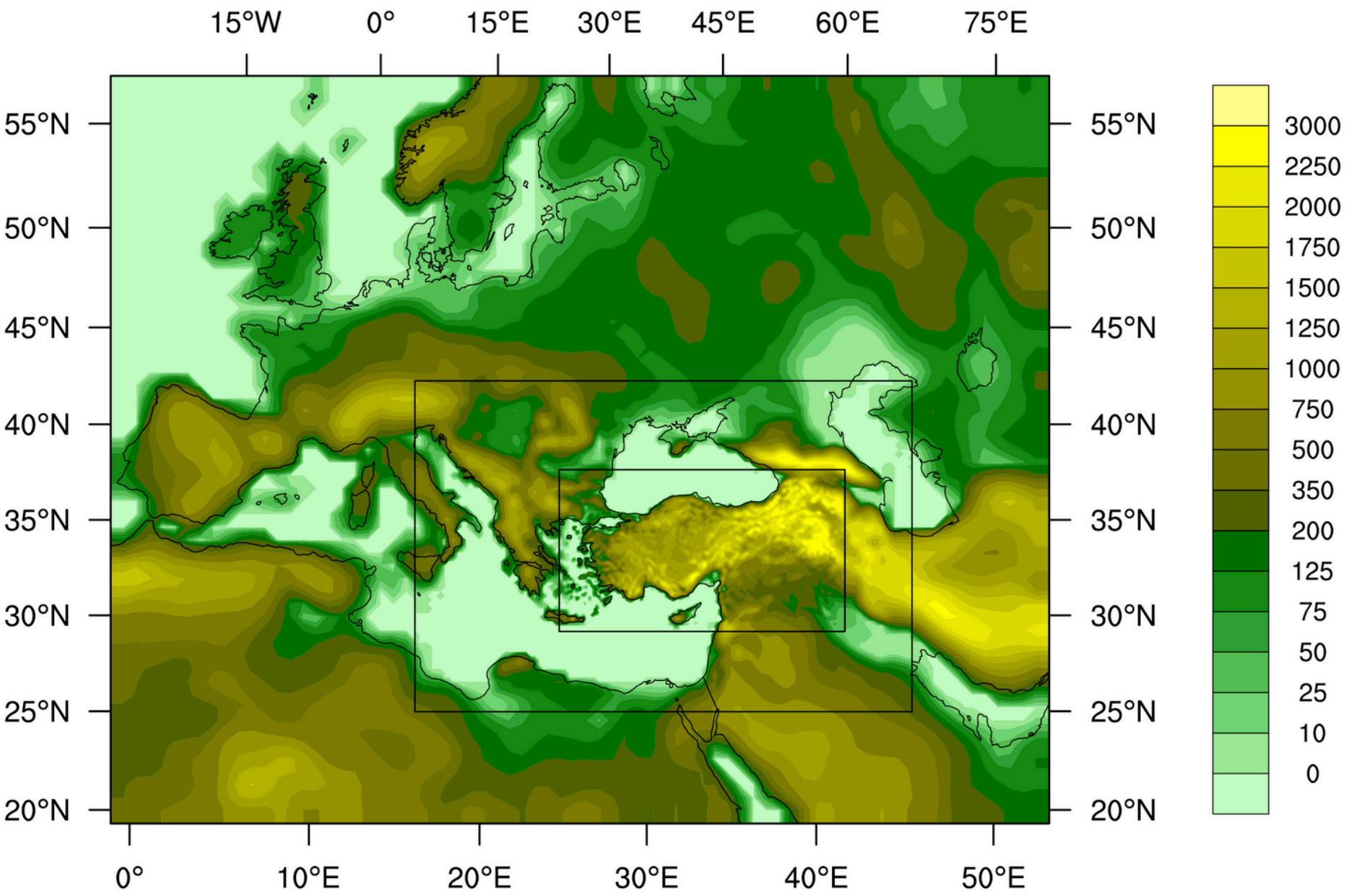
Spaghetti-style contours



Climate divisions



Climate divisions are built into NCL and PyNGL



Ufuk Turuncoglu, ITU - Turkey Climate Change Scenarios

- Overview
- NCL basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

To “run” an NCL script:

- [*Install NCL and set up environment, covered later*]
- [Make sure you have “~/.hluresfile”]
- Create a file using a UNIX editor that contains NCL script commands, say, “myfile.ncl”
- Run the file on the UNIX command line with:

```
ncl myfile.ncl
```

- View output or graphical file

```
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
```

begin/end are optional

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

Open the file

This is like doing an "ncdump –h"

Retrieves WRF variable

Use print/printVarSummary for debugging

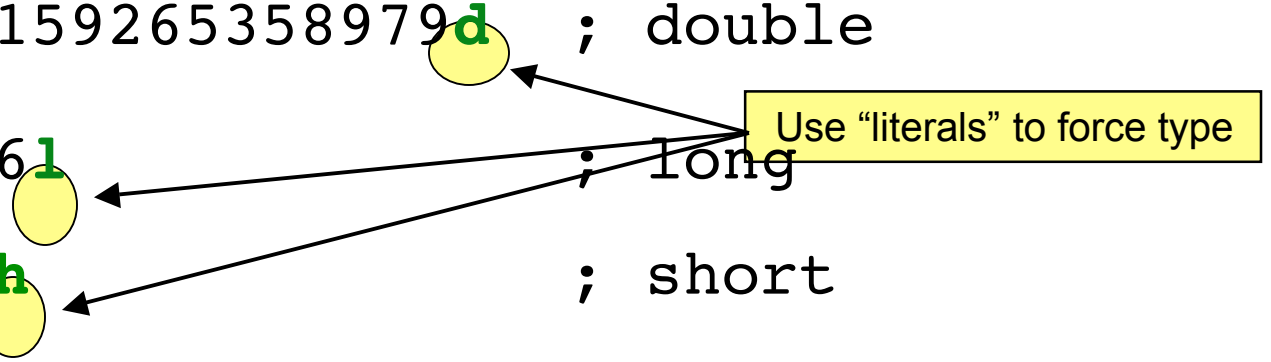
array arithmetic, like f90

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
```



New types introduced in NCL V5.2.0

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
s = "var_" + (j+1) + "_f"    ; s = "var_3_f"
```

Type conversions

```
; Can't change to "higher" type; use delete
ff = 1.5e20    ; float
ff = 1000      ; ok, still a float
ff = 1d36      ; error, "type mismatch"

delete(ff)
ff = 1d36      ; double
```

Old ones were "doubletofloat", "floattoint", etc.

Note about "tointeger" issue in WRFUserARW.ncl

```
ix = tofloat(dx)          ; 345.789
ix = tointeger(dx)        ; 345
```

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,ny,nz)
y1 = y(0,0,0)             ; first value
yn = y(nx-1,ny-1,nz-1)    ; last value
```

Array assignment: (/. . ./)

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

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

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

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

; Will be filled with -9999.
```

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)

  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
  xx = ind(pr.lt.500. .and. pr.gt.60.)
```

"where" is usually better than "ind"

Metadata

- Metadata is information about a variable or a file.
- In NetCDF-land, metadata consists of:
  - Attributes (units, history, grid type, long name)
  - Named dimensions
  - Coordinate arrays (must be one-dimensional)
- WRF-ARW data doesn't normally have traditional 1D coordinate arrays, WRF coordinates are generally 2D (XLAT, XLONG)
- The "_FillValue" attribute is a special one indicating a variable's missing value. "missing_value" not recognized
- When you do an "ncdump -h" on a NetCDF file, you see all the metadata
- NCL variables are based on this metadata model.

Missing values (_FillValue attribute)

- "_FillValue" is a NetCDF *and* NCL reserved attribute
- Most NCL functions ignore _FillValue:

```
x              = (/1,2,3,-999,5/) ; no msg val yet
xavg           = avg(x)           ; = -197.6
x@_FillValue   = -999             ; now has a msg val
xavg           = avg(x)           ;(1+2+3+5)/4 = 2.75
```

- Must be same as type of variable

Use '@' to reference attributes

- "missing_value" attribute has **no** special status to NCL.
  If "T" has "missing_value" attribute and no "_FillValue":

```
T@_FillValue = T@missing_value
```

- Best not to use zero as a _FillValue
- Sample NCL (netCDF) default missing values:

integer	-999
float	-999.
double	-9999.
string	"missing"

short	-99
byte	0xff
character	0x00
logical	Missing

"print" / "printVarSummary" will print _FillValue value.

"print" is very verbose

Missing value functions

- Use any, all, and ismissing functions to query a variable for missing values:

```
if (.not.any(ismissing(T))) then
  do something
end if
if (all(ismissing(T))) then
  do something
end if
```

- Use num & ismissing to count missing values:

```
nmsg = num(ismissing(T))
```

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

(0)     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 any time
- Highest "type" will be assigned to variable on left of "="

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

  A = data_DJF - data_JJA       ; A will 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 reorder, reshape, reverse

```
; Reshaping an array

  t1D = ndtooned(T)                    ; Convert to 1D array
  t2D = onedtond(t1D, (/N,M/) )        ; Convert to N x M array
```

```
; Reordering an array
```

Requires named dimensions be present

```
; Let T(time,lat,lon)
  t = T(lat|:,lon|:,time|:)    ; Can't assign to same var
```

```
; Reversing dimensions of an array

; Let T(lev,lat,lon)
  T = T(::-1,:,:)      ; Will reverse coordinate array too
```

Functions for manipulating arrays

http://www.ncl.ucar.edu/Document/Functions/array_manip.shtml

Array Subscripting

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

http://www.ncl.ucar.edu/Document/Manuals/Ref_Manual/NclVariables.shtml#Subscripts

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(0:0,:,::5)             ; 1 x nlat x nlon/5
  t = T(:1,45:45,10:20)        ; 2 x 1 x 21
```

NCL syntax characters

;	comment (on line by itself, or at end of line)
@	reference/create attributes
!	reference/create named dimensions
&	reference/create coordinate variables
{…}	coordinate subscripting
$…$	enclose strings when (im/ex)port variables via addfile
(/…/)	array constructor characters
:	array syntax
|	separator for named dimensions
\	continuation character [to span multiple lines]
::	syntax for external shared objects (fortran/C)
->	use to (im/ex)port variables via addfile function

- Overview
- NCL basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

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

```
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)

more global attrs

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

Using “ncl_filedump”

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

```
ncl_filedump wrfout_d01_2005-08-27_00:00:00.nc

ncl_filedump –V RAINC wrfout_d01_2005-08-27_00:00:00.nc
```

Back to NCL scripts: Two ways to read a variable off a file

- Use "->" syntax

- Use "wrf_user_getvar" function
  - Developed to make it easier to get derived variables
  - It is an NCL script function, so must load "WRFUserARW.ncl" script
  - You can modify this script (more later)
  - 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
```

```
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
```

printVarSummary(u) results

named dimensions

no coordinate arrays

variable attributes

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(max(slp))         ; Print maximum of slp
print(typeof(slp))      ; Print type of slp
print(getvaratts(slp)) ; Print attributes of slp

; Can assign to variables
dims    = dimsizes(slp)
slp_min = min(slp)
slp_max = max(slp)
attrs   = getvaratts(slp)
slp_avg = avg(slp)
```

Most of above info is printed as part of printVarSummary procedure

Creating a new variable & adding attributes

```
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")
td2 = wrf_user_getvar(f,"td2",0)   ; Units are "C"

td_f = 1.8 * td2 + 32.    ; Can operate on whole array
td_f@units        = "F"   ; Add some attributes
td_f@description = "Surface Dew Point Temp"

; To preserve metadata
td_f = td2 ; Easy way to copy metadata, can be expensive
td_f = 1.8 * td2 + 32
td_f@description = "Surface Dew Point Temperature"
td_f@units        = "F"
printVarSummary(td_f)

; To write new variable to an existing file
f = addfile("wrfout_d01_2005-08-27_00:00:00.nc","w")
. . .
f->td_f = td_f    ; Write "td_f" to same file
```

- Overview
- NCL basics
- File input/output
- **Data Analysis**
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

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, uses some built-in functions

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

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

http://www.ncl.ucar.edu/Document/Functions/wrf.shtml

WRF-NCL built-in functions

Can use NCL built-in function, 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
geopt/geopotential Geopotential
p/pres/pressure Pressure ***rh/rh2*** Relative Humidity
slp Sea Level Pressure ***td/td2*** Dew Point Temperature ***tc/tk*** Temperature
th/theta Potential Temperature
ua/va/wa Wind on mass points
uvmet/uvmet10 U/V components of wind rotated to earth coords
z/height Height

http://www.ncl.ucar.edu/Document/Functions/WRF_arw/

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 (*varin*, *unstagDim*)
  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, opts)
```

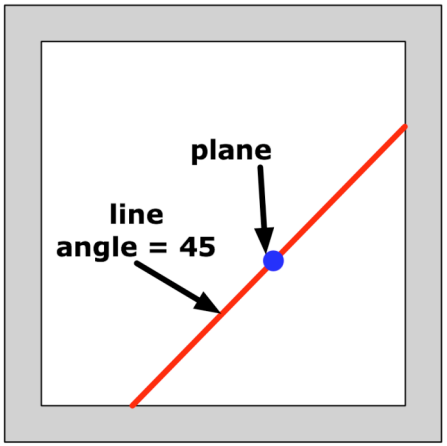
Other WRF-NCL "*WRFUserARW.ncl*" functions

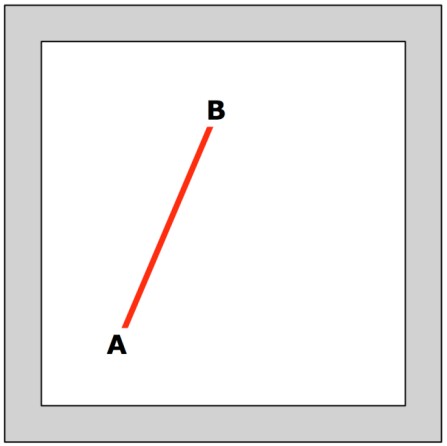
- wrf_user_intrp3d
  Interpolate horizontally to a given pressure or height level
  Interpolate vertically *(pressure/height),* along a given line

```
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 ij

```
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.

res@returnInt - Default is True
If set to False, the return values will be real.
(wrf_user_ll_to_ij *only)*

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:

```
function wrf_user_getvar(nc_file:file,
varin[*]:string, time:integer)
```

- 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:

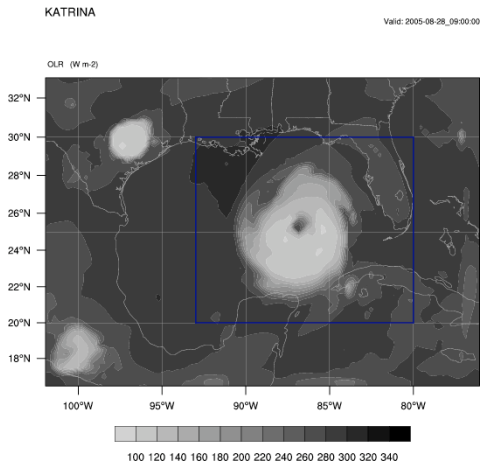
```
load "./WRFUserARW.ncl"
xxx = wrf_user_getvar(f,"XXX",0)
```

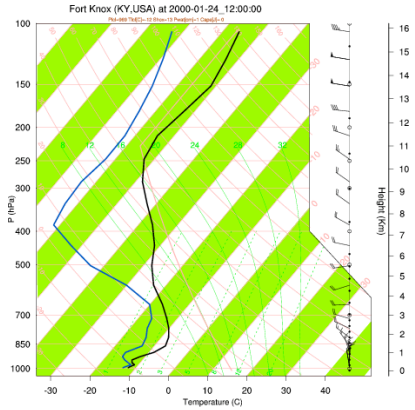
2. Remove all but the modified "wrf_user_getvar" function from your copy, rename the function ("wrf_user_getvar2"), and rename the file ("my_new_script.ncl"). To use the new function, you need to load the above script and your new script:

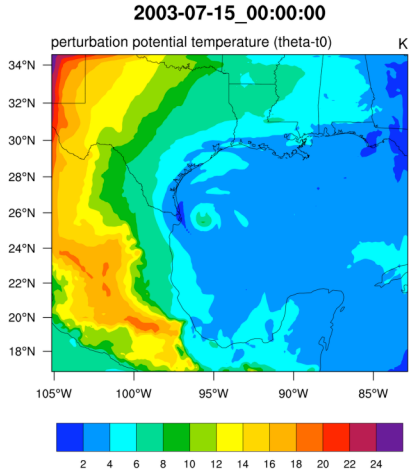
```
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
load "./my_new_script.ncl"

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

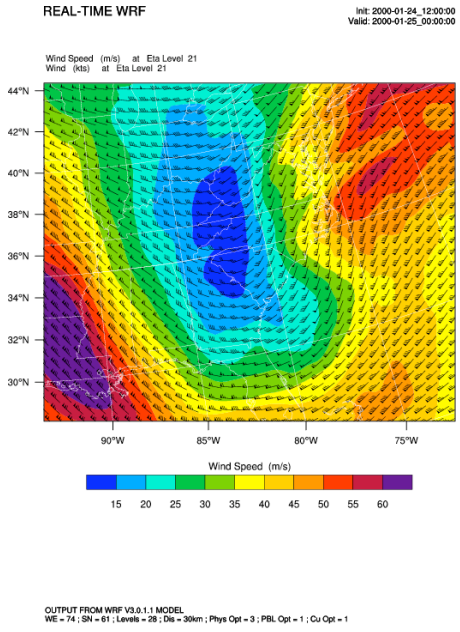
- Overview
- NCL basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

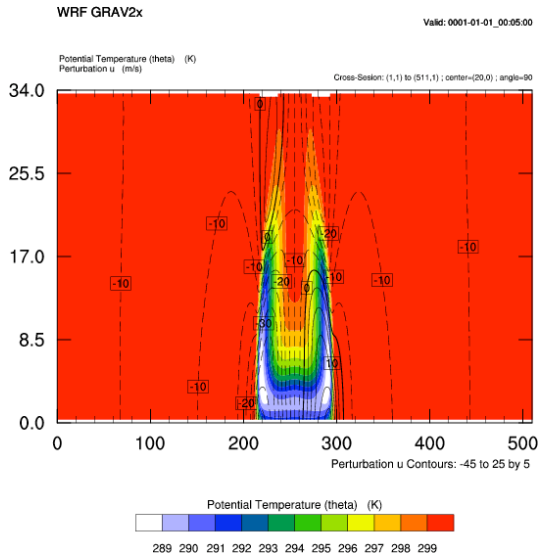
Step-by-step WRF-ARW visualizations

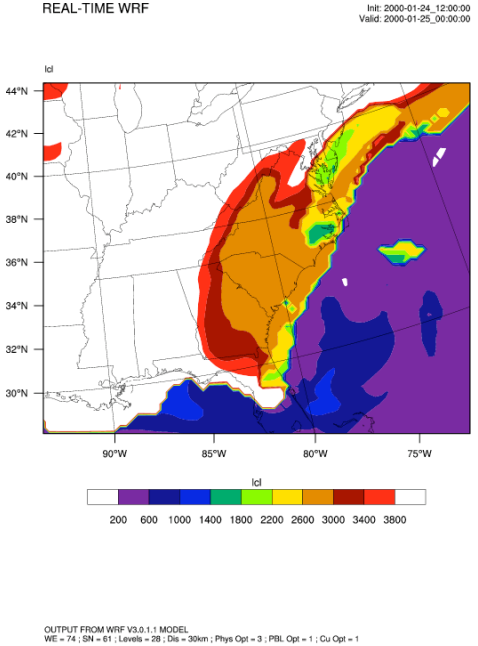












http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/

Step-by-step: filled contours using wrf_xxxx

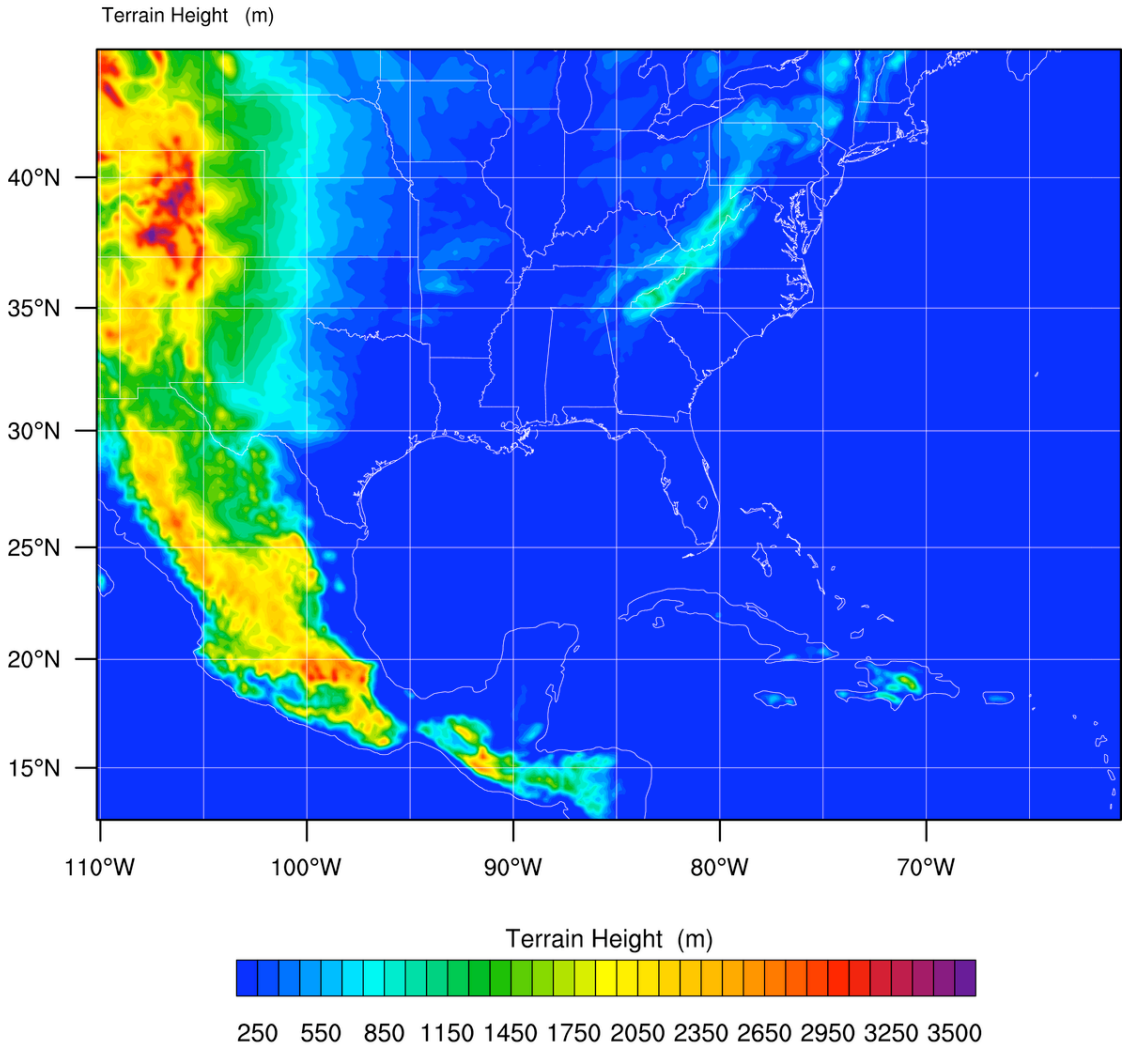
```
; Load the necessary scripts
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

; 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("ps","hgt")  ; "hgt.ps"

; 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

```
. . .
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,"U10",0)

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

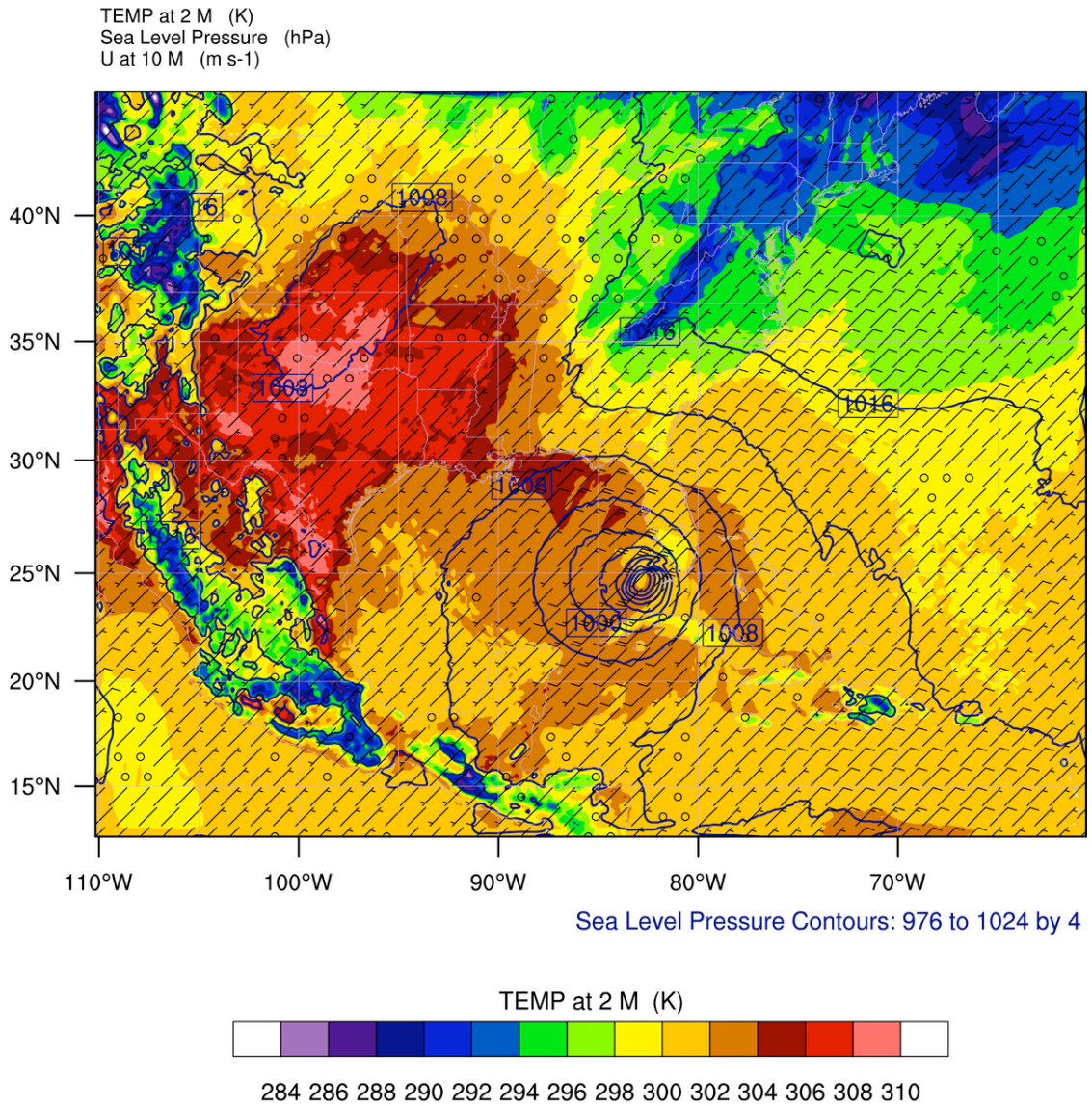
```
; Line contours
os                   = True
os@cnLineColor       = "NavyBlue"
os@cnLineThicknessF  = 2.0
c_slp                = wrf_contour(f,wks,slp,os)
```

```
; 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



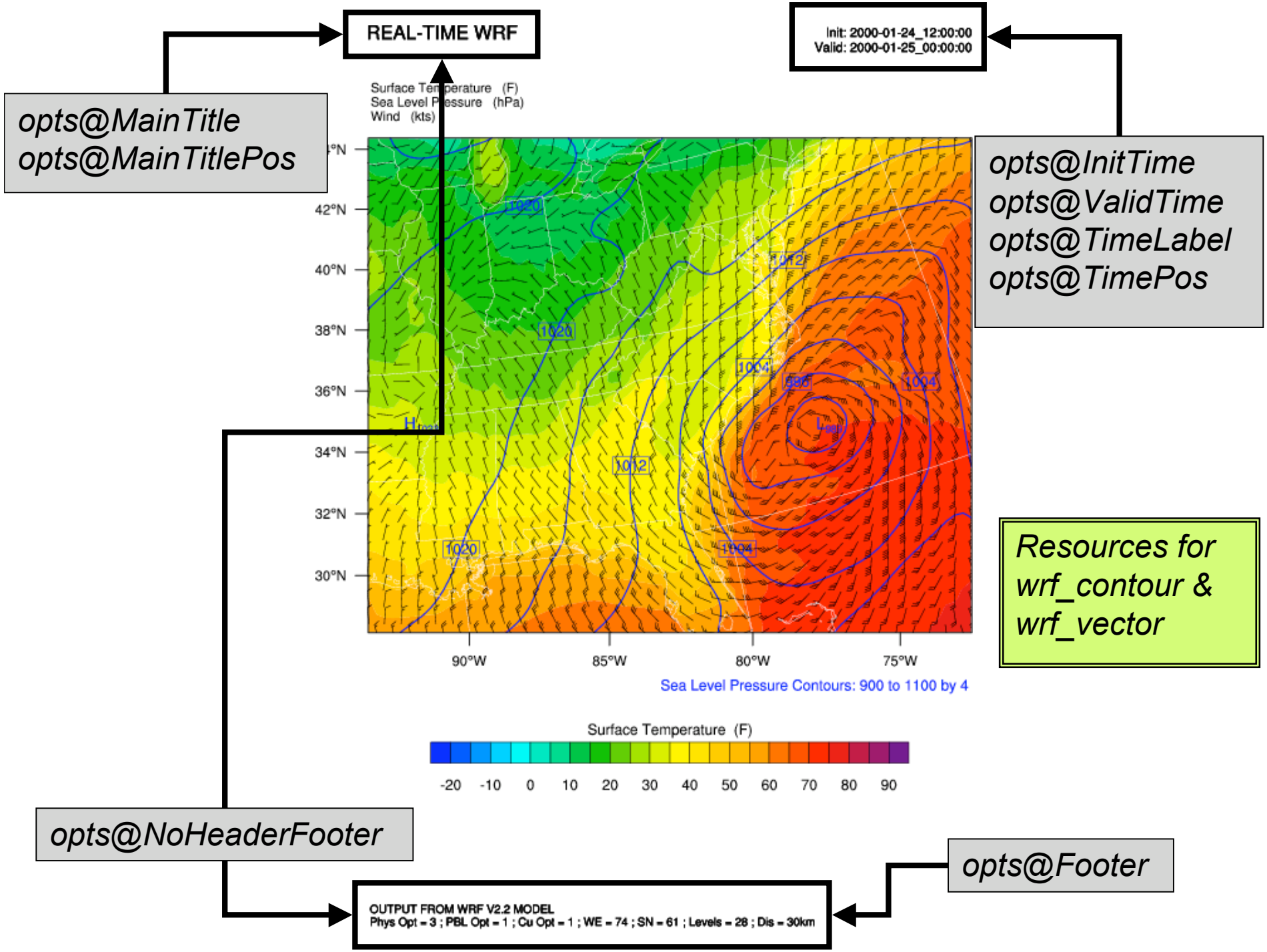
OUTPUT FROM WRF V2.1.2 MODEL
WE = 400 ; SN = 301 ; Levels = 35 ; Dis = 12km ; Phys Opt = 3 ; PBL Opt = 1 ; Cu Opt = 1

wrf_contour/wrf_vector

Create line/shaded/filled contours and vectors

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

- *opts@MainTitle* Main title on the plot
- *opts@MainTitlePos* Main title position
- *opts@NoHeaderFooter* Turn off headers & footers
- *opts@Footer* Add model information as a footer
- *opts@InitTime* Plot initial time on graphic
- *opts@ValidTime* Plot valid time on graphic
- *opts@TimeLabel* Valid time
- *opts@TimePos* Time position
- *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)*

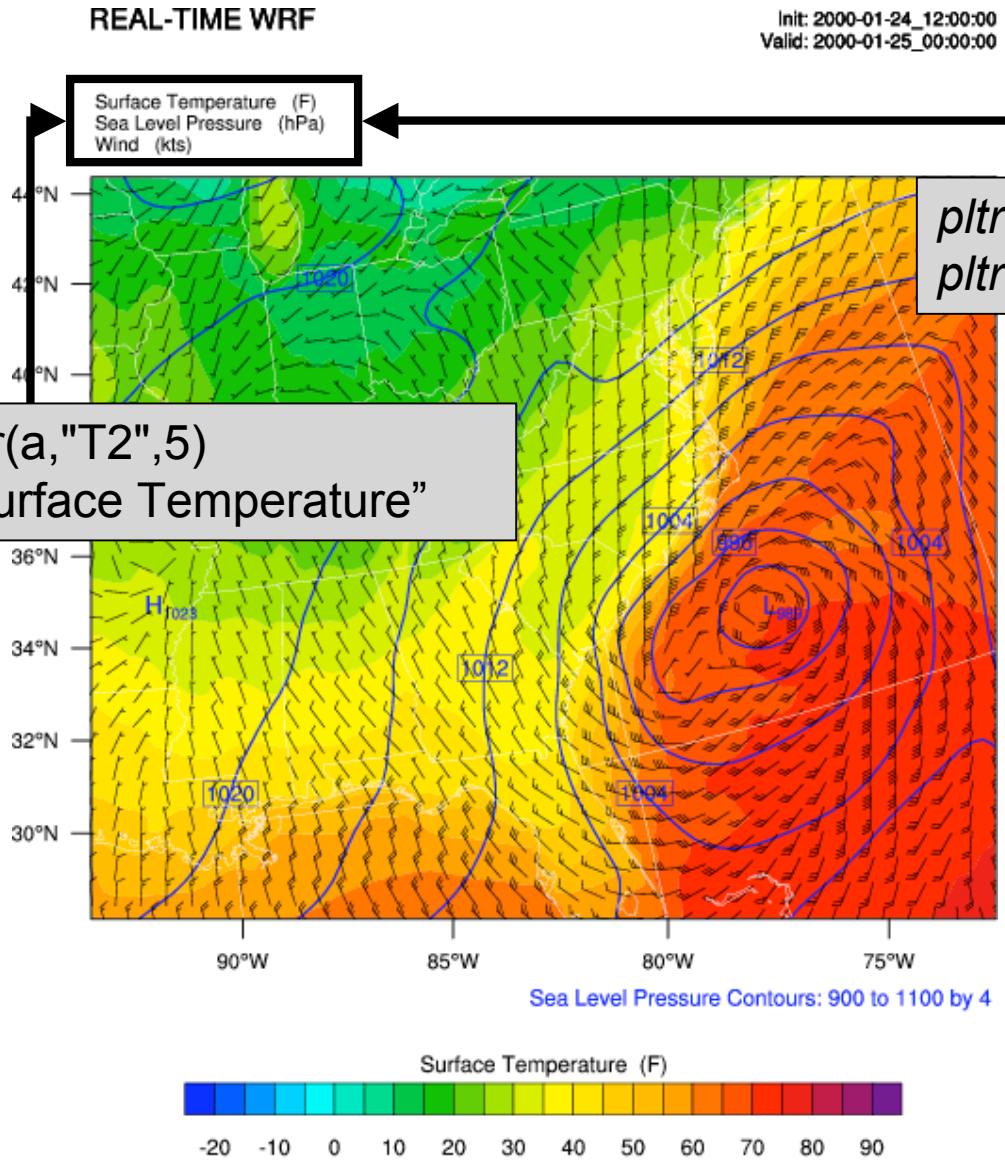

REAL-TIME WRF
Init: 2000-01-24_12:00:00
Valid: 2000-01-25_00:00:00
opts@MainTitle
opts@MainTitlePos
opts@InitTime
opts@ValidTime
opts@TimeLabel
opts@TimePos
Surface Temperature (F)
Sea Level Pressure (hPa)
Wind (kts)
42°N
40°N
38°N
36°N
34°N
32°N
30°N
90°W
85°W
80°W
75°W
1020
1012
1004
996
Sea Level 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)
```

- mpres@mpGeophysicalLineColor; mpres@mpNationalLineColor; mpres@mpUSStateLineColor; mpres@mpGridLineColor; mpres@mpLimbLineColor; mpres@mpPerimLineColor
- 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.
- 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



pltres@NoTitles
pltres@CommonTitle

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

Resources for wrf_overlays

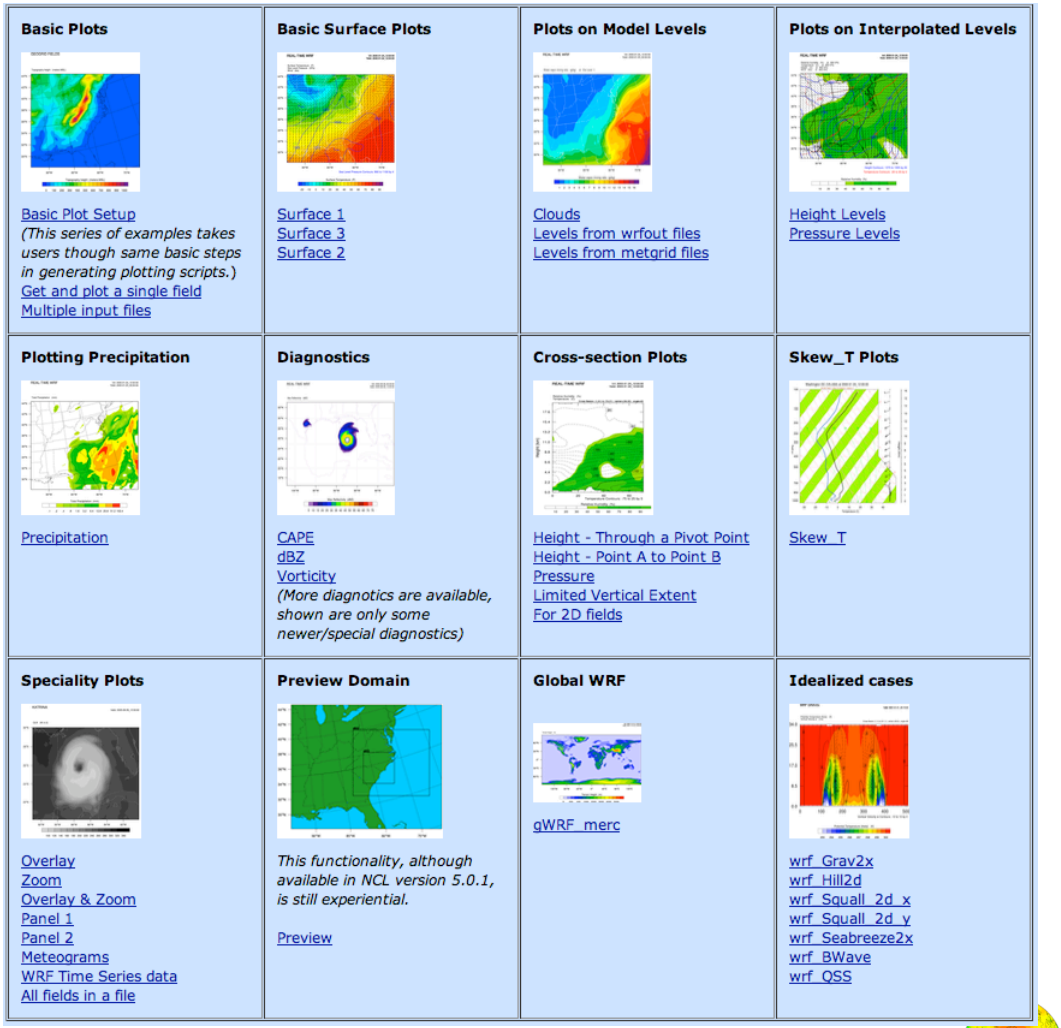
WRF ARW OnLineTutorial

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

Scripts maintained by Cindy Bruyère.

Latest version of WRFUserARW.ncl file usually available here.

Scripts and full-sized images available.



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

More info on plot resources

- The special WRF-NCL graphical functions have special resources they recognize

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

- Most general NCL resources can also be used to tweak plots (some are set internally and can't be changed)

http://www.ncl.ucar.edu/Document/Graphics/Resources/

- Overview
- NCL basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

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!

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
```

- Overview
- NCL basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

Common mistakes or problems

- Forgot .hluresfile
  (colors and fonts will look wrong)

- Call wrf_xxxx functions with the wrong units

- *"cnLineColour" is not a resource in ContourPlot at this time"*
  - Misspelling a resource, "cnLineColour"
  - Using the wrong resource with the wrong plot (i.e. using "vcRefMagnitudeF" in a contour plot).

- Data values in plot look off-scale
  - Maybe "_FillValue" attribute not set or not correct.

Debugging tips

- Start with an existing script, if possible
- 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
- To further examine data, use:
  - print(min(x)) and print(max(x)) ; Minimum/maximum of data
  - print(num(ismissing(x))) ; Count number of msg vals
- For graphics, make sure spelling the resource name correctly
- Group graphical resources alphabetically
- Read errors and warnings carefully

Things to watch for: *memory & efficiency*

- 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
- Reordering the same array multiple times
  - Do once and store to local variable

- Overview
- NCL basics
- File input/output
- Data Analysis
- Visualization
- Calling Fortran code from NCL
- Debugging, common mistakes
- Installation, setup, URLs

Installing NCL and setting up environment

- ESG one-time registration (login/password)
- Download appropriate precompiled binary
- “gunzip” and “tar –xvf” the 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

Customizing your NCL graphics environment
"~/.hluresfile"

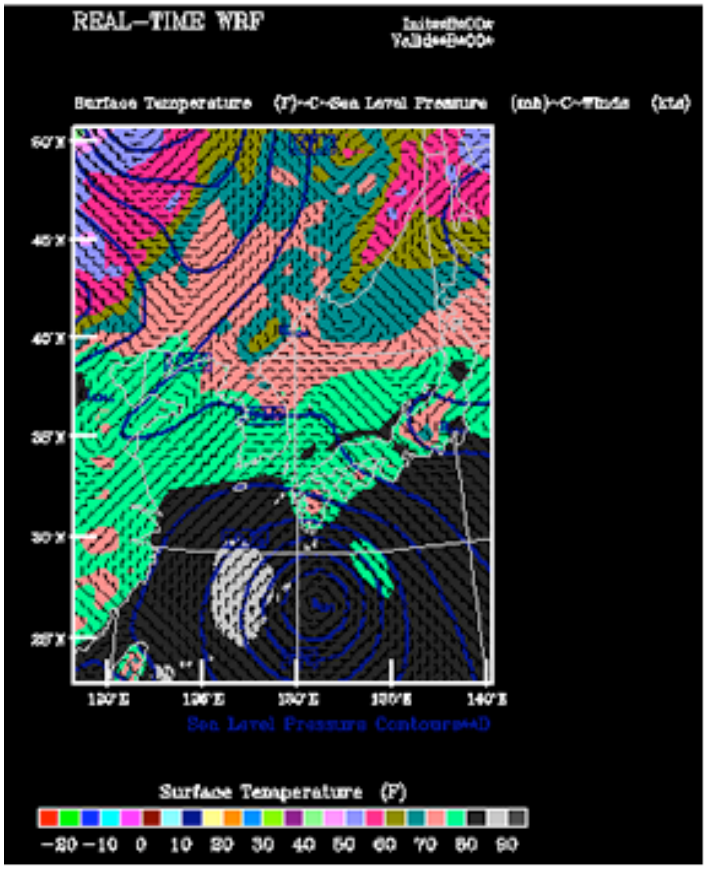
- Download ".hluresfile" file, put in home directory
  - Changes your background, foreground colors to white/black
  - Changes font from **times-roman** to **helvetica**
  - Changes "function code" (default is a colon)
  - ***WRF-NCL users: use to change the default color map***

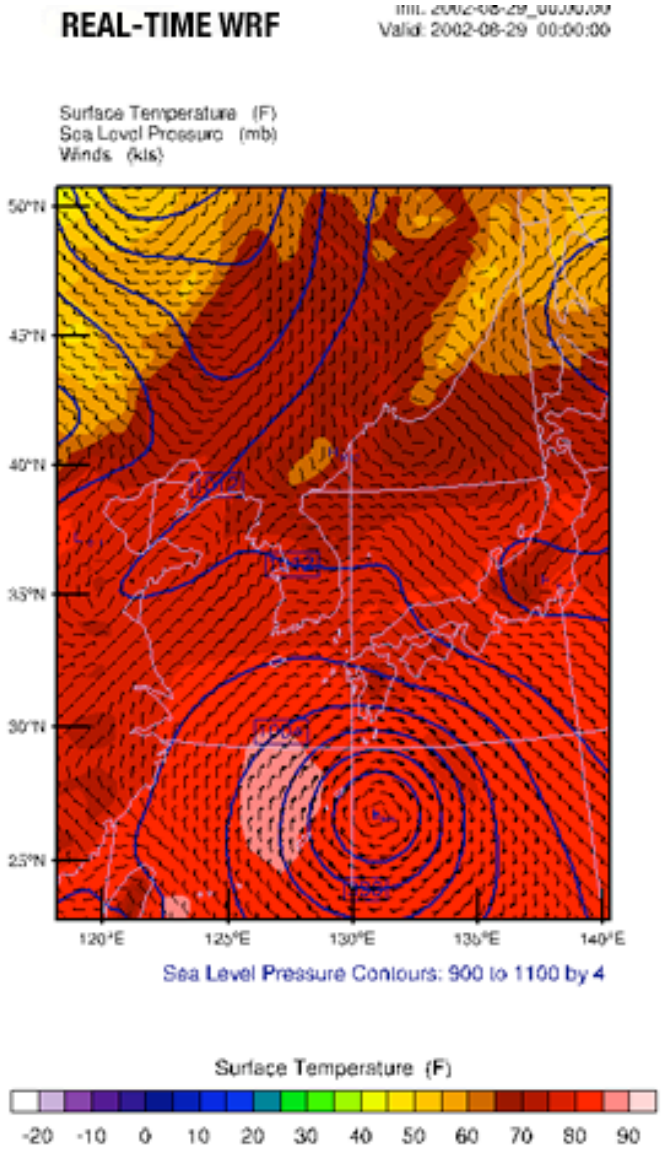
http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml

Sample ".hluresfile"

```
*wkForegroundColor   : black
*wkBackgroundColor   : white
*wkColorMap          : BlAqGrYeOrReVi200
*Font                : helvetica
*TextFuncCode        : ~
*wkWidth             : 900
*wkHeight            : 900
```

With and without a ".hluresfile"





What's new in V5.2.0

- New "PNG" graphical output format
- New variable types: (unsigned int, unsigned long, etc)
- HDF-EOS5 reader
- New functions
- New color tables
- Lots of bug and memory fixes

What's coming in V6.0.0/6.1.0

- Ability to create larger than 2 Gb variables
- Transparency capabilities in graphics
- Parallelism and improvement on slow algorithms
- Not tied to a single color map; no limit on colors!

Useful URLS

- Online WRF-NCL Graphics Tutorial
  http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/
- WRF-NCL functions (built-in and "WRFUserARW.ncl")
  http://www.ncl.ucar.edu/Document/Functions/wrf.shtml
- Graphical resources
  http://www.ncl.ucar.edu/Document/Graphics/Resources/
- Download NCL
  http://www.ncl.ucar.edu/Download/
- Application examples (includes old WRF examples)
  http://www.ncl.ucar.edu/Applications/
- Detailed NCL reference manual
  http://www.ncl.ucar.edu/Document/Manuals/Ref_Manual/
- NCL Workshops
  http://www.ncl.ucar.edu/Training/Workshops/
- NCL email lists to join
  http://www.ncl.ucar.edu/Support/email_lists.shtml

Questions?

wrfhelp@ucar.edu
ncl-talk@ucar.edu