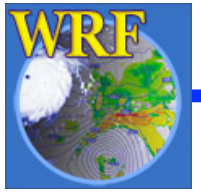
Post-processing Tools

Post-processing Tools

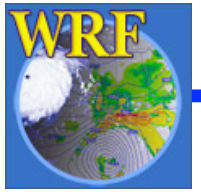
Cindy Bruyère



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

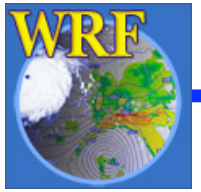


Graphical Packages

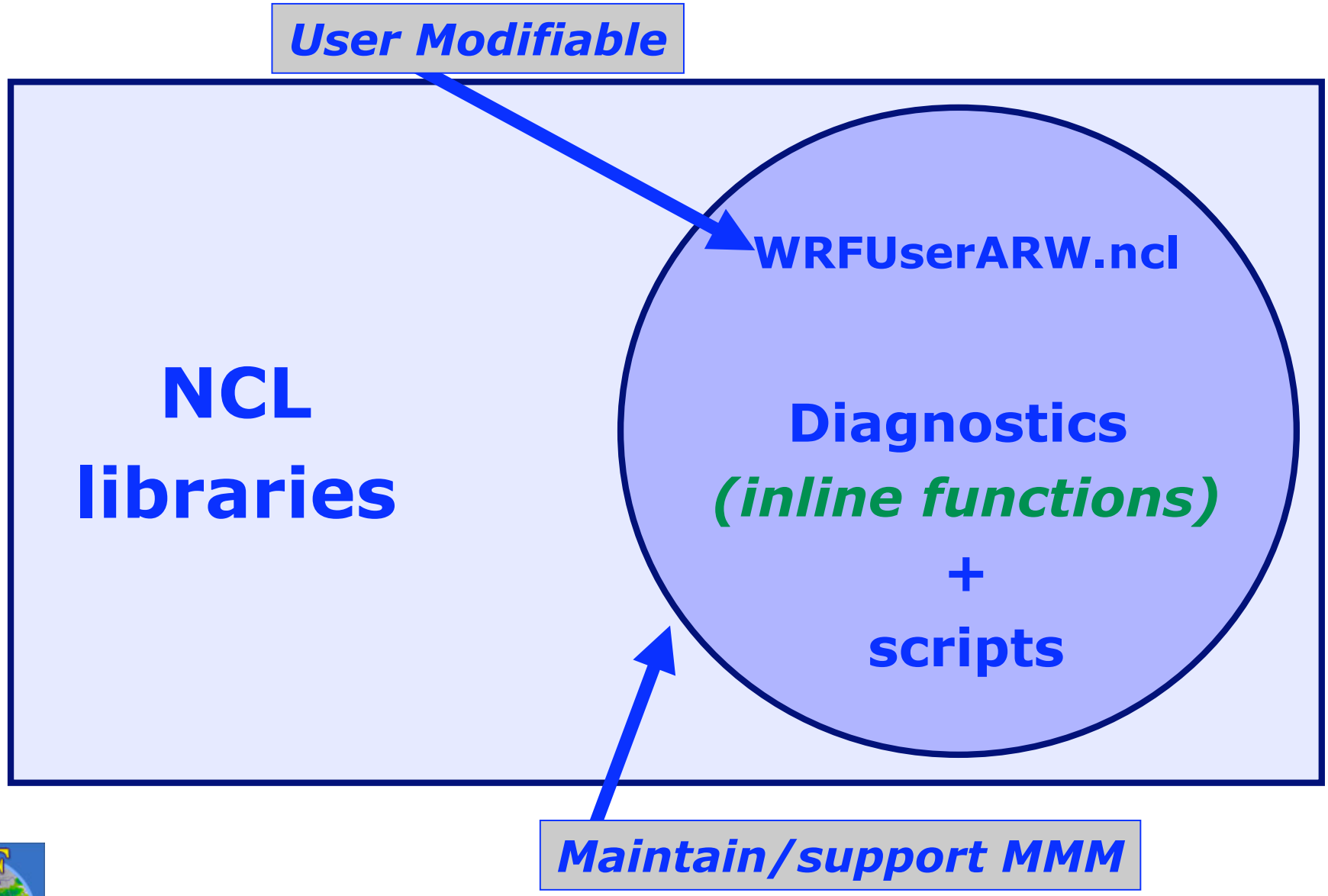
	NCL	RIP4	ARWpost (GrADS / Vis5D)	WPP	VAPOR
Directly ingest WRF data	***Y***	**N** *converter*	**N /** ***(Y)*** *converter*	**N** *converter*	**N** *converter*
Intermediate files	***N***	lots	large file	**Y**	large file
WPS DATA	***Y***	***Y***	***Y***	**N**	**N**
wrfinput data	***Y***	***Y***	***Y***	**N**	**N**
Idealized data files	***Y***	***Y***	***Y***	**N**	**N**
Input format	*netCDF*	*netCDF*	*netCDF / GRIB1*	*netCDF / binary*	*netCDF*
Vertical Output Coordinate	eta pressure height	eta pressure height	eta pressure height	pressure	eta
Software required ***(All binaries are free)***	*NCL*	*NCARG*	*GrADS/vis5d*	*GrADS / GEMPAK*	*VAPOR*
Diagnostics	*some*	*> 100*	*some*	*some*	*limited*

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 5.1.0 or later
- **NCARG_ROOT environment variable**
  - setenv NCARG_ROOT /usr/local/ncl



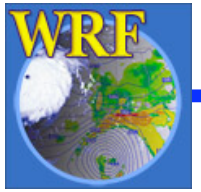
NCL & WRF


User Modifiable
WRFUserARW.ncl
NCL
libraries
Diagnostics
(inline functions)
+
scripts
Maintain/support MMM

Home Directory → ~/.hluresfile ← Very Important

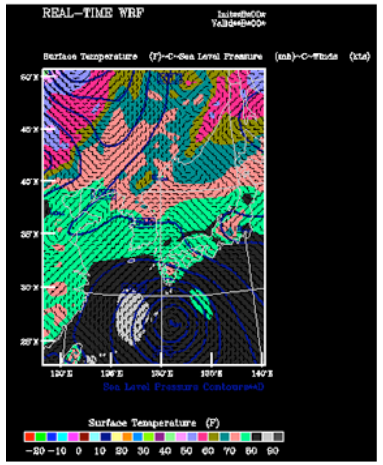
- **Required by NCL libraries**
- **Must be in your "~/" directory *(home directory)***
- **Control**
  - color table ; font
  - white/black background
  - size of plot
  - control characters

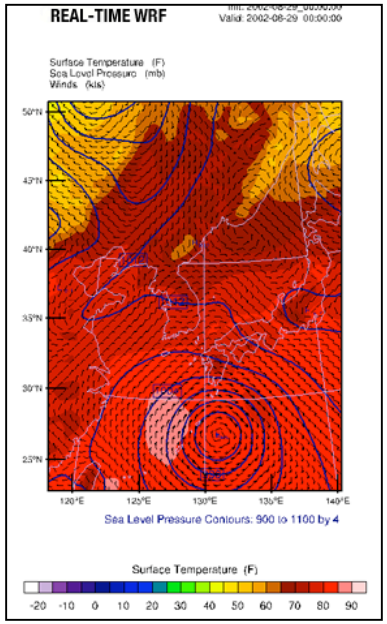
- **http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml**



~/.hluresfile

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

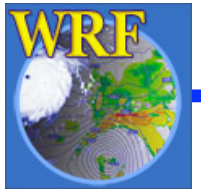




http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/.hluresfile

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*

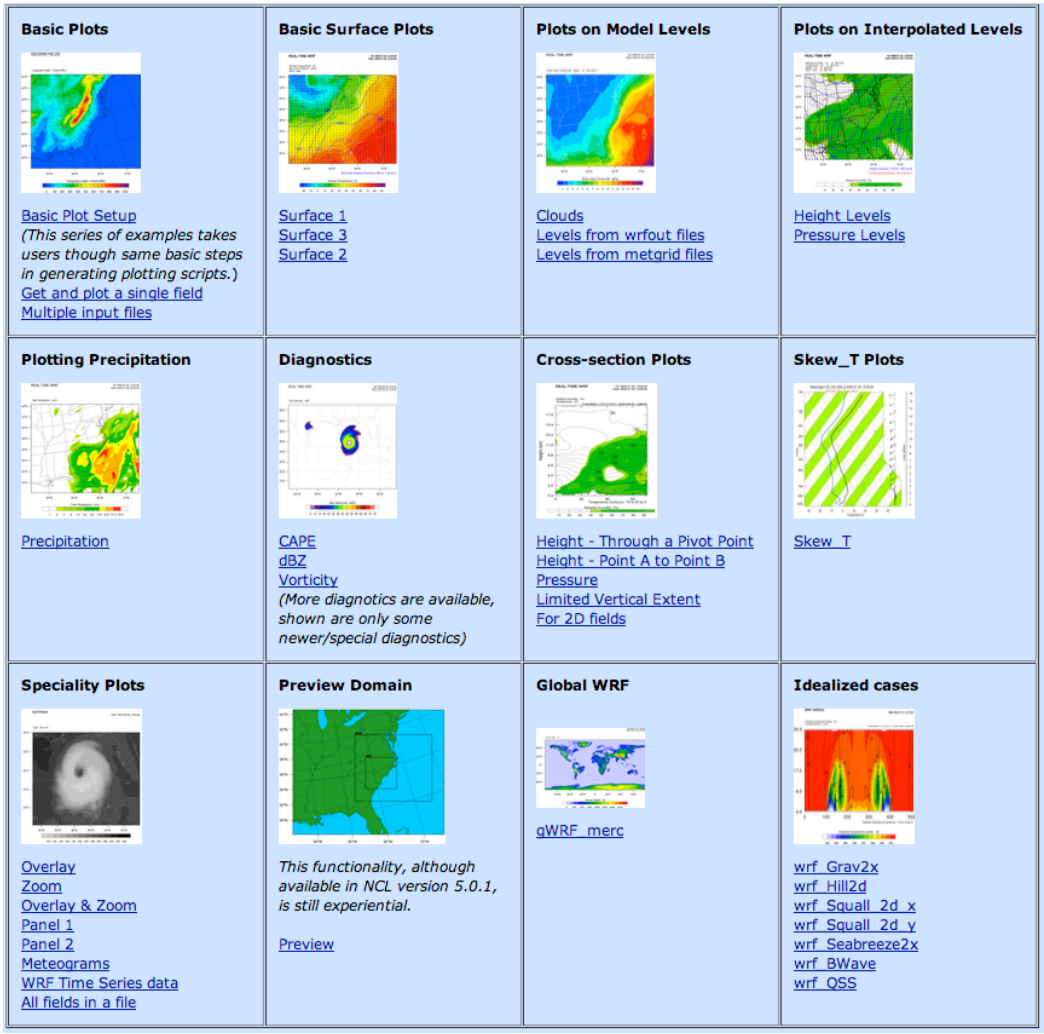


Generate Plots: *A good start - OnLine Tutorial*

http://www.mmm.ucar.edu/

wrf/OnLineTutorial/Graphics/

NCL/index.html



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

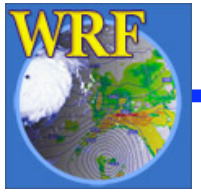
gWRF_merc

Idealized cases

wrf_Grav2x
wrf_Hill2d
wrf_Squall_2d_x
wrf_Squall_2d_y
wrf_Seabreeze2x
wrf_BWave
wrf_QSS

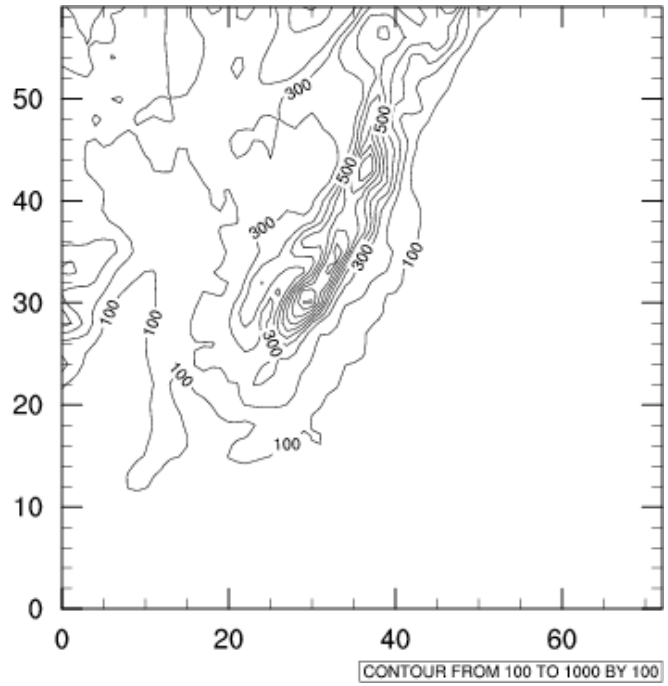
Creating a Plot : NCL script

```
load ncl library scripts

begin

  ; Open graphical output
  ; Open input file(s)

  ; Read variables

  ; Set up plot resources & Create plots
  ; Output graphics

end
```



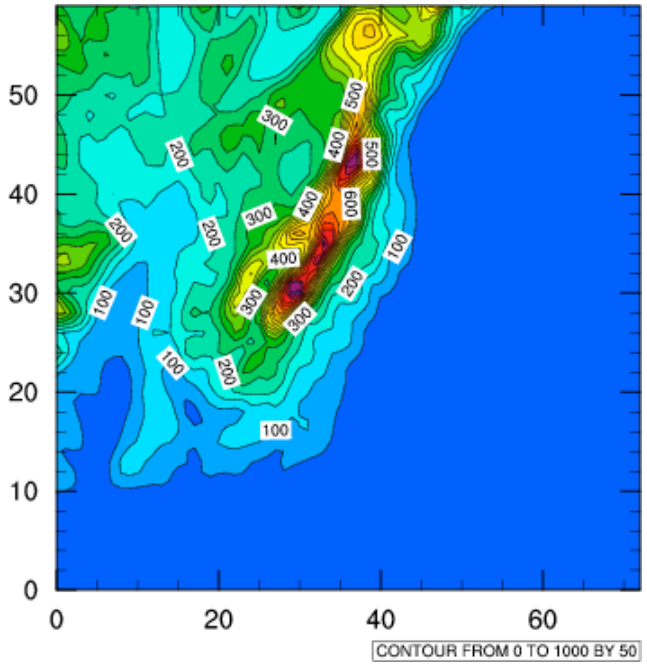
Generate Plots

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

begin
  a = addfile("./geo_em.d01.nc","r")
  wks = gsn_open_wks("pdf","plt_ter1")
  ter = a->HGT_M(0,:,:)
  plot = gsn_contour(wks,ter,True)
end
```



Generate Plots

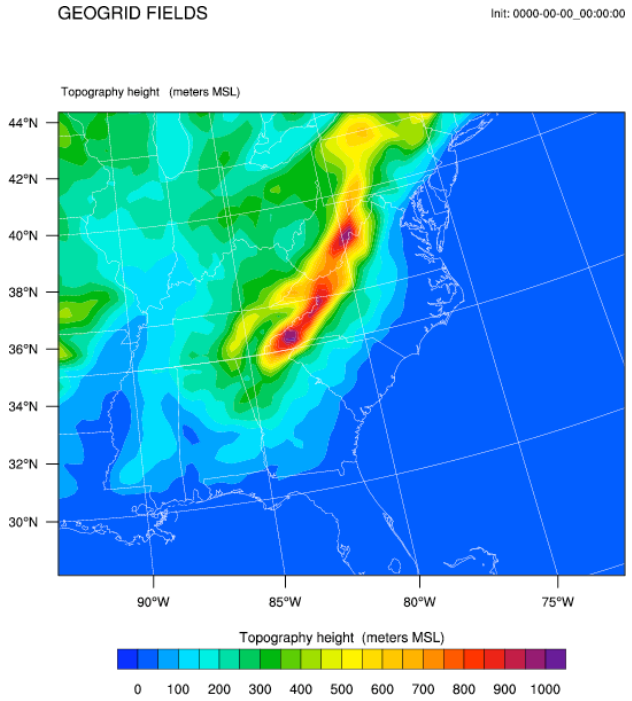
```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"

begin
  a = addfile("./geo_em.d01.nc","r")
  wks = gsn_open_wks("pdf","plt_ter1")
  ter = a->HGT_M(0,:,:)
  res = True
  res@cnFillOn = True
  res@gsnSpreadColors = True
  res@cnLevelSelectionMode = \
     "ManualLevels"
  res@cnMinLevelValF = 0.
  res@cnMaxLevelValF = 1000.
  res@cnLevelSpacingF = 50.
  plot = gsn_contour(wks,ter,res)
end
```



Basic NCL resources - *there are over 1400 controlling contours, labelbars, legends, maps, etc.*

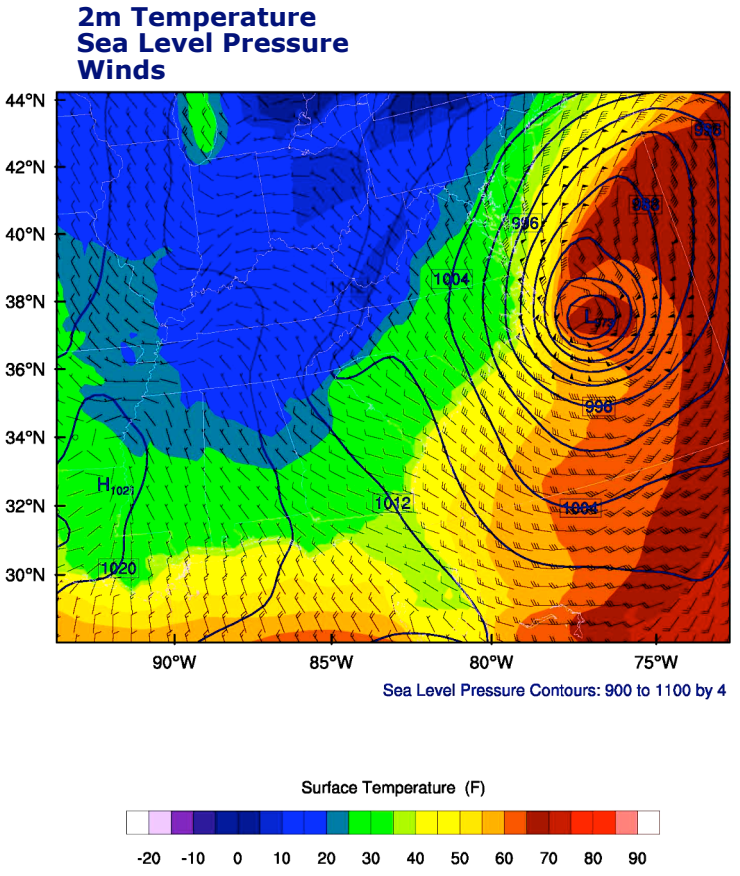
Generate Plots

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

begin
  a = addfile("./geo_em.d01.nc","r")
  wks = gsn_open_wks("pdf","plt_ter5")
  res = True
  res@MainTitle = "GEOGRID FIELDS"
  pltres = True
  mpres = True

  ter = wrf_user_getvar(a,"HGT_M",0)
  res@cnFillOn = True
  res@ContourParameters = (/0.,1000.,50./)
  contour = wrf_contour(a,wks,ter,res)
  plot = wrf_map_overlays(a,wks,(/contour/),\
    pltres,mpres)
end
```



Generate Plots

```
slp = wrf_user_getvar(a,"slp",5)
t2  = wrf_user_getvar(a,"T2",5)
u10 = wrf_user_getvar(a,"U10",5)
v10 = wrf_user_getvar(a,"V10",5)

os@cnLineColor = "NavyBlue"
c_slp = wrf_contour(a,wks,slp,os)
ot@cnFillOn = True
c_tc = wrf_contour(a,wks,t2,ot)
ov@NumVectors = 47
vec = wrf_vector(a,wks,u10,v10,ov)

plot = wrf_map_overlays(a, wks, \
       (/c_tc,c_slp,vec/), pltres, mpres)
```

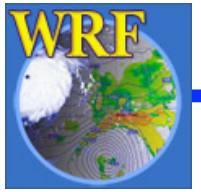


Special WRF Functions

- **wrf_user_getvar**
  Get fields from input file

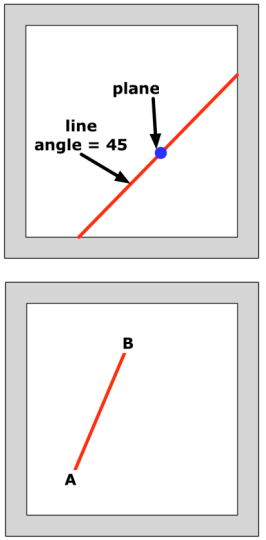
  ter = wrf_user_getvar(a,"**HGT**",0)
  t2 = wrf_user_getvar(a,"**T2**",-1)
  slp = wrf_user_getvar(a,"**slp**",1)

  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,
  z/height: Height, **ua/va/wa**: wind on mass points,
  uvmet/uvmet10: wind rotated to earth coordinates



Special WRF Functions

- **wrf_contour / wrf_vector**
  Create line/shaded & vector plots
- **wrf_map_overlays / wrf_overlays**
  Overlay plots created with wrf_contour and wrf_vector
- **wrf_map**
  Create a map background - not used often
- **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



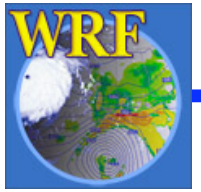
Functions

- **Special WRF NCL Built-in Functions**
  Mainly functions to calculate diagnostics
  Seldom need to use these directly

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

- **Special WRF functions**
  Developed to make it easier to generate plots
  $NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl

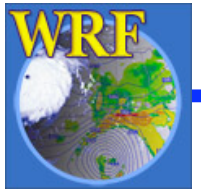
```
slp = wrf_user_getvar(nc_file,"slp",time)
```



WRF / Built-in Functions

Use NCL inline functions, in place of wrf_user_getvar

```
T       = nc_file->T(time,:,:,:)
P       = nc_file->P(time,:,:,:)
PB      = nc_file->PB(time,:,:,:)
QVAPOR = nc_file->QVAPOR(time,:,:,:)
PH      = nc_file->PH(time,:,:,:)
PHB     = nc_file->PHB(time,:,:,:)
T = T + 300.
P = P + PB
QVAPOR = QVAPOR > 0.000
PH    = ( PH + PHB ) / 9.81
z = wrf_user_unstagger(PH,PH@stagger)
tk = wrf_tk( P , T )
slp = wrf_slp( z, tk, P, QVAPOR )
```

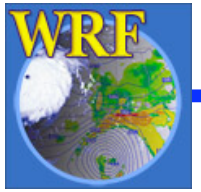


WRF / Built-in Functions

Use NCL inline functions, in place of wrf_user_getvar

```
T      = nc_file->T(time,:,:,:)
P      = nc_file->P(time,:,:,:)
PB     = nc_file->PB(time,:,:,:)
QVAPOR = nc_file->QVAPOR(time,:,:,:)
PH     = nc_file->PH(time,:,:,:)
PHB    = nc_file->PHB(time,:,:,:)
T = T + 300.
P = P + PB
QVAPOR = QVAPOR > 0.000
PH    = ( PH + PHB ) / 9.81
z = wrf_user_unstagger(PH,PH@stagger)
tk = wrf_tk( P , T )
slp = wrf_slp( z, tk, P, QVAPOR )
```

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

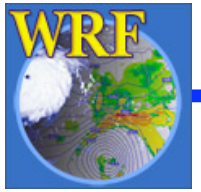


Resources

- **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)



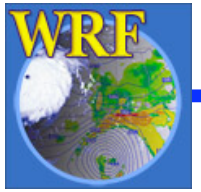
Calling FORTRAN code from NCL

- **Easier to use F77 code, but works with F90 code**

- **Need to isolate definition of input variables and wrap it with special comment statements:**

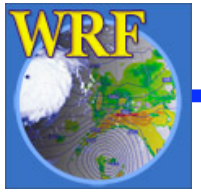
```
C NCLFORTSTART
C NCLEND
```

- **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 preallocate 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
```

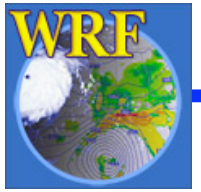


myTK.so - *Create & use in NCL script*

```
% WRAPIT myTK.f
```

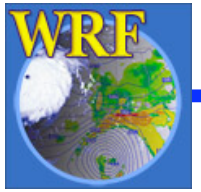
This will create a "myTK.so" file

```
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

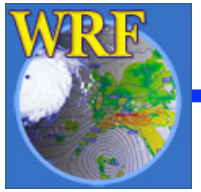
- **Can use simple FORTRAN 90 code**
- **Your FORTRAN 90 program may not contain any of the following features:**
  - pointers or structures as arguments,
  - missing/optional arguments,
  - keyword arguments, or
  - recursive procedure.



FORTRAN 90 code

myTK.f90

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


  pi(:,:,:)=(pres(:,:,:)/1000.)**(287./1004.)
  tk(i,j,k) = pi(:,:,:)*theta(i,j,k)

end subroutine compute_tk
```



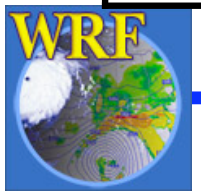
FORTRAN 90 code

myTK.f90

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


  pi(:,:,:)=(pres(:,:,:)/1000.)**(287./1004.)
  tk(i,j,k) = pi(:,:,:)*theta(i,j,k)

end subroutine compute_tk
```

myTK90.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
```

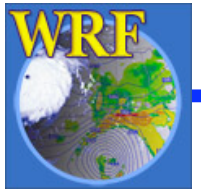


myTK90.so - *Create & use in NCL script*

```
% WRAPIT myTK90.stub  myTK.f90
```

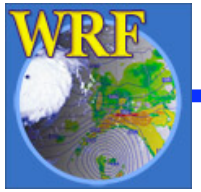
This will create a "myTK90.so" file

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
external myTK90 "./myTK90.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
       myTK90 :: compute_tk (tk,p,t,dim(2),dim(1),dim(0))
end
```

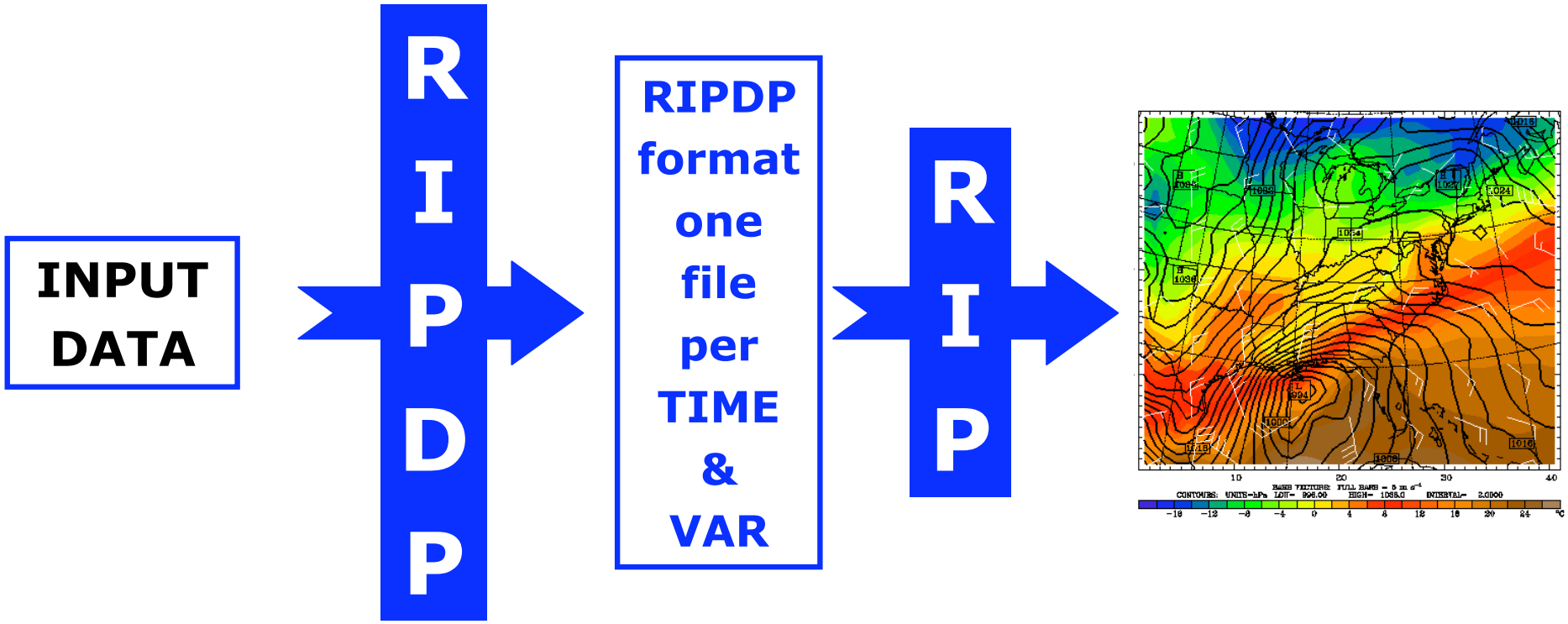


RIP4

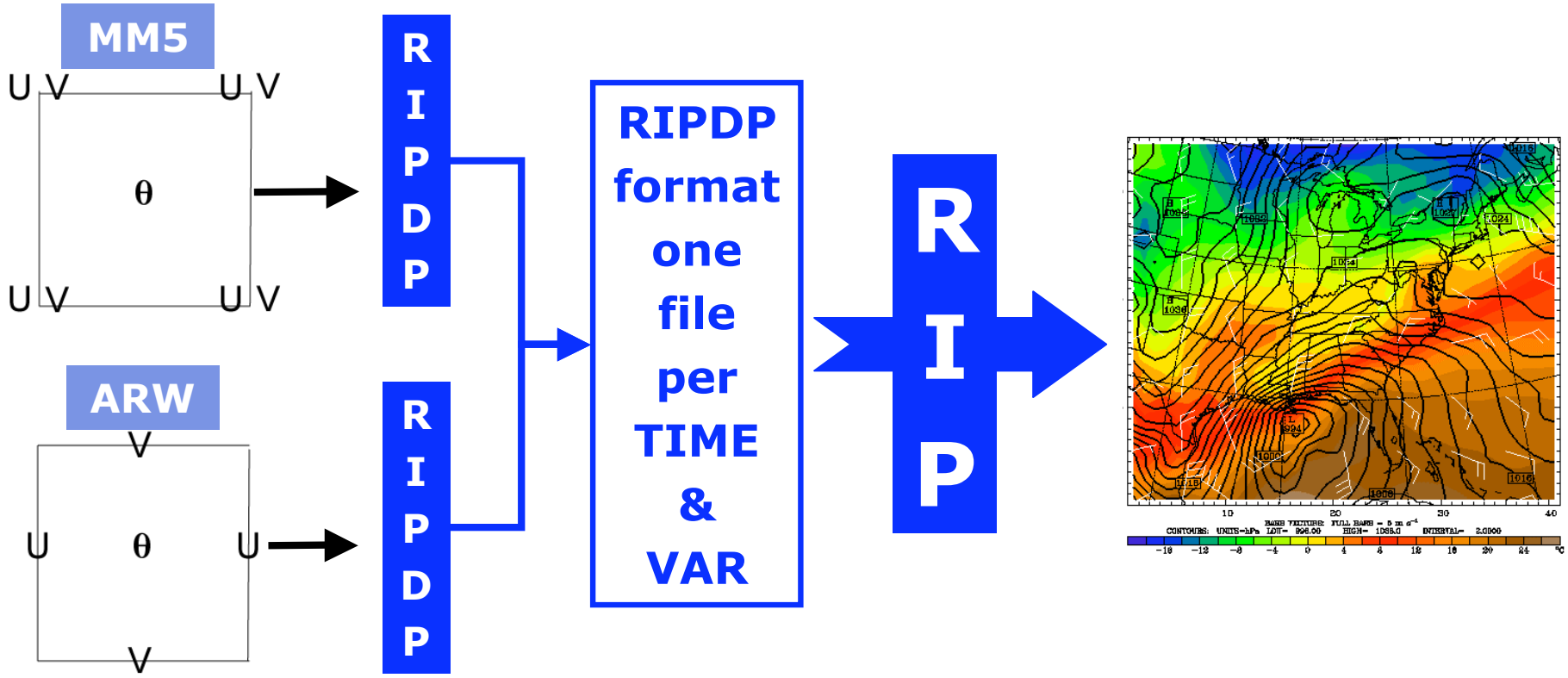
- **Read Interpolate Plot version 4**
- **Develop by Mark Stoelinga (UW/NCAR) & MMM/NCAR Staff**
- **Originally developed for the MM5 model**
- **Generate a number of graphical plots**
  - Horizontal, cross-section, skewT
- **Current Version: 4.5**
  - configure / compile



RIP4

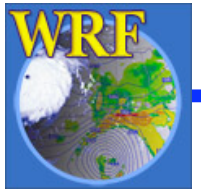


RIP4 - Grids


MM5
U V
U V
θ
U V
U V
ARW
V
U
θ
U
V
RIPDP
RIPDP
RIPDP format one file per TIME & VAR
RIP

General

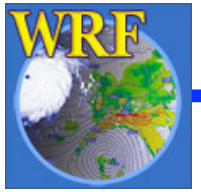
- **Requires NCAR Graphics low-level routines**
  - ***http://ngwww.ucar.edu***
- **NCL Version 5:**
  - ***http://www.ncl.ucar.edu***
  - Released November 2007
  - Combine NCL and NCAR Graphics
  - Open Source
  - Recommended
- **Download Code:**
  - ***http://www.mmm.ucar.edu/wrf/users/download/get_source.html***



General

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

- **OnLine Tutorial:**
  - ***http://www.mmm.ucar.edu/wrf/users/graphics/RIP4/RIP4.htm***



RIP4 on your computer

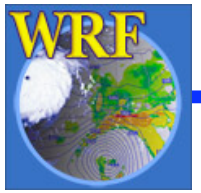
- **set environment variables**
  setenv RIP_ROOT /usr/$USER/RIP4 ***(rip_root)***
  setenv NCARG_ROOT /usr/local/ncarg ***(/usr/local/ncl)***

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

- **Compile**
  ./compile

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

ripdp_wrfarw



ripdp & rip

- **ripdp**
  - RIP Data Preparation
  - Converter
    RIPDP converts WRF formatted data (*netCDF*) into RIP format (*B - grid*)
  - Output
    RIPDP puts each Variable at each Time into a separate file
    Creates LOTS of files

- **rip**
  - Reads the output generated by *ripdp*
  - Makes use of a User Input File (UIF) *(rip_sample.in)* to control plots
  - Output
    X11, pdf, ps

Running ripdp

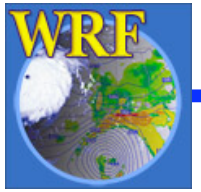
Optional

```
ripdp_wrfxxx   [–n namelist-file]          \
      <model_data_name>     [basic/all]   \
      <input_file1 input_file2>
```

Example:

```
ripdp_wrfarw   RIPDP/arw   all   wrfout*
```

use directory as part of the model_data_name

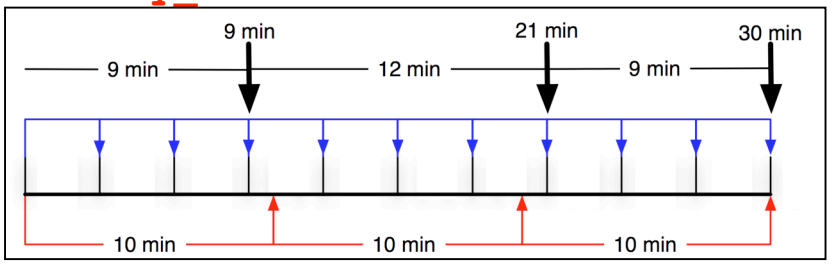


ripdp namelist

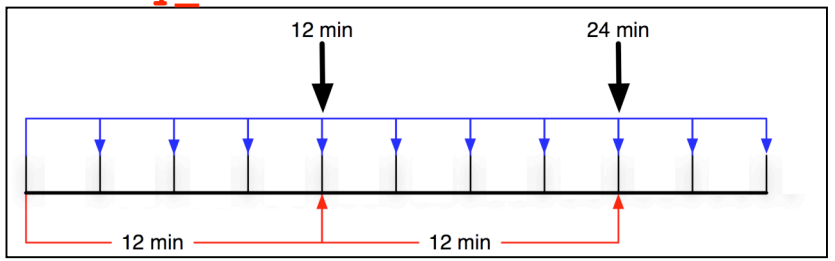
- **ptimes** *(times for ripdp to process)*
  - `0,1,2,3,4,5,6` (0,1,2,3,4,5,6)
  - `0,-6,1` (0,1,2,3,4,5,6)
  - `0,  2,-4,1,  6` (0, 2,3,4, 6)

- **tacc**: *input files not on exact times* [`time_step=180` (3 min)]

`history_interval=10`



`history_interval=12`



- **discard**: fields if '**all**' is selected on the command line
- **retain**: fields if '**basic**' is selected on the command line

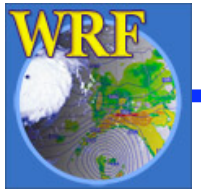
Running rip

- **Edit the User Input File (UIF)**

- **setenv NCARG_ROOT /usr/local/ncarg**

  setenv NCARG_ROOT /usr/local/ncl
  (if you installed NCL version 5)

- **setenv RIP_ROOT *your-rip-directory***
  - *Can overwrite this with **rip_root** in input namelist*



Running rip

created by ripdp

**rip [-f] model-data-set-name **

rip-execution-name

User Input File (UIF)

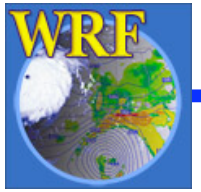
Example:

rip [-f] RIPDP/*xxx* rip_sample.in

use directory as part of the model_data_set_name

output

[rip_sample.out]
rip_sample.TYPE



rip UIF

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

Namelist controlling general parameters

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

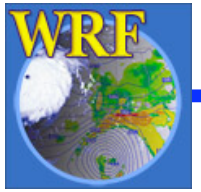
Namelist for trajectory calculations
Only used if itrajcalc=1, in userin namelist

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

Frame specification group (FSG)

Plot specification line (PSL)

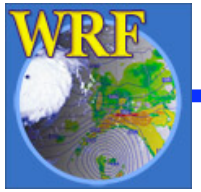
Plot Specification Table (PST)



rip namelist - *&userin*

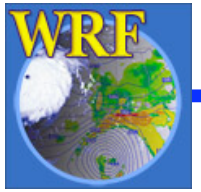
- **Use namelist to control**
  - processing times, intervals
  - title information
  - text quality on a plot
  - whether to do time series, trajectory, or to write output for Vis5D

- **Full explanation for namelist variables is available in the user document**



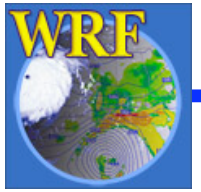
rip namelist - *&userin*

- **idotitle – first part of first title line**
- **titlecolor – color of title lines**
- **ptimes, ptimeunits – times to process**
- **tacc – tolerance for processing data**
- **timezone –display of local time**
- **iusedaylightrule – 1 applied, 0 not applied**
- **iinittime – plotting of initial time**
- **ivalidtime – plotting of valid time**
- **inearsth – plot times as 2 / 4 digits**
- **flmin, frmax, fbmin, ftmax – frame size**
- **ntextq – text quality**



rip namelist - *&userin*

- **ntextcd – text font**
- **fcoffset – 12 means, hour 12 WRF forecast is considered hour 0 by RIP**
- **idotser – generate time series output**
- **idescriptive – more descriptive titles**
- **icgmsplit – split metacode into several files**
- **maxfld – reserve memory for RIP (10-15)**
- **itrajcalc – 0, 1 ONLY when doing trajectory calculations *(use also namelist trajcalc)***
- **imakev5d – 0, 1 generate Vis5D data**
- **rip_root - override RIP_ROOT**
- **ncarg_root - output type: X11, cgm, pdf, ps**

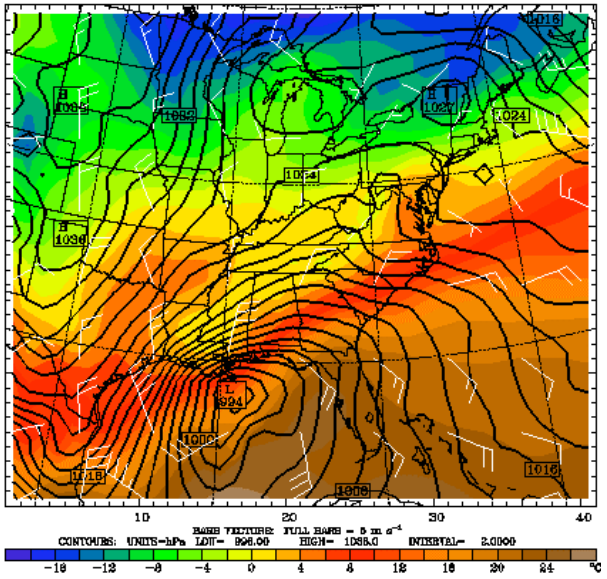


Creating a Plot

feld=
diagnostics - tmc
native - PSFC

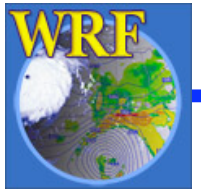
levs=2fb
levs=1,2,3
levs=800,500
levs=800,-300,100

```
==========================================
feld=tmc; ptyp=hc; vcor=s; levs=1fb; >
   cint=2; cmth=fill; >
   cosq=32,light.violet,-16,blue, >
   0,yellow,16,orange,32,light.gray
feld=slp; ptyp=hc; cint=2; linw=2
feld=uuu,vvv; ptyp=hv; vcmx=1; >
    colr=white;intv=5
feld=map; ptyp=hb
feld=tic; ptyp=hb
==========================================
```



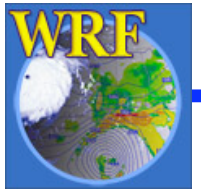
ARWpost

- **Converter**
  - Requires GrADS / vis5d to display data.
- **GrADS software only needed to display data.**
- **If vis5d output is required, vis5sd libraries are needed to compile the code.**
- **Generate a number of graphical plots**
  - Horizontal, cross-section, skewT, meteogram, panel

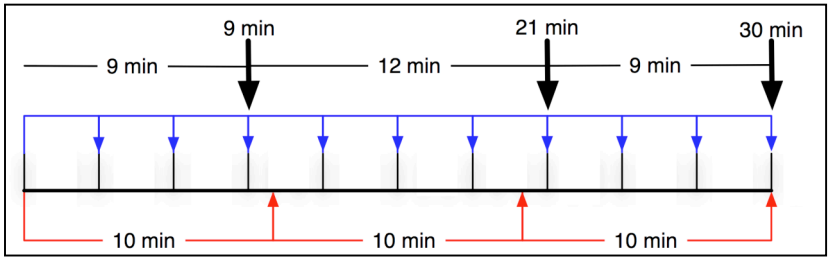


General

- **Download Code** (http://www.mmm.ucar.edu/wrf/users)
- **OnLine Tutorial**
  http://www.mmm.ucar.edu/wrf/users/graphics/ARWpost/ARWpost.htm
- **MUST have WRF compiled *(similar to WPS)***
  ./configure & ./compile
- **For GrADS output**
  - GrADS libraries only needed to display data *(freely available)*
  - http://grads.iges.org/grads/grads.html
- **For vis5d output**
  - vis5d libraries needed for compilation *(freely available)*
  - http://www.ssec.wisc.edu/~billh/vis5d.html

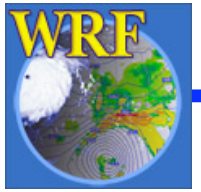


namelist.ARWpost

start_date ***end_date***	Start & end date Format: *YYYY-MM-DD_HH:mm:ss*
interval_seconds	Seconds between times to process. *Code will skip times not required. Data can be in multiple files.*
tacc	If model output is not at regular intervals, use closest time within *tacc* seconds of time requested. *(150 sec)* 
debug_level	Set high for extra information

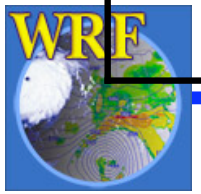
namelist.ARWpost

io_form_input	2=netCDF, 5=GRIB1
input_root_name	**Path** and **root** name of files to use as input. *Do not only provide directory name.* Can use wild characters.
output_root_name	Output root name. output_root_name.**dat** & output_root_name.**ctl**, OR output_root_name.**v5d**
output_type	Options are 'grads' (*default*) or 'v5d'
mercator_defs	Set to true if mercator plots are distorted

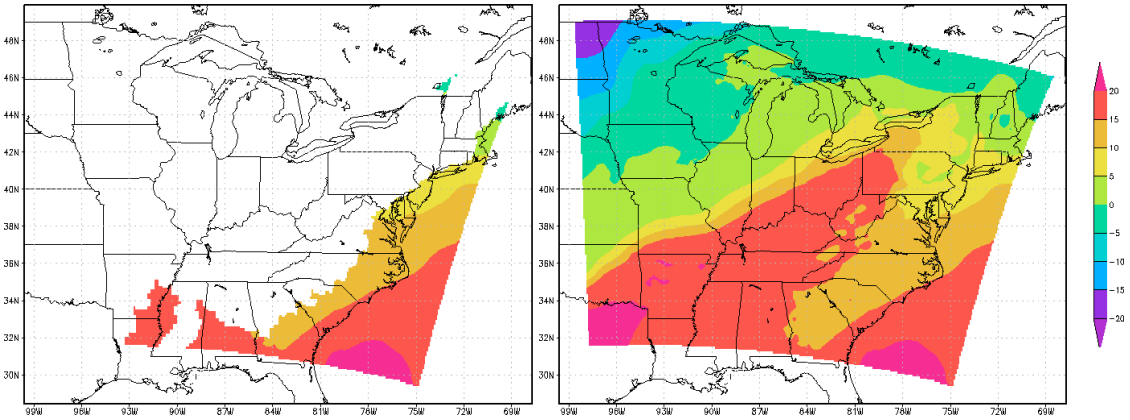


namelist.ARWpost

split_output	Split your GrADS output files into a number of smaller files *(a common .ctl file will be used for all .dat files).*
frames_per_outfile	If *split_output* is .**True**., how many time periods are required per output (.dat) file.
plot	Which fields to process. (*all, list, all_list)* Order has no effect, i.e., "all_list" and "list_all" "**list**" - list variables in "**fields**"
fields	Fields to plot. Only used is list was used in the "plot" variable. **Must use to generate diagnostics.**
Available diagnostics: cape, cin, mcape, mcin, clfr, dbz, max_dbz, geopt, height, lcl, lfc, pressure, rh, rh2, theta ,tc, tk, td, td2, slp, umet, vmet, u10m, v10m, wdir, wspd, wd10, ws10	



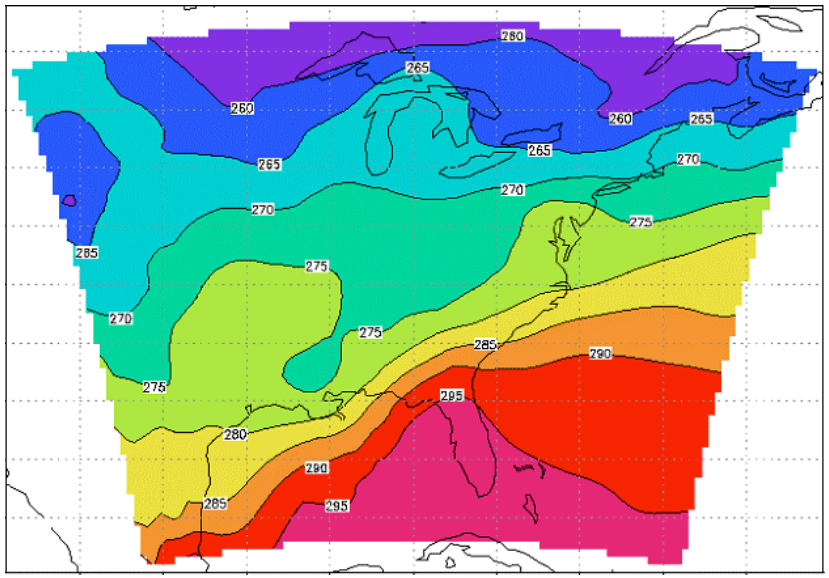
namelist.ARWpost

interp_method	0 = sigma levels, -1 = code defined "nice" height levels, 1 = user defined height or pressure levels
interp_levels	Only used if interp_method=1 Supply levels to interpolate to, in hPa (*pressure*) or km (*height above sea level*) Supply levels bottom to top
extrapolate	Extrapolate below ground *(default .false.)* 

GrADS specific notes

- **To display images**
  - Requires GrADS software
  - Freely available from: http://grads.iges.org/grads/grads.html
  - Documentation: http://grads.iges.org/grads/gadoc/index.html

- **Projection**
  - Data is plotted on model projection

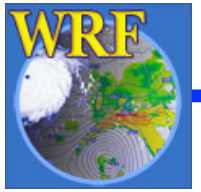


GrADS - .ctl file

```
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
```

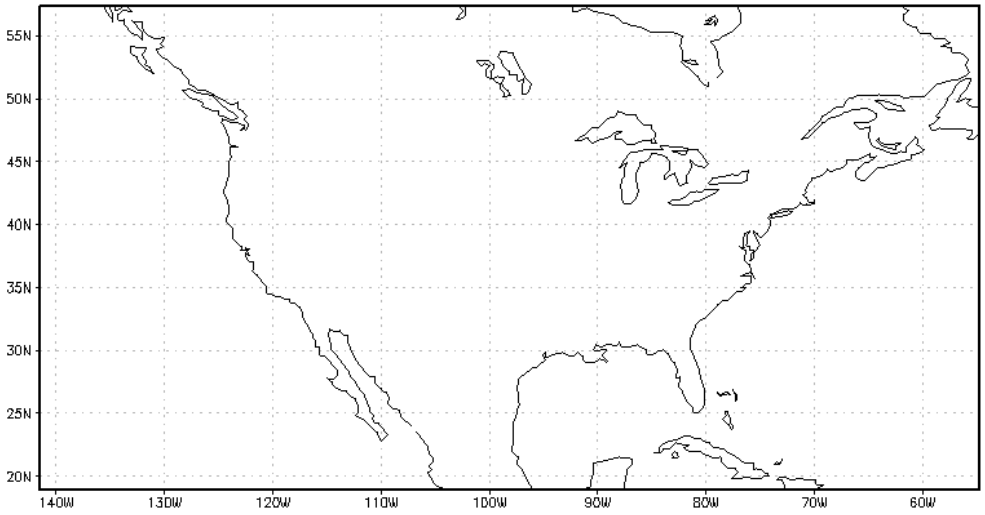
options byteswapped

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



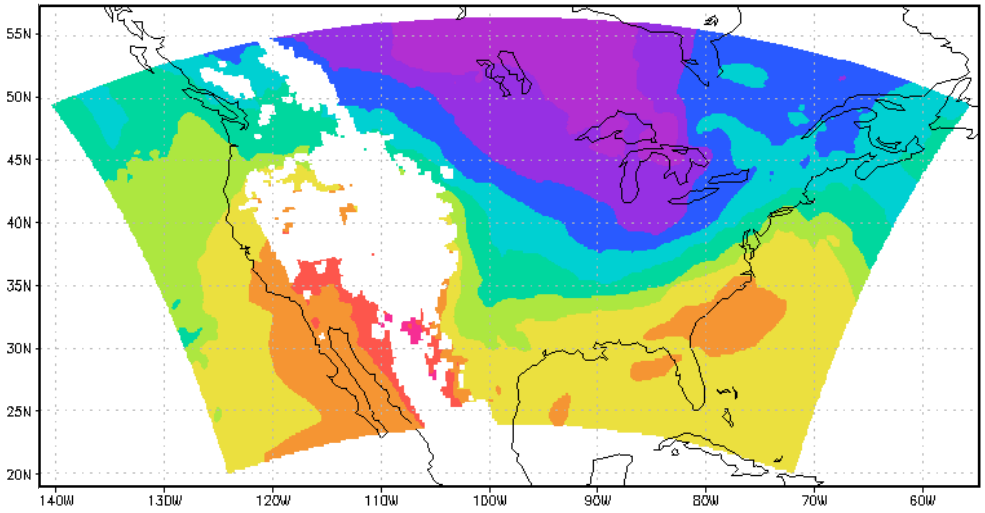
GrADS - .ctl file

```
dset ^test.dat
options byteswapped
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
```



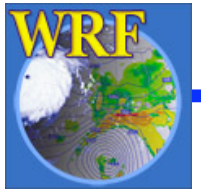
GrADS - .ctl file

```
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
```



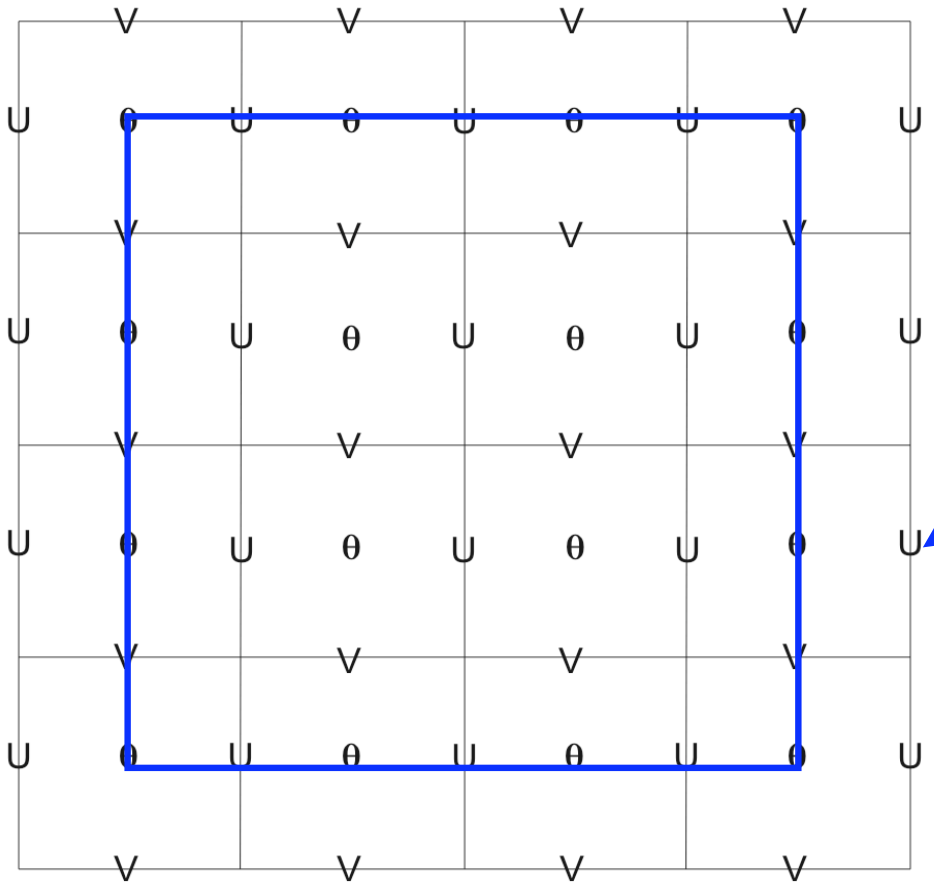
GrADS conversion - question

- **Why is a converter needed if GrADS can display netCDF files?**
  - Can only display model surface coordinates
  - Cannot interpolate to height or pressure levels
  - All diagnostics must be added via GrADS script files
  - GRIB1 model output can also be read directly by GrADS, but above issues are still valid
  - For GRIB1, there is also a stagger problem



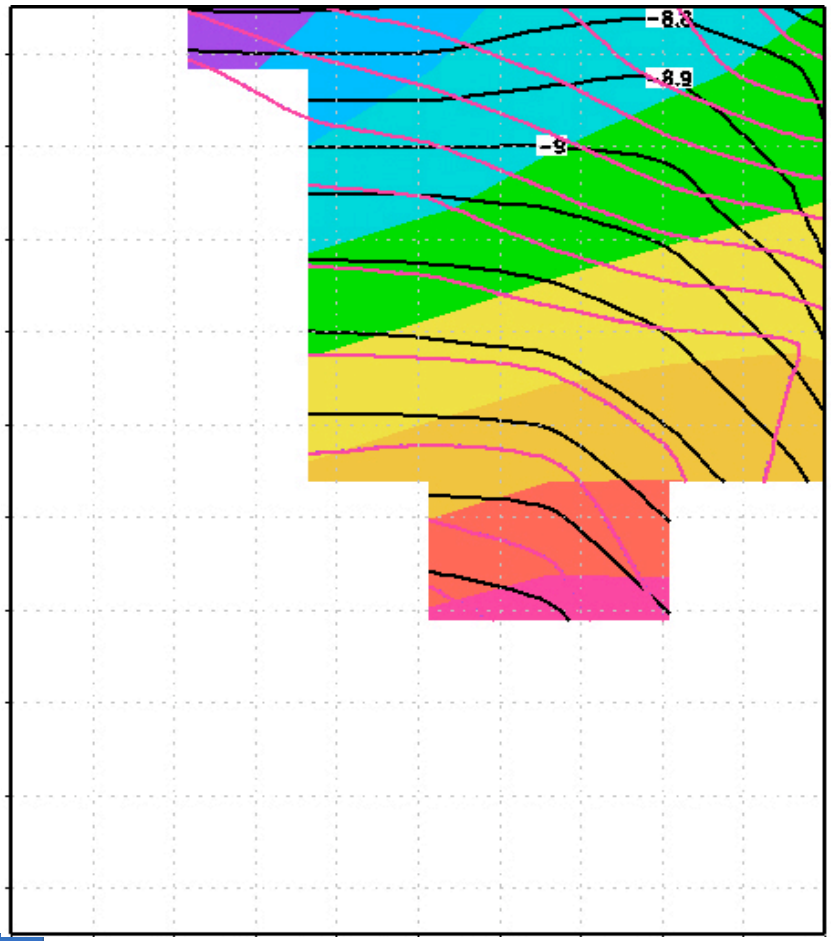
GrADS conversion - question

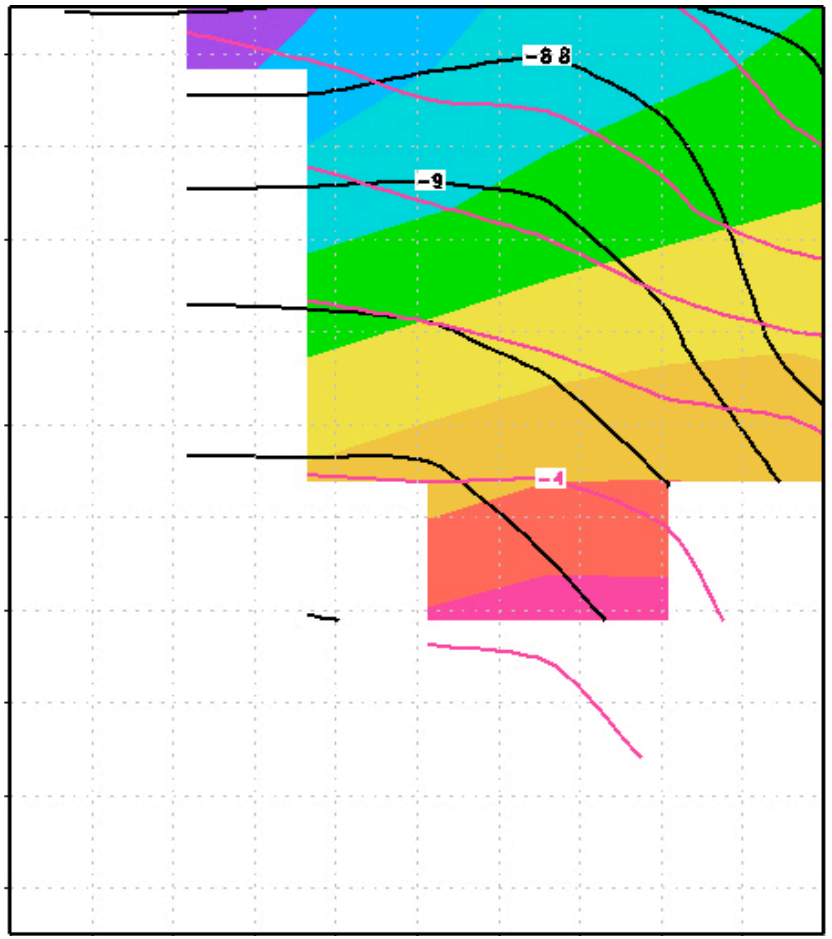
- **Why is a converter needed if GrADS can display netCDF files?**



Staggering

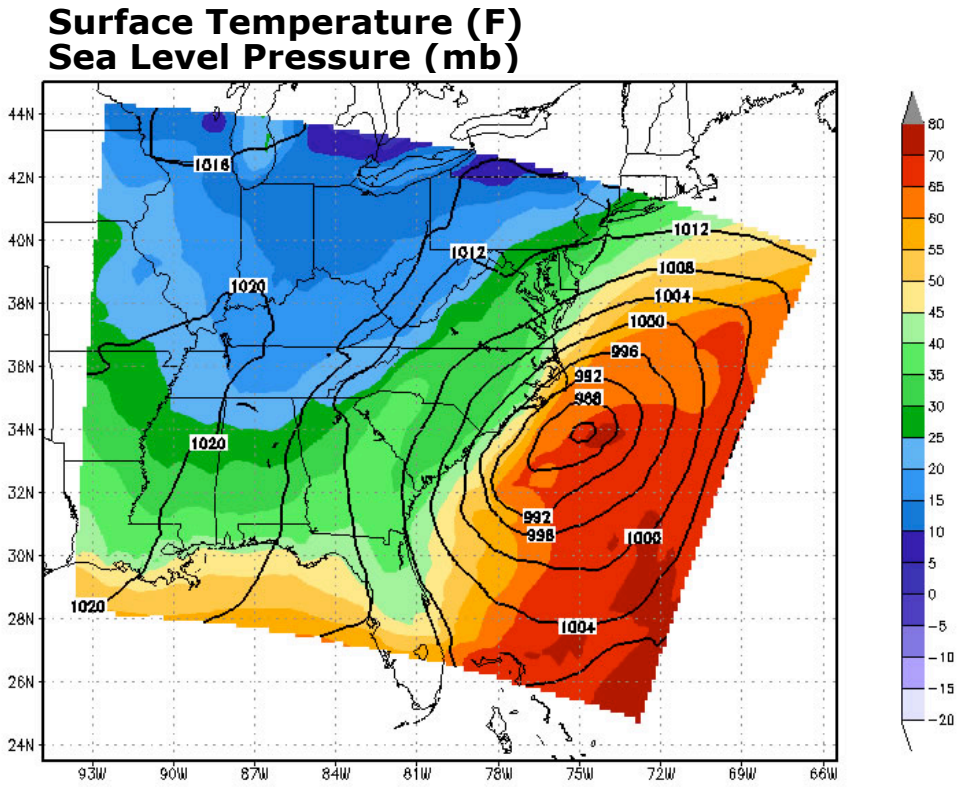
shaded=T ; black=U ; red=V





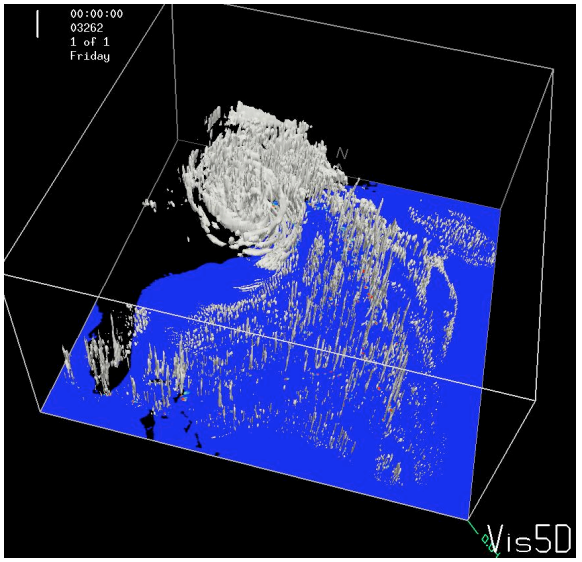
Creating a Plot

```
open em_real.ctl
set mpdset hires
set display color white

define tf=1.8*tc + 32
set gxout shaded
set z 1
d tf
run cbar.gs

set gxout contour
set ccolor 1
set cint 4
d slvl
```

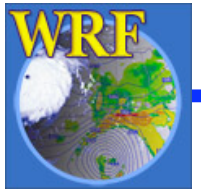


vis5d specific notes

- **vis5d is a three-dimensional visualization software**
- **vis5d is free and can be downloaded from:**
  http://www.ssec.wisc.edu/~billh/vis5d.html
- **Run**
  vis5d output_root_name.v5d
- **Graphical Interface**



Other Post-processing Tools

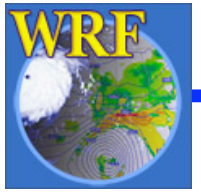


VAPOR

Computational and Information Systems Laboratory
National Center for Atmospheric Research

Visualization and Analysis Platform for Oceanic, atmospheric and solar Research

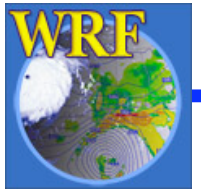
Alan Norton
alan@ucar.edu
vapor@ucar.edu
National Center for Atmospheric Research



WRF in VAPOR

Computational and Information Systems Laboratory
National Center for Atmospheric Research

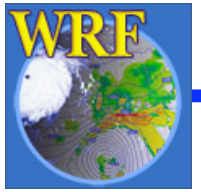
- **Interactive 3D visualization of WRF-ARW data *(wrfout files only)***
- **WRF functionality has been added in v1.2**
- **Available free on Linux, Windows, Mac**
- **Interactive rendering and animation (using GPU acceleration)**
- **Simple 2-step data conversion from WRF output to VAPOR**
  - wrfvdfcreate & wrf2vdf
- **Volume rendering**
- **Intuitive color/opacity editor**
- **Isosurface rendering**



WRF in VAPOR

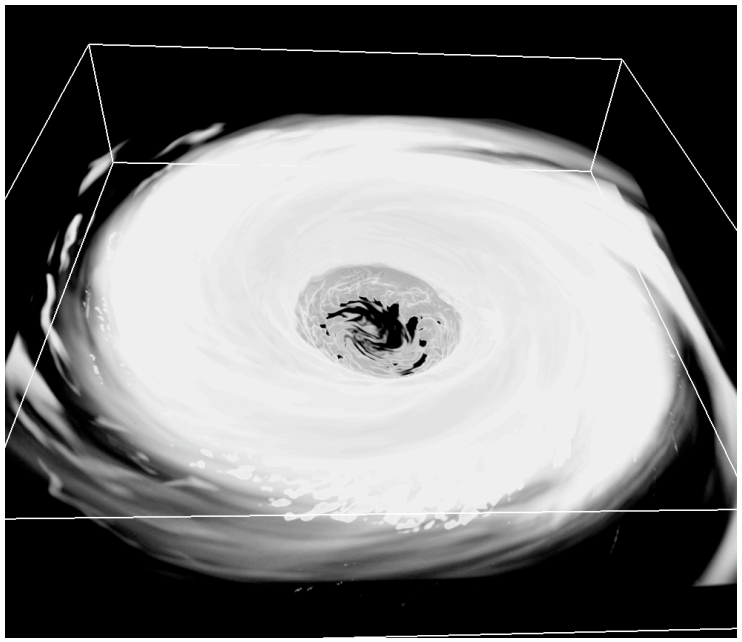
Computational and Information Systems Laboratory
National Center for Atmospheric Research

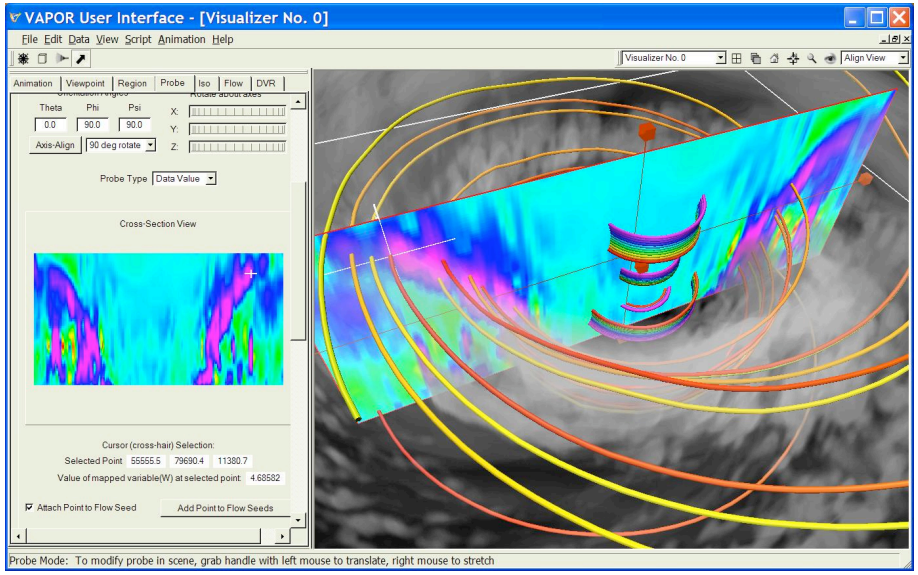
- **Steady and unsteady flow integration**
- **Interactive seed placement**
- **Data probing**
- **Contour plotting**
- **Terrain surface image render**
- **Interactive performance on terabyte datasets**
- **Downloads, documentation, examples at: http://www.vapor.ucar.edu**
- **http://www.vapor.ucar.edu/doc/WRFsupport.pdf**

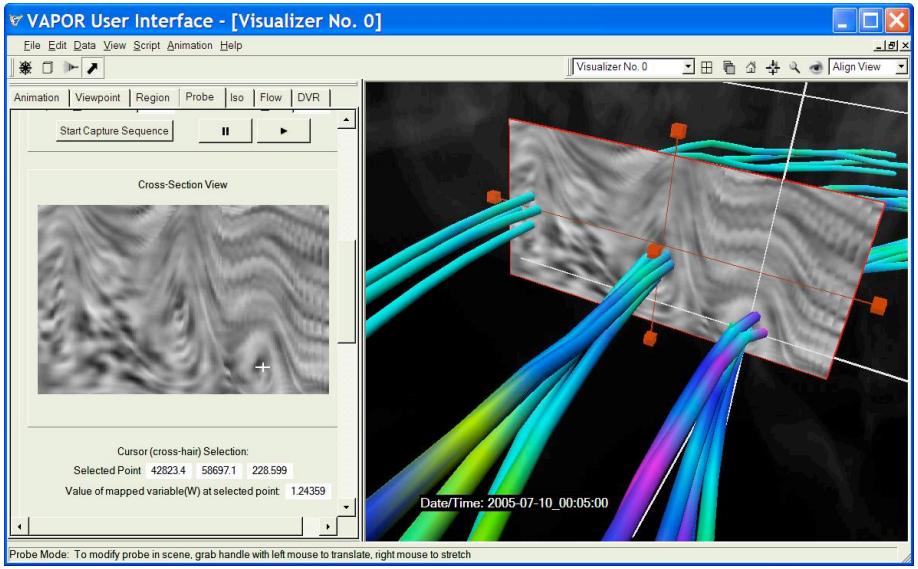


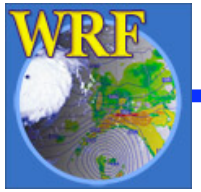
WRF in VAPOR

Computational and Information Systems Laboratory
National Center for Atmospheric Research









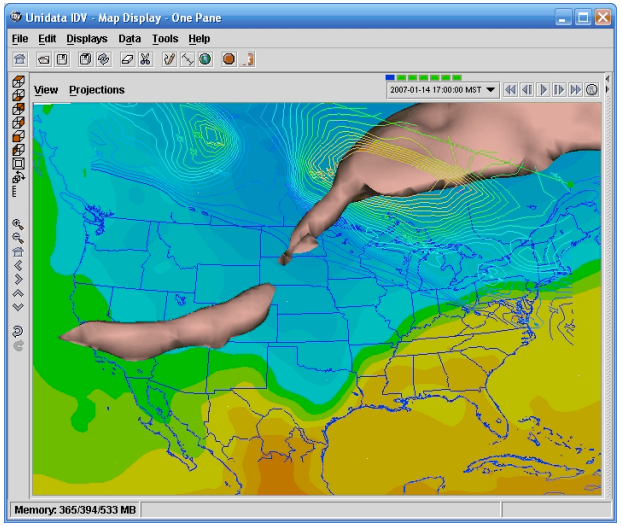
IDV

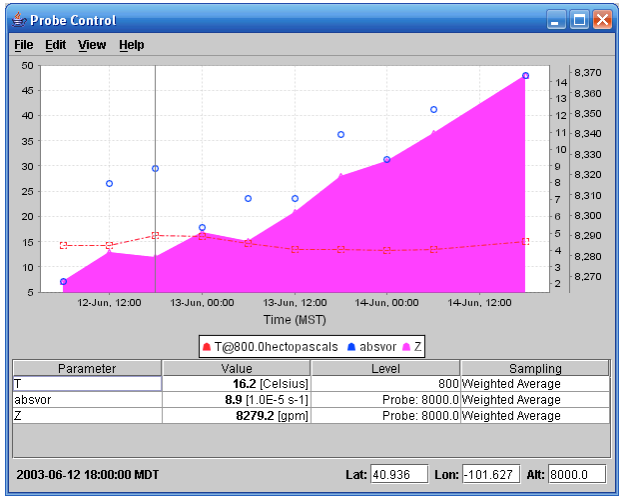
Integrated Data Viewer

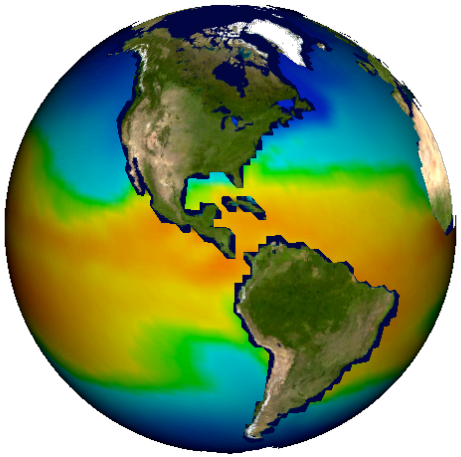
Don Murray and Jeff McWhirter
Unidata Program Center/UCAR

What is the IDV?

- **Visualization and analysis tool for geoscience data developed and supported by Unidata**
- **Freely available Java™ framework and application**
- **Integrated 2D/3D displays of a wide range of data**
- **Built on VisAD library**

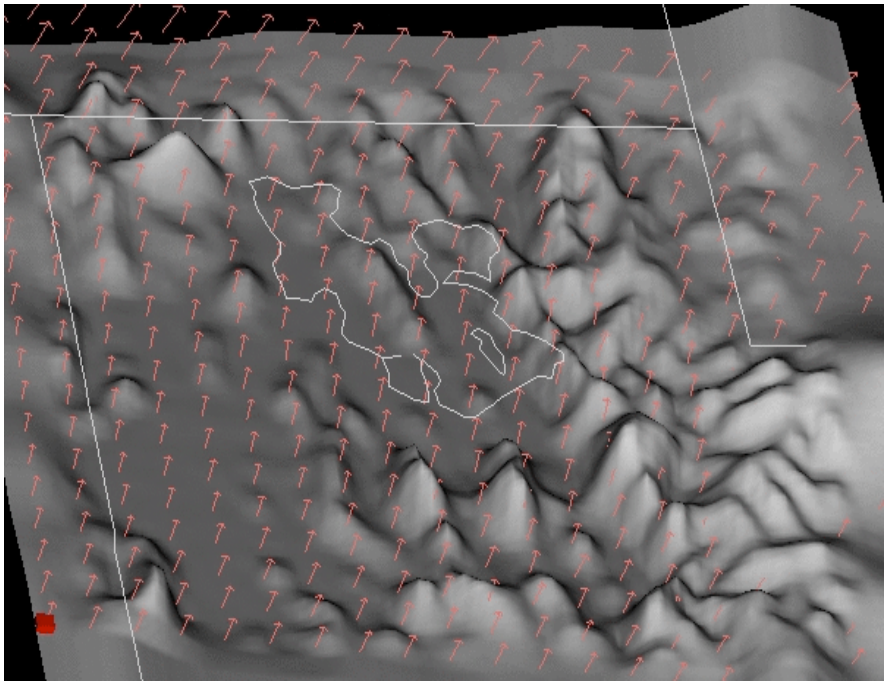






IDV Strengths

- **Easy to download and install on any platform**
- **Remote and local access to datasets**
- **2D/3D visualization**
- **Bundle mechanism**
- **Support for multi-disciplinary datasets integrated from a variety of sources**
- **Flexible framework supports customization (GEON-IDV, field projects, McIDAS-V)**
- **Extensive documentation**
- **Community driven development**



Model simulation of wind, isentropic potential vorticity and low level moisture flow over the Great Salt Lake basin

Supported Data Sources

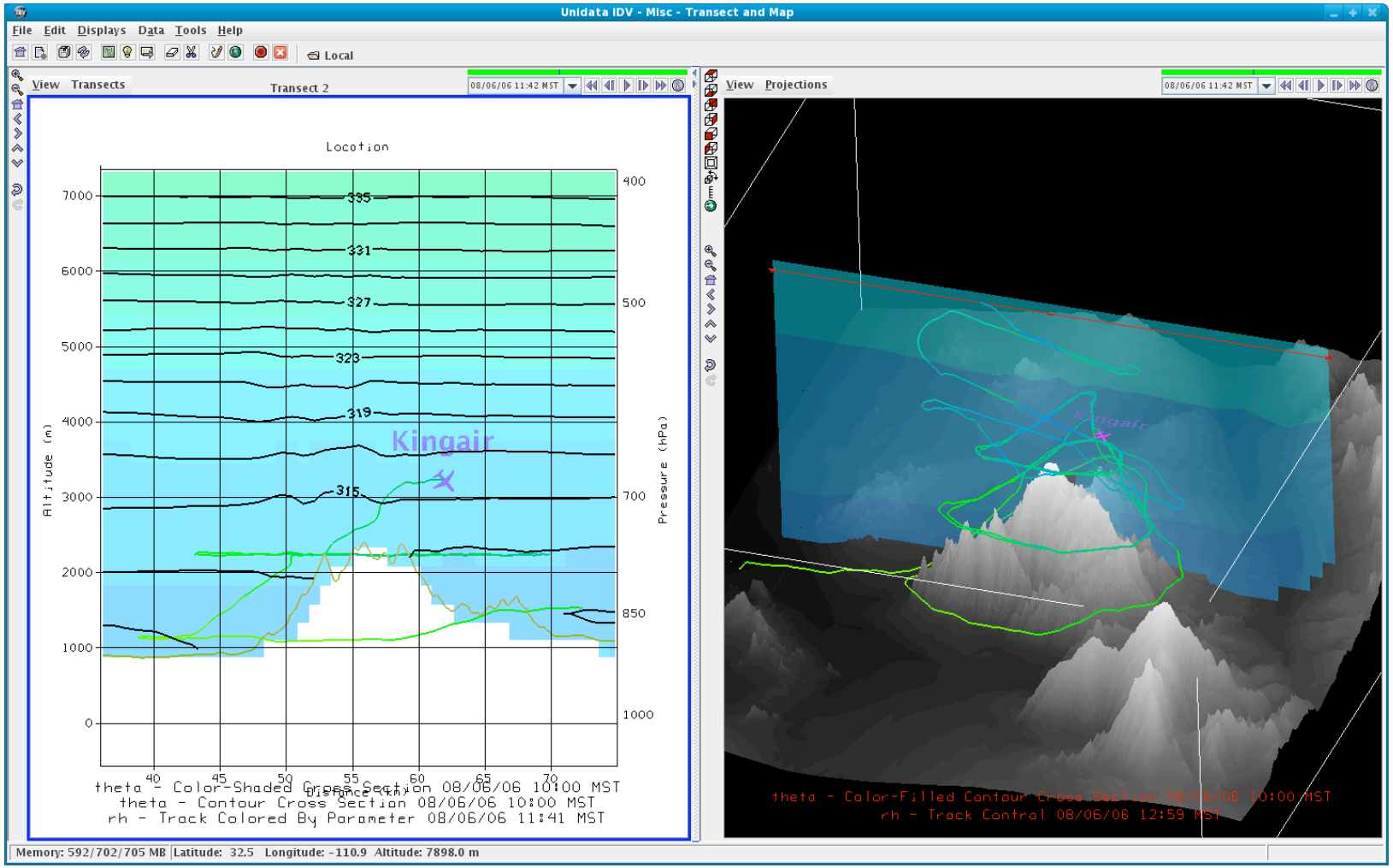
- **Data Types:**
  - Gridded model output
  - Satellite imagery
  - Radar data
  - Point observations
  - Balloon soundings
  - NOAA Profiler Network winds
  - Aircraft Tracks
  - Fronts
  - GIS data (WMS, shapefile)
  - Quick Time movies
  - Web Cams
- **Vertical Coordinates**
  - Pressure
  - Height/Depth
  - Other (2D only)
- **Sample of Supported Formats:**
  - netCDF
  - GRIB
  - Vis5D
  - KML
  - CSV
  - GEMPAK grid
  - ADDE
- **Access Methods:**
  - Local files
  - HTTP
  - ADDE, TDS and OPeNDAP servers
  - WMS

ADDE = Abstract Data Distribution Environment
TDS (THREDDS) = Thematic Realtime Environmental Distributed Data Services

Commonly Used Displays

- **Contours**
- **Flow vectors**
- **Vertical Cross Sections**
- **Isosurfaces**
- **Volume Rendering**
- **Charts**
- **Data probes**
- **Non-lat/lon projections**
- **Multiple views (e.g. Plan and cross section)**
- **Subsurface (mantle, ocean)**

CuPIDO Obs with WRF Simulation



Data source: Bart Geerts, U. Wyoming

For Further Information

- **Integrated Data Viewer homepage**
  - http://www.unidata.ucar.edu/software/idv
- **RAMADDA homepage**
  - http://www.unidata.ucar.edu/software/ramadda/
- **VisAD homepage**
  - http://www.ssec.wisc.edu/~billh/visad.html
- **All IDV questions/comments**
  - support-idv@unidata.ucar.edu