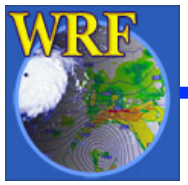
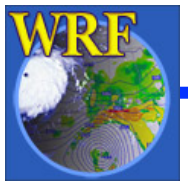
AHW Post Processing

Cindy Bruyère



Processing AHW data

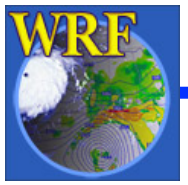
- **ARW output**
  - Which packages can read this data
- **Moving nests**
  - How do we deal with moving nest data
- **Hurricane tracking**
  - Generating and plotting hurricane tracks
- **How to add diagnostics**



Graphical Packages

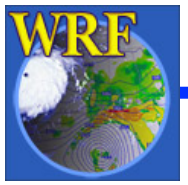
- **NCL** — UG: 9-2
  - Graphical package
- **ARWpost** — UG: 9-28
  - Converter (GrADS & vis5d)
- **RIP4** — UG: 9-19
  - Converter and interface to graphical package NCAR Graphics
- **WPP** — UG: 9-35
  - Converter (GrADS & GEMPAK)
- **VAPOR** — UG: 9-50
  - Converter and graphical package
  - *Support: VAPOR*
- **IDV** — unidata.ucar.edu
  - GRIB (from WPP)
  - GEMPAK (from wrf2gem)
  - vis5d (from ARWpost)
  - CF complaint data (from wrf_to_cf)
  - *Support: unidata*
- **GEMPAK**
  - Data from wrf2gem or WPP
  - *Support: unidata*

MatLab / IDL / R / ferret



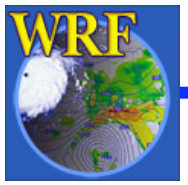
NCL

- NCAR Command Language
- http://www.ncl.ucar.edu
- Read WRF-ARW data directly
- Generate a number of graphical plots
  - Horizontal, cross-section, skewT, meteogram, panel
- Download
  - http://www.ncl.ucar.edu/Download
  - Fill out short registration form *(there is a short waiting period)*
  - Read and agree to OSI-based license
  - Get version 5.2 or later
- NCARG_ROOT environment variable
  - setenv NCARG_ROOT /usr/local/ncl



Generate Plots

- Create a script
  - *wrf_real.ncl*
    (start with a sample script)
- Set NCARG_ROOT environment variable:
  - *setenv NCARG_ROOT /usr/local/ncl*
- Ensure you have an ~/.hluresfile file
- Run NCL script
  - *ncl wrf_real.ncl*



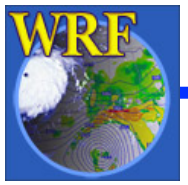
NCL and WRF_NCL

- Combine strength of WRF_NCL specific and NCL general capabilities

```
a = addfile("./wrfout.d01.nc","r")


t2 = a->T2(5,:,:)
t2 = wrf_user_getvar(a,"T2",5)


qv = a->QVAPOR(5,:,:,:)
qv = wrf_user_getvar(a,"QVAPOR",5)


t2 = a->T2
t2 = wrf_user_getvar(a,"T2",-1)
```

```
res@cnLevelSelectionMode = \
     "ManualLevels"
res@cnMinLevelValF = 0.
res@cnMaxLevelValF = 1000.
res@cnLevelSpacingF = 50.
contour=wrf_contour(a,wks,ter,res)


res@ContourParameters = \
     (/0.,1000.,50./)
contour=wrf_contour(a,wks,ter,res)
```

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


mpres@mpGridSpacingF = 45
 plot = wrf_map_overlays \
  (a,wks,(/contour/),pltres,mpres)
```

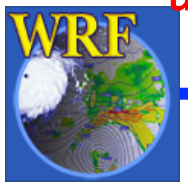


Special WRF Functions

- **wrf_user_getvar**
  Get fields from input file

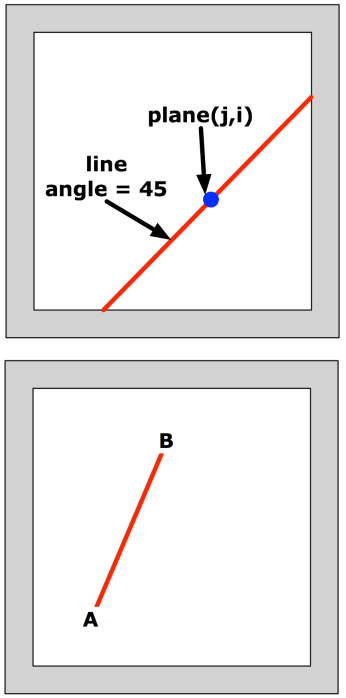
  ter = wrf_user_getvar(a,"HGT",0) {ter=a->HGT(0,:,:)}
  t2 = wrf_user_getvar(a,"T2",-1) {t2=a->T2}
  slp = wrf_user_getvar(a,"slp",1)

  avo/pvo: Absolute/Potential Vorticity,
  cape_2d: 2D mcape/mcin/lcl/lfc,
  cape_3d: 3D cape/cin,
  dbz/mdbz: Reflectivity (3D and max),
  geopt/geopotential: Geopotential,
  p/pres/pressure: Pressure,
  rh/rh2: Relative Humidity (3D and 2m),
  slp: Sea Level Pressure,
  td/td2: Dew Point Temperature (3D and 2m),
  tc/tk: Temperature (C and F), th/theta: Potential Temperature,
  z/height: Height, ua/va/wa: wind on mass points,
  uvmet/uvmet10: wind rotated to earth coordinates (3D and 10m)



Special WRF NCL Functions

- **wrf_user_getvar**
  Get native and diagnostic variables
- **wrf_contour / wrf_vector**
  Create line/shaded & vector plots
- **wrf_map_overlays / wrf_overlays**
  Overlay plots created with wrf_contour and wrf_vector
- **wrf_user_intrp3d / wrf_user_intrp2d**
  Interpolate horizontally to a given pressure/height (3d data only)
  Interpolate vertically along a given line
- **wrf_user_ll_to_ij / wrf_user_ij_to_ll**
  Convert: lat/lon ⟺ ij
- **wrf_user_list_times**
  Get list if times available in input file
- **wrf_user_unstagger**
  Unstaggers an array

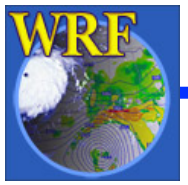


Resources

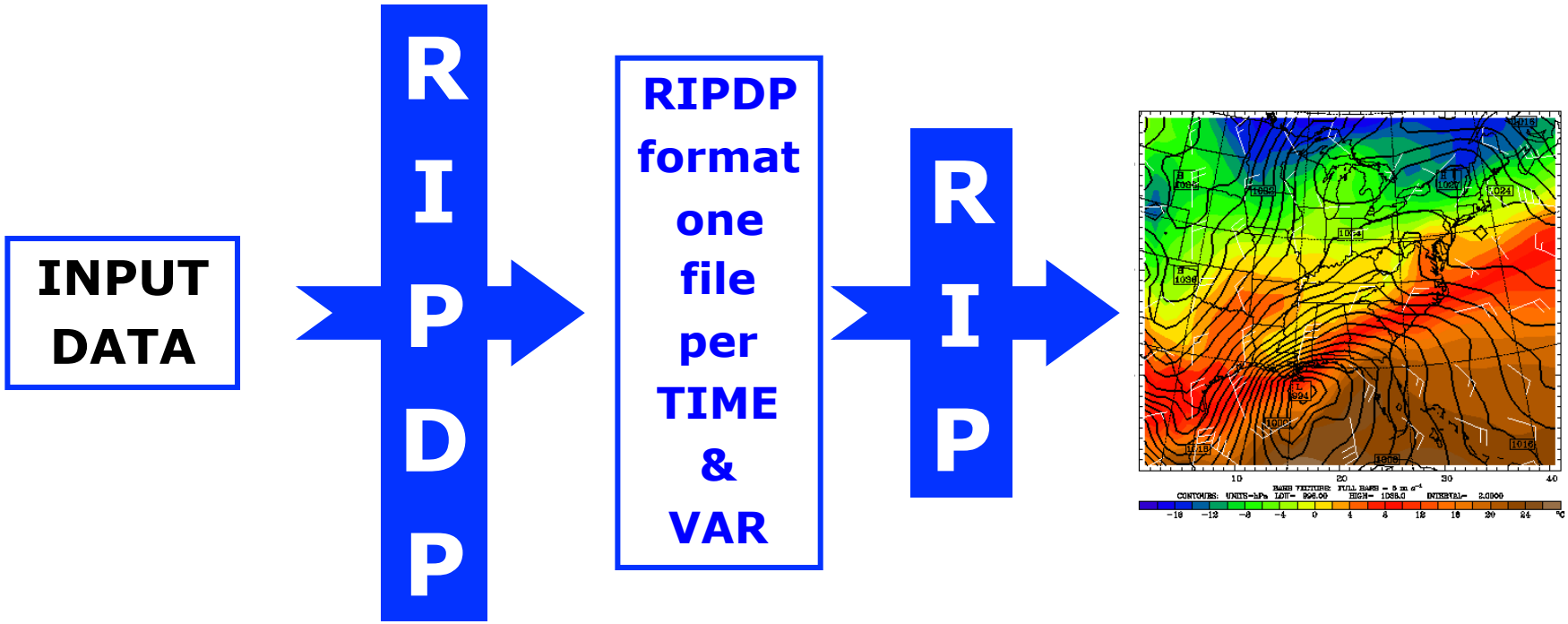
- **The special WRF functions have unique resources:**
  http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_functions.htm

- **All general NCL resources can also be used to control the plot:**
  http://www.ncl.ucar.edu/Document/Graphics/Resources

  - am (annotation manager)
  - app (app)
  - ca (coordinate array)
  - cn (contour)
  - ct (coordinate array table)
  - dc (data comm)
  - err (error)
  - gs (graphic style)
  - gsn (gsn high-level interfaces)
  - lb (label bar)
  - lg (legends)
  - mp (maps)
  - pm (plot manager)
  - pr (primitives)
  - sf (scalar field)
  - st (streamline)
  - tf (transformation)
  - ti (title)
  - tm (tickmark)
  - tr (irregular transformation)
  - tx (text)
  - vc (vectors)
  - vf (vector field)
  - vp (view port)
  - wk (workstation)
  - ws (workspace)
  - xy (xy plots)

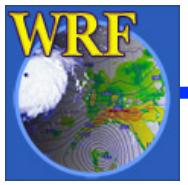


RIP4



General

- Requires NCAR Graphics low-level routines
  - *http://ngwww.ucar.edu*
- NCL Version 5:
  - *http://www.ncl.ucar.edu*
  - Released November 2007
  - Combine NCL and NCAR Graphics
  - Open Source
  - Recommended
- Download Code:
  - *http://www.mmm.ucar.edu/wrf/users/download/get_source.html*
  - *http://www.dtcenter.org/wrf-nmm/users/downloads/index.php*



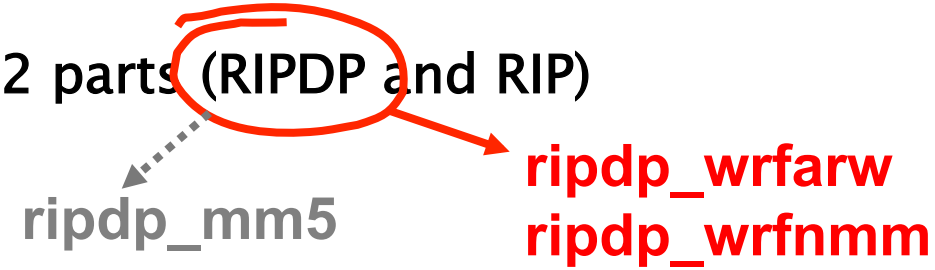
RIP4 on your computer

- **set environment variables**
  setenv RIP_ROOT /usr/$USER/RIP4 *(rip_root)*
  setenv NCARG_ROOT /usr/local/ncarg *(/usr/local/ncl)*

- **Configure**
  ./configure
  (check configure.rip to ensure netCDF paths are correct)

- **Compile**
  ./compile

- **RIP4 has 2 parts (RIPDP and RIP)**



Running ripdp & rip

ripdp_wrf*xxx* [-n namelist-file]
<model_data_name> \ [basic/all]
<input_file(s)>

rip [-f] <model_data_name> rip-execution-name

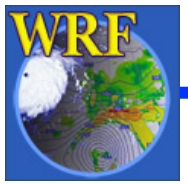
Example:

ripdp_wrfarw RIPDP/arw all wrfout*

rip [-f] RIPDP/arw rip_sample.in

output

[rip_sample.out]
rip_sample.TYPE



rip UIF

```
&userin
    .......
&end
```

Namelist controlling general parameters

```
&trajcalc
    .......
&end
```

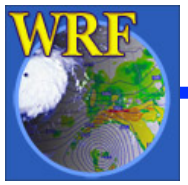
Namelist for trajectory calculations
Only used if itrajcalc=1, in userin namelist

```
=====================================
---------    Plot Specification Table   ----------
=====================================
feld= .........
feld= .........
=====================================
feld= .........
feld= .........
=====================================
```

Frame specification group (FSG)

Plot specification line (PSL)

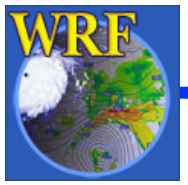
Plot Specification Table (PST)



General

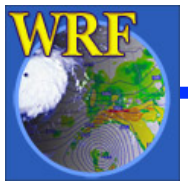
- **Documentation**
  - In program tar file under the Doc/ directory
  - *http://www.mmm.ucar.edu/wrf/users/docs/ripug.htm*
  - *http://www.dtcenter.org/wrf-nmm/users/docs/user_guide/RIP/ripug.htm*

- **OnLine Tutorial:**
  - *http://www.mmm.ucar.edu/wrf/users/graphics/RIP4/RIP4.htm*
  - *http://www.dtcenter.org/wrf-nmm/users/OnLineTutorial/NMM/RIP/index.php*



ARWpost - *converter*

- **Download Code** (http://www.mmm.ucar.edu/wrf/users)
- **OnLine Tutorial**
  http://www.mmm.ucar.edu/wrf/users/graphics/ARWpost/ARWpost.htm
- **For GrADS output**
  - GrADS libraries only needed to display data *(freely available)*
  - http://grads.iges.org/grads/grads.html
- **Version 2 (old – not recommended)**
  - Could produce vis5d output
  - Needed WRFV3 complied

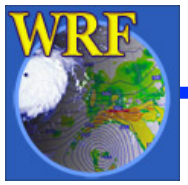


GrADS - .ctl file

```
dset ^test.dat
options byteswapped
undef 1.e37
title  OUTPUT FROM WRF V2.2 MODEL
pdef  259 163 lcc  40.000  -98.000  130.000   82.000
      60.00000  30.00000  -98.00000  22000.000  22000.000
xdef  877 linear -141.49254   0.09909910
ydef  389 linear   18.88639   0.09909910
```

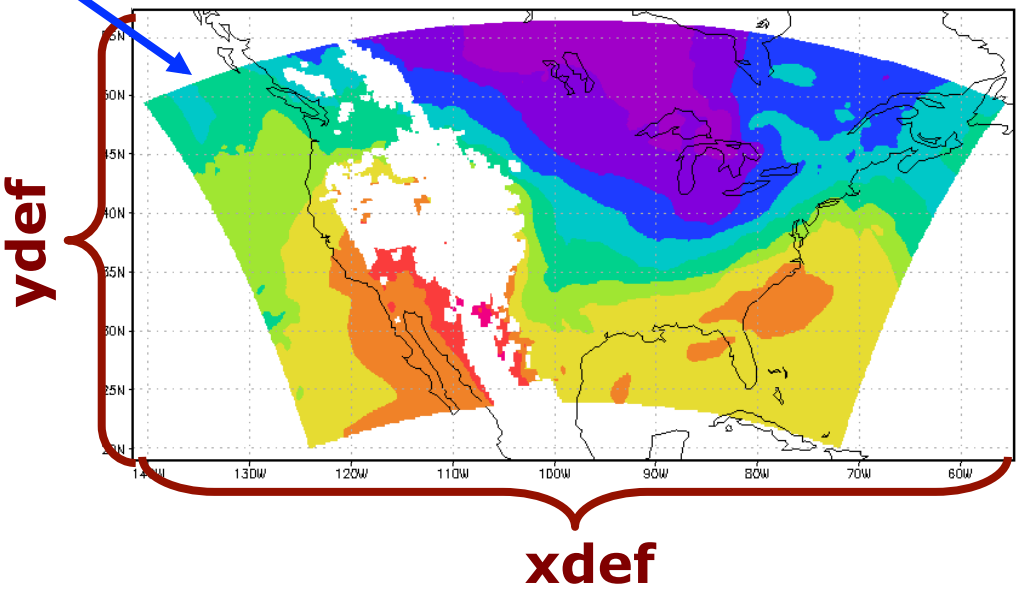
options byteswapped

Needed on some machines - if you get NaNs when you plot, ***remove*** *this line from .ctl file*



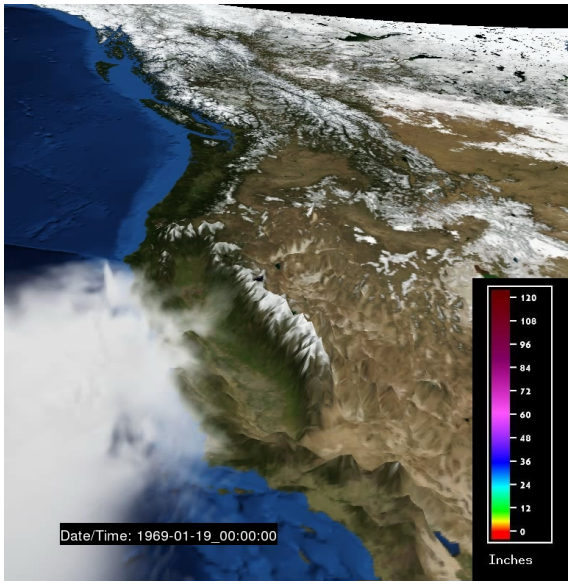
GrADS - .ctl file

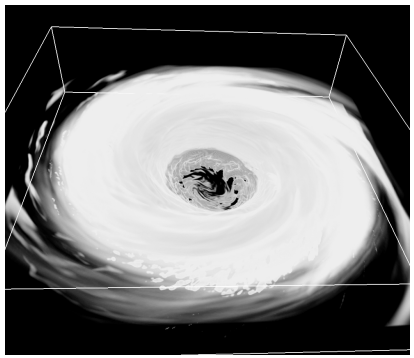
```
dset ^test.dat
options byteswapped
title  OUTPUT FROM WRF V2.2 MODEL
pdef  259 163 lcc  40.000  -98.000  130.000   82.000
60.00000  30.00000  -98.00000  22000.000  22000.000
xdef  877 linear -141.49254   0.09909910
ydef  389 linear   18.88639   0.09909910
```

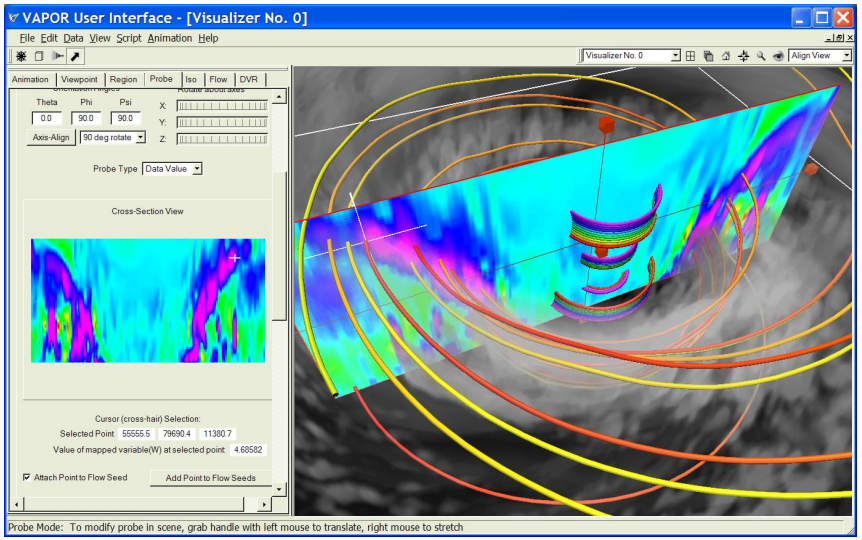


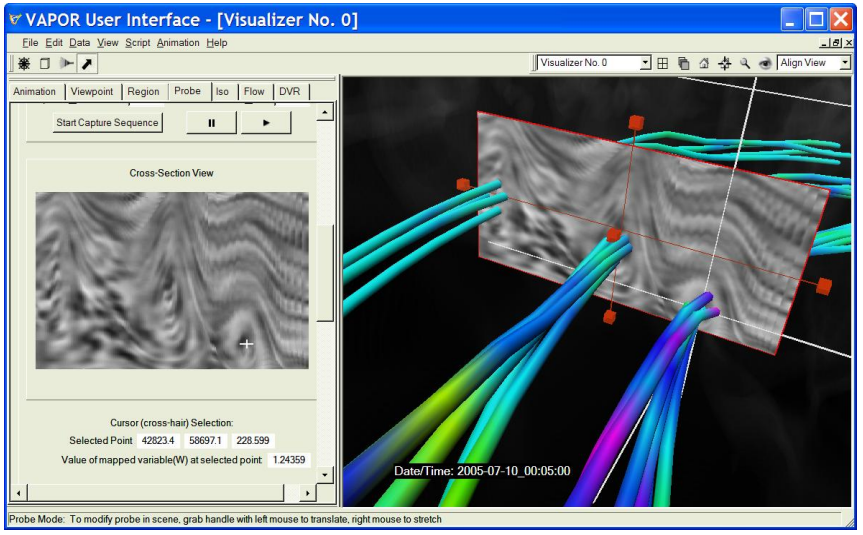
WRF in VAPOR

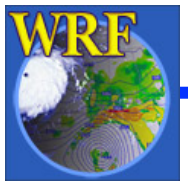
Computational and Information Systems Laboratory
National Center for Atmospheric Research







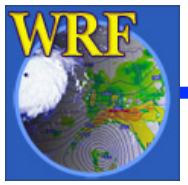




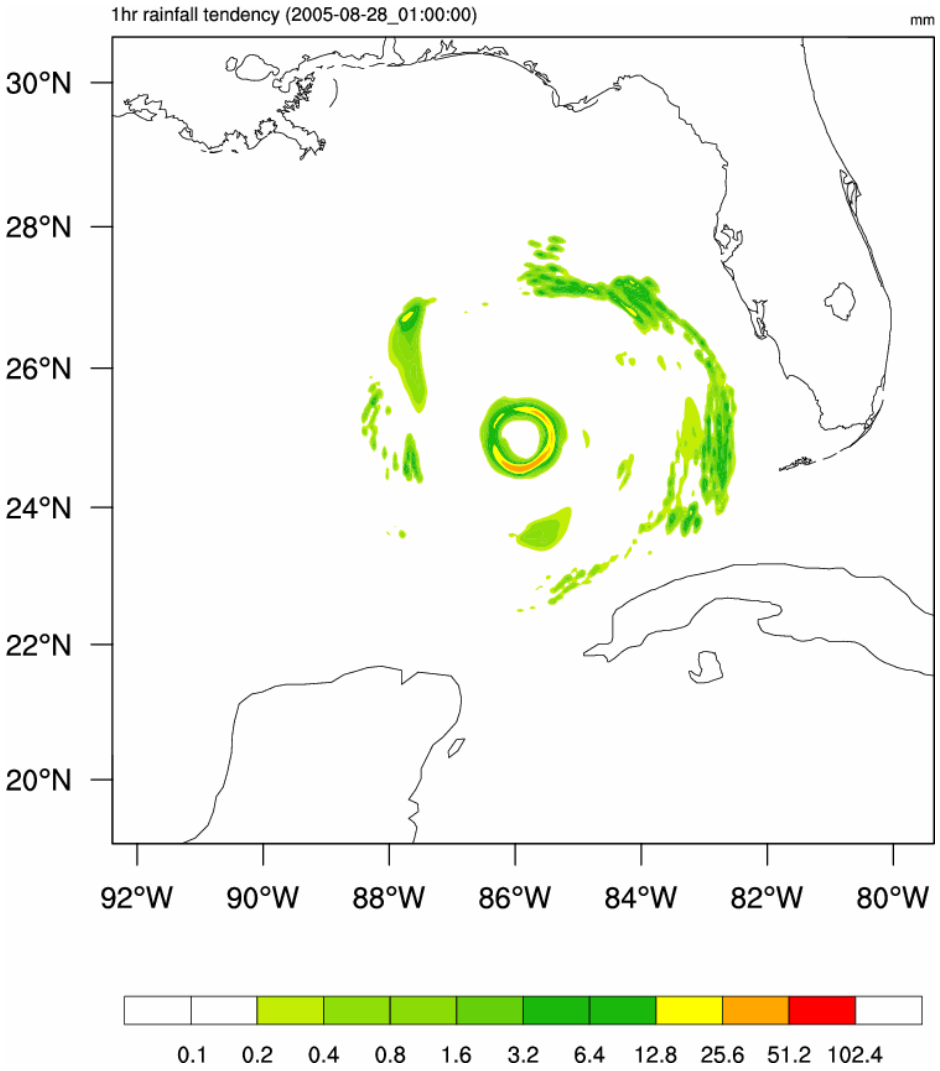
Moving Nest

- **One file per time**
- **Process each time separately**

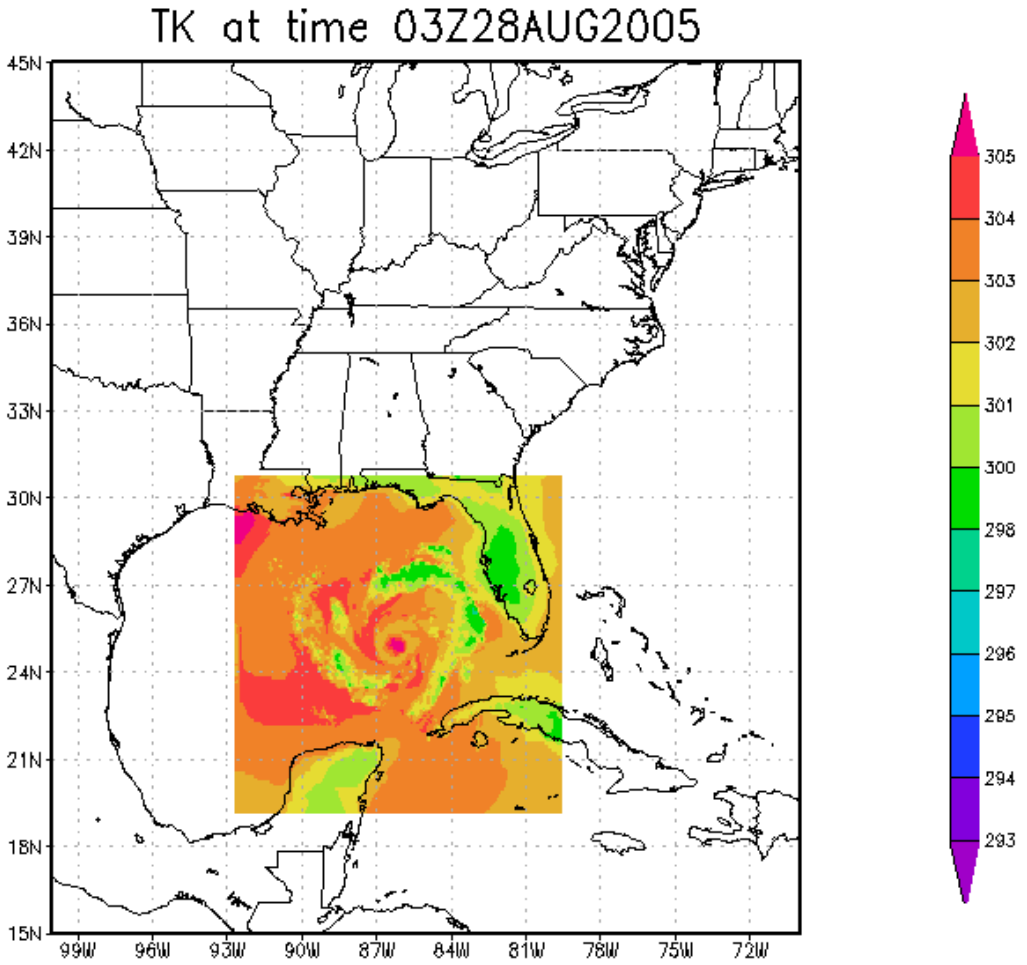
- **Issues**
  - Can plot any variable as per normal, but the background will move
    - *NCL, and to some extend Grads, can use fix backgrounds with moving domains*
  - Rain fields are accumulative, so just plotting the field is not much use
    - *At the moment only NCL can plot the tendencies correctly*



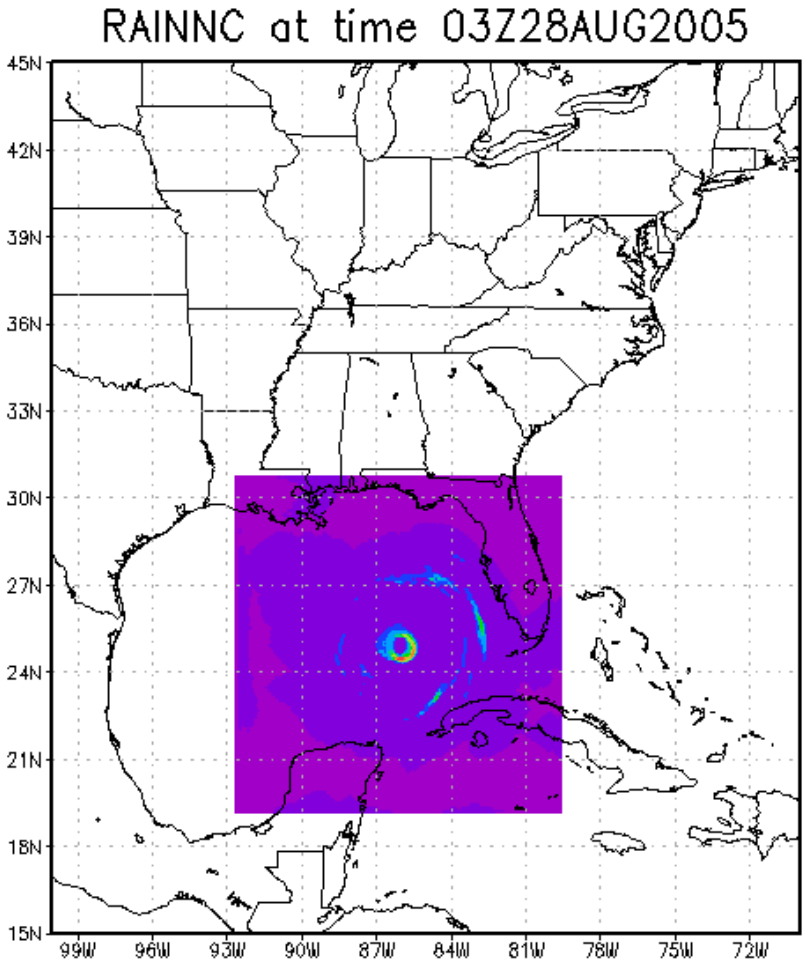
Moving Nests



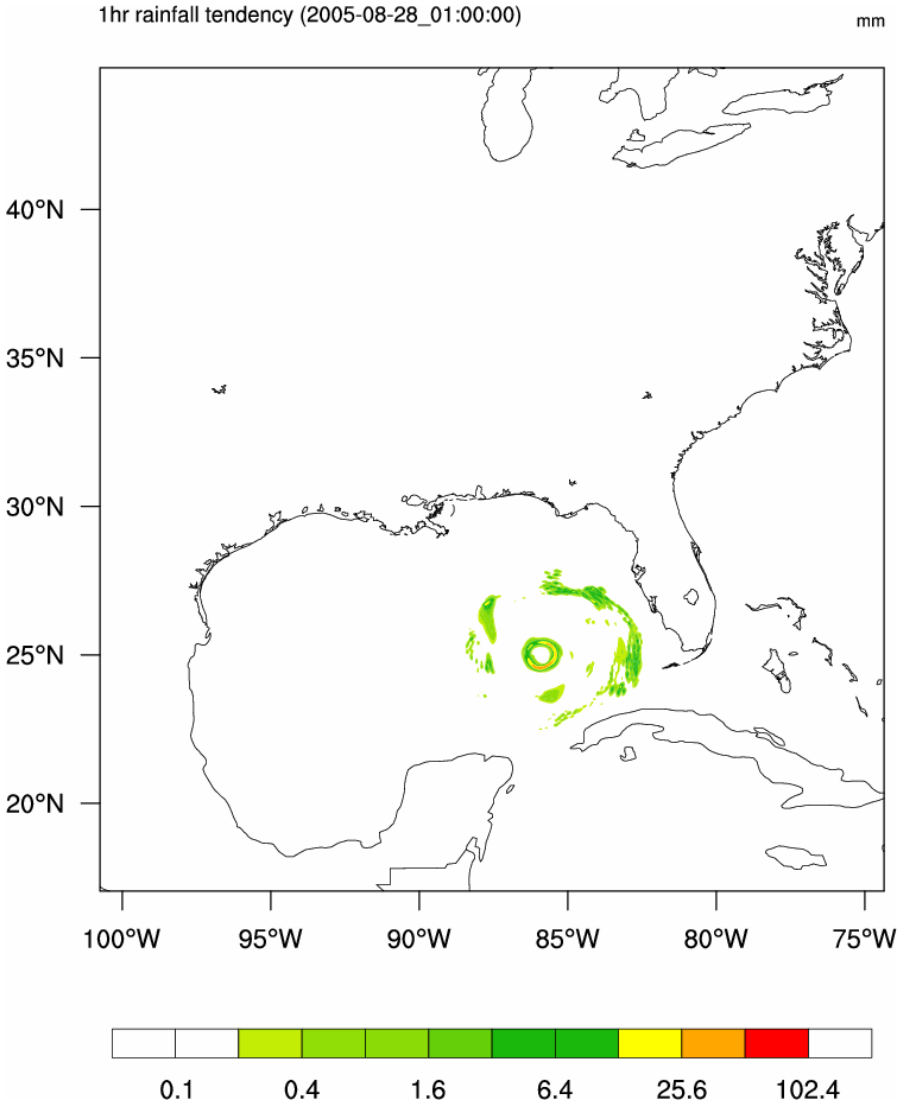
Moving Nests



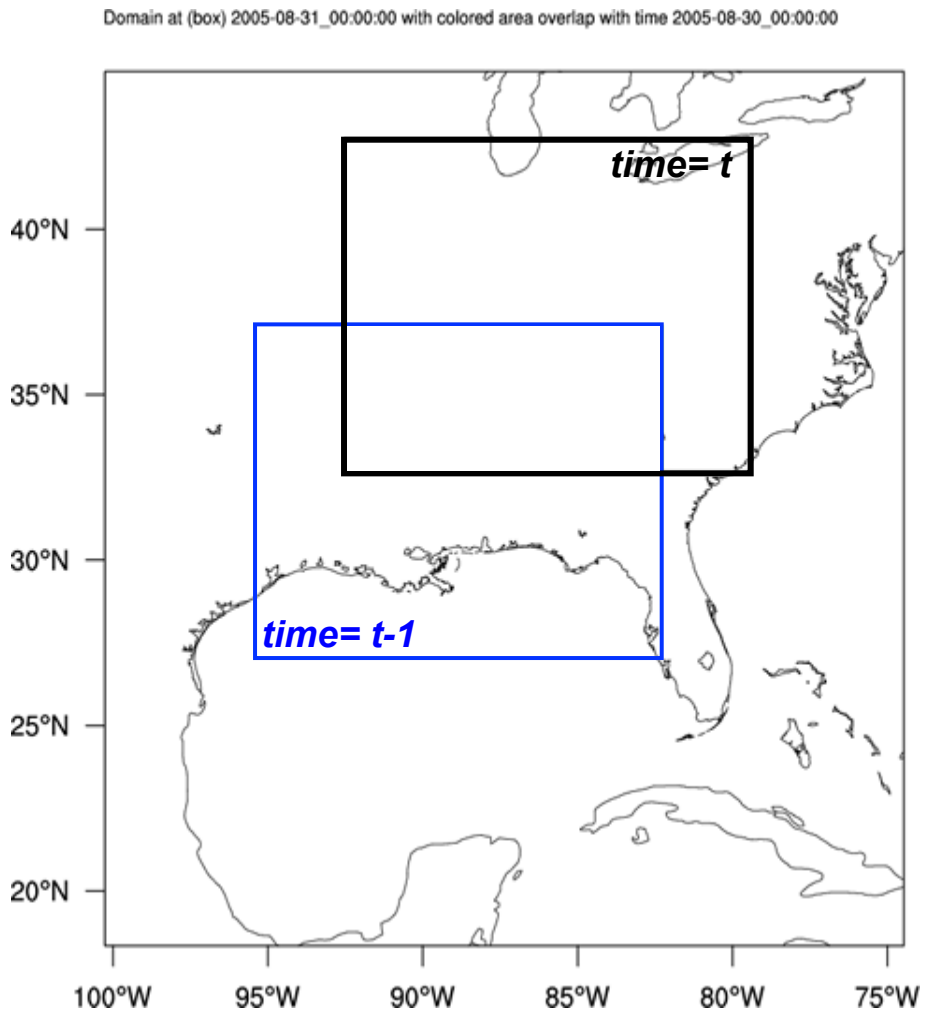
Moving Nests



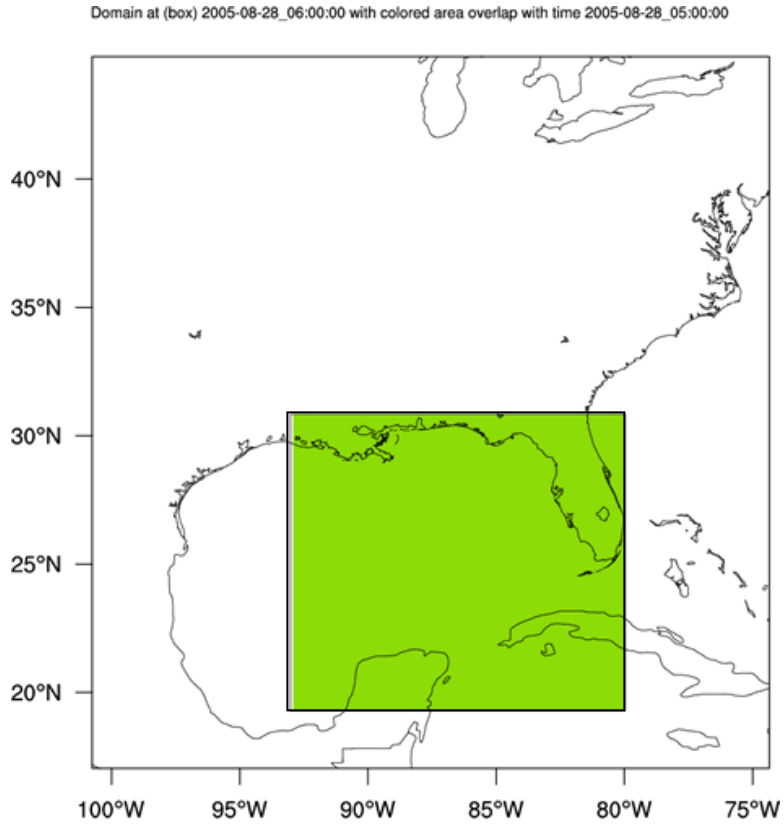
Moving Nests

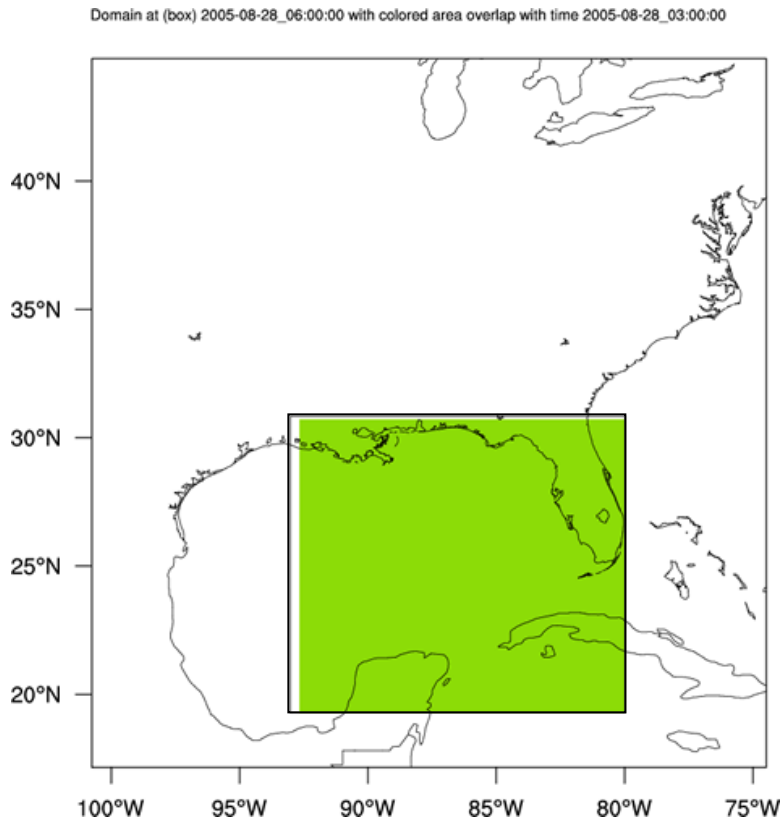


Moving Nest : *Precip Tendencies*



Moving Nest : *Precip Tendencies*

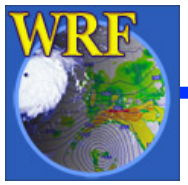




Hurricane Tracking

- **How to track a storm**
  - AHW will output data at run time *(output to the rsl.out.* files)*
  - Text output

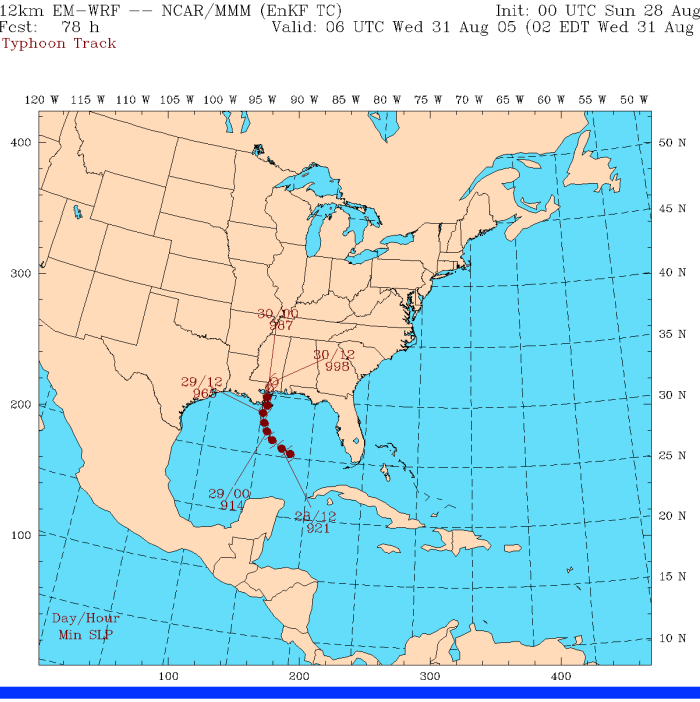
```
ATCF 2005-08-26_00:00:00   25.79   -80.39   990.3   50.4
ATCF 2005-08-26_00:15:00   25.83   -80.41   994.3   52.9
```



Hurricane Tracking

- **How to track a storm**
  - Run RIP4 *(Does NOT work for moving nest)*

```
lat    lon     y       x       t     SLP     wind   xmax    ymax
24.93  85.97  130.12  222.62  0.00  964.08  47.44  223.00  134.00
25.13  86.16  131.87  221.12  1.00  957.02  57.38  222.00  134.00
/
&user_in
ptimes=0,
==========                                                ==========
----------                                                ----------
==========                                                ==========
feld=map; ptyp=hk                                         0.; >
       mllm=land
feld=map; ptyp=hk                                         0.; mllm=land
feld=track; ptyp=                                         ; tynt=6.
feld=tic; ptyp=hk
==========                                                ==========
```

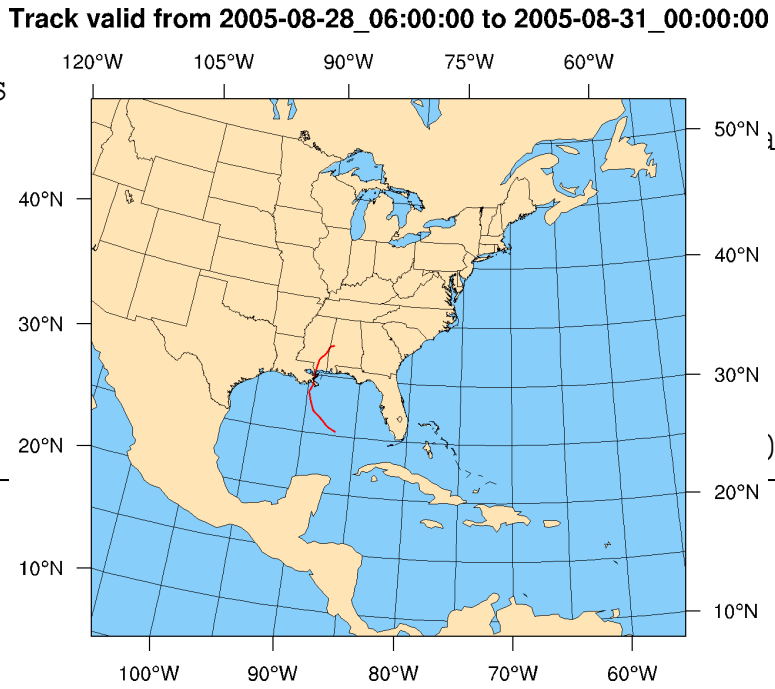


Hurricane Tracking

- **How to track a storm**
  - Run NCL tracking program *(Does NOT work for moving nest)*

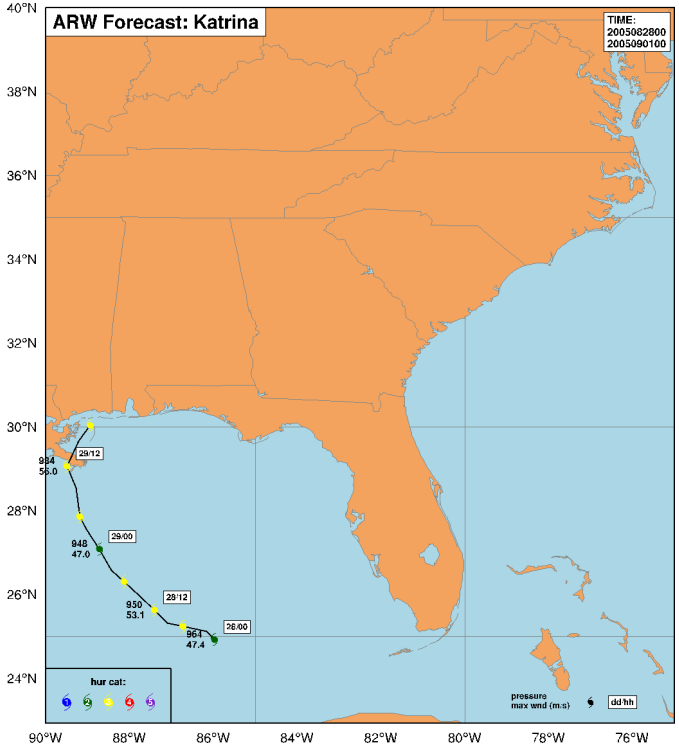
```
lat    lon     t     SLP     wind
24.93 -85.97 0.00 964.08 47.44
25.13 -86.16 1.00 957.02 57.38
```

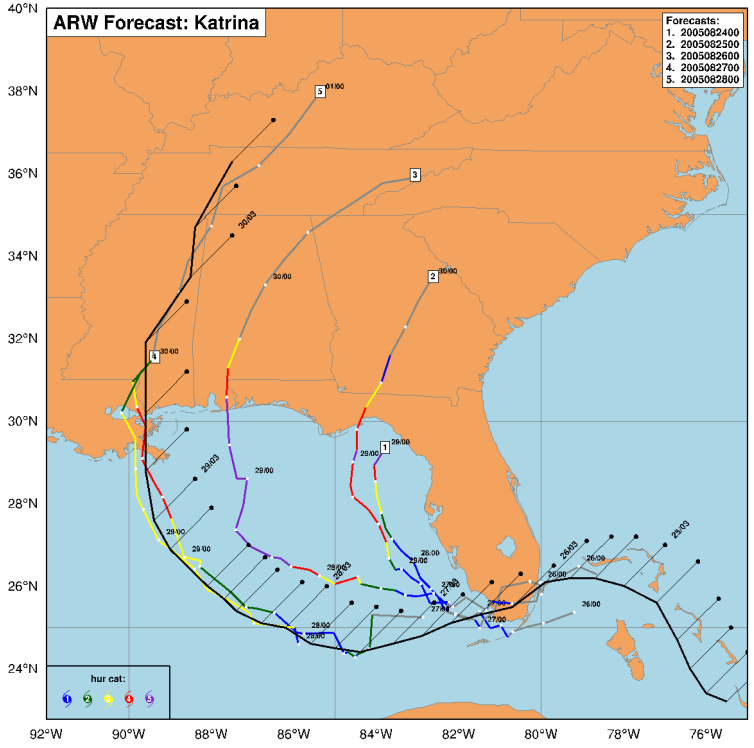
```
Date of storm
Number of storms
name      lat1                                  id      wind    -1


Eg.
05082800
1
KATRINA   24.8N                                  )0KM   51M/S   -1KM
```



Hurricane Tracking

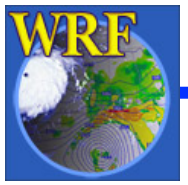
- **How to plot a storm track**
  - RIP4 and NCL your get a plot when track is generated
  - RIP4, NCL and WRF generated tracks can be used in NCL track plotting script *(CreateTracks.ncl)*





How to add diagnostics

- **RIP4**
  - Create a subroutine (note RIP4 expects the code to be in "j/l/-k" orientation)
  - Add links to the RIP4/src/fields.f routine
  - Add new subroutine to RIP4/src/Makefile
- **ARWpost**
  - Create a subroutine
  - Add links to ARWpost/src/module_diagnostics.f90
  - Add new subroutine to ARWpost/src/Makefile
- **NCL**
  - Create an NCL script and load/call from other NCL scripts
  - Link in a FORTRAN / C program into an NCL script

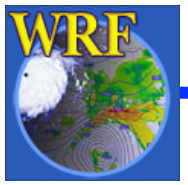


Calling FORTRAN code from NCL

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

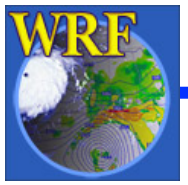
```
C NCLFORTSTART
C NCLEND
```

- 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 pre-allocate arrays!



myTK.f

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

        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)
          enddo
          enddo
        enddo

        end
```

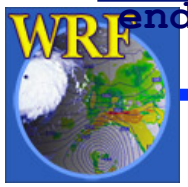


myTK.so - *Create & use in NCL script*

```
% 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
```



FORTRAN 90 code

- Can use simple FORTRAN 90 code
- Your FORTRAN 90 program may not contain any of the following features:
  - pointers or structures as arguments,
  - missing/optional arguments,
  - keyword arguments, or
  - recursive procedure.