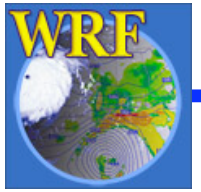
Post-processing Tools: NCL

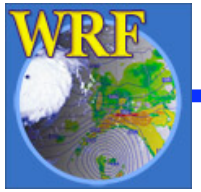
(WRF-ARW Only)

Cindy Bruyère

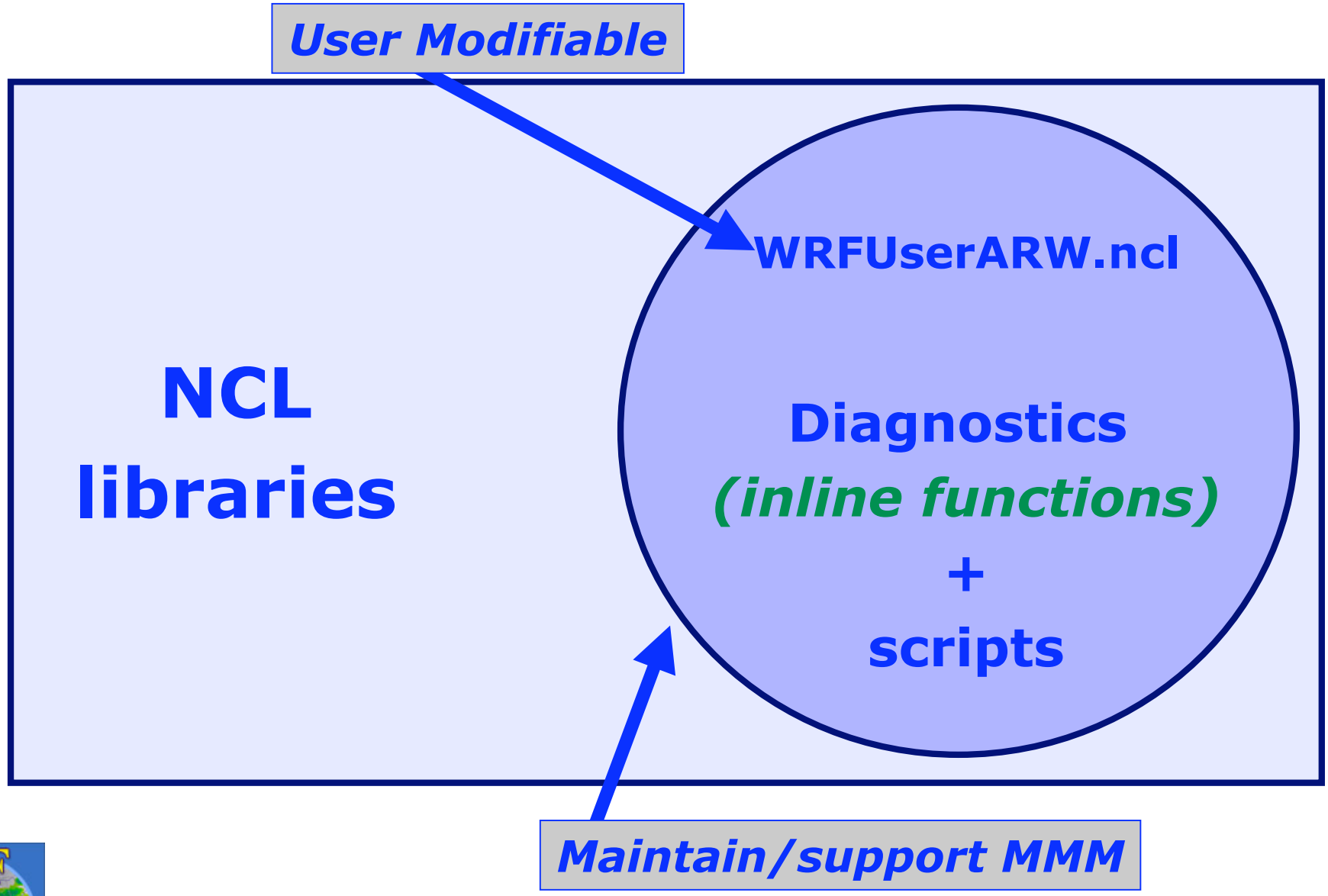


NCL

- **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



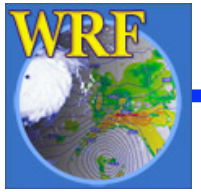
NCL & WRF


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

Home Directory → ~/.hluresfile ← Very Important

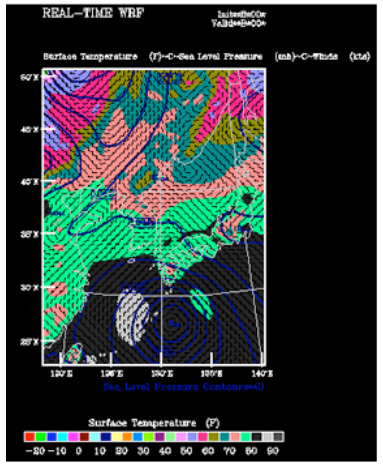
- **Required by NCL libraries**
- **Must be in your "~/" directory *(home directory)***
- **Control**
  - color table ; font
  - white/black background
  - size of plot
  - control characters

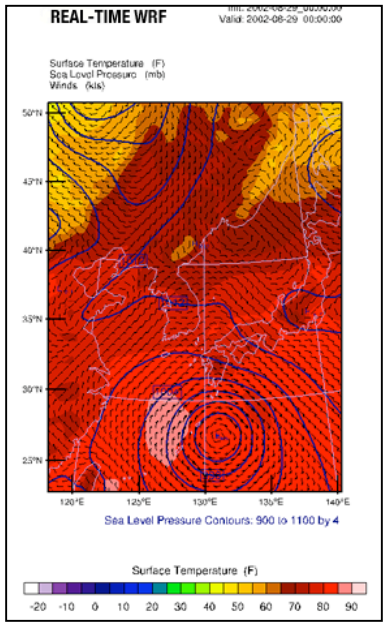
- **http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml**



~/.hluresfile

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

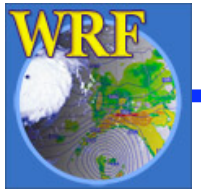




http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/.hluresfile

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*

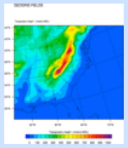
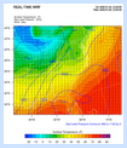
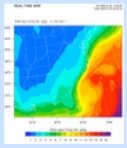
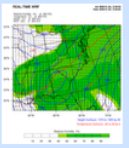
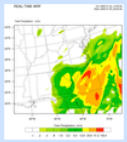
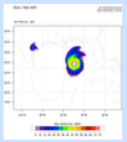
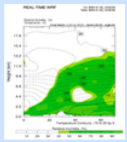
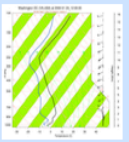
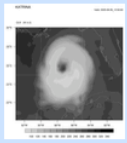
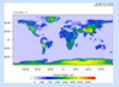
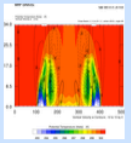


Generate Plots: *A good start - OnLine Tutorial*

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

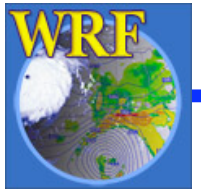
wrf/OnLineTutorial/Graphics/

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

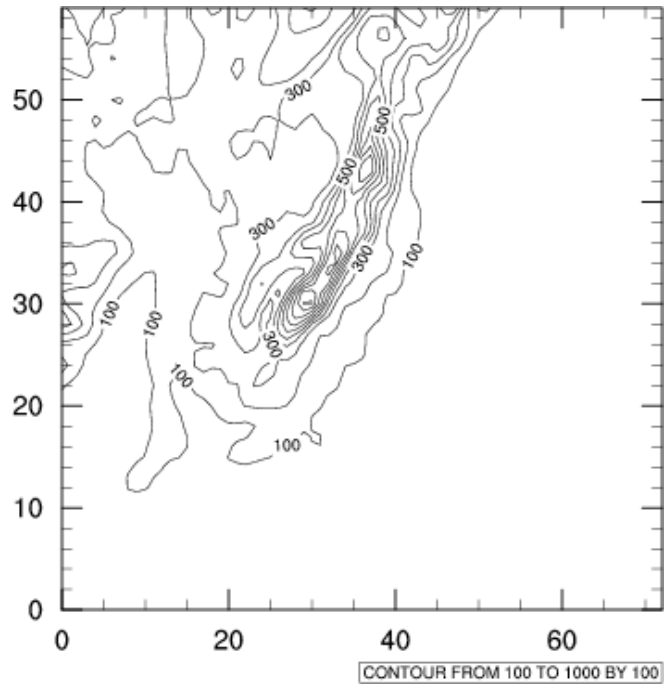
Creating a Plot : NCL script

```
load ncl library scripts

begin

  ; Open graphical output
  ; Open input file(s)

  ; Read variables

  ; Set up plot resources & Create plots
  ; Output graphics

end
```



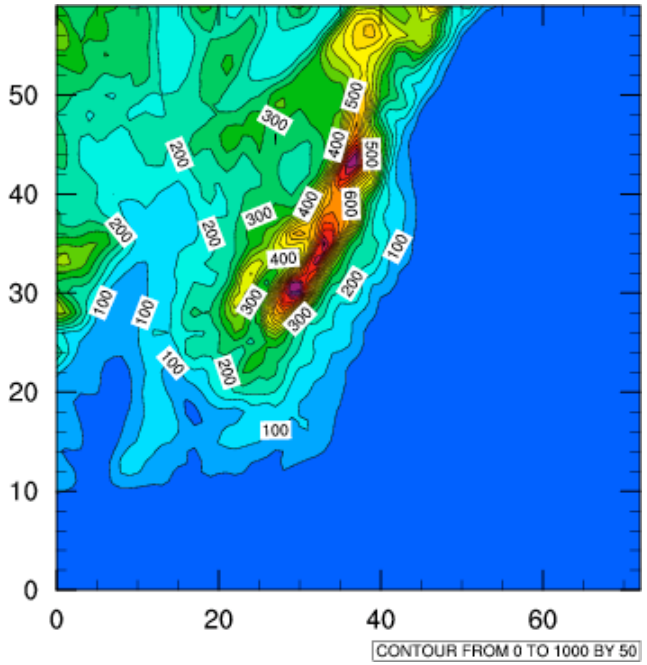
Generate Plots

```
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

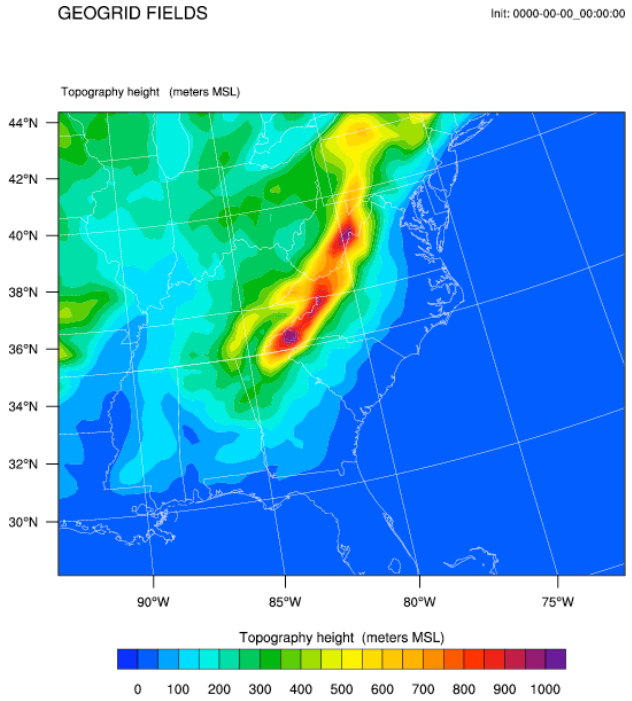
```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"

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.*

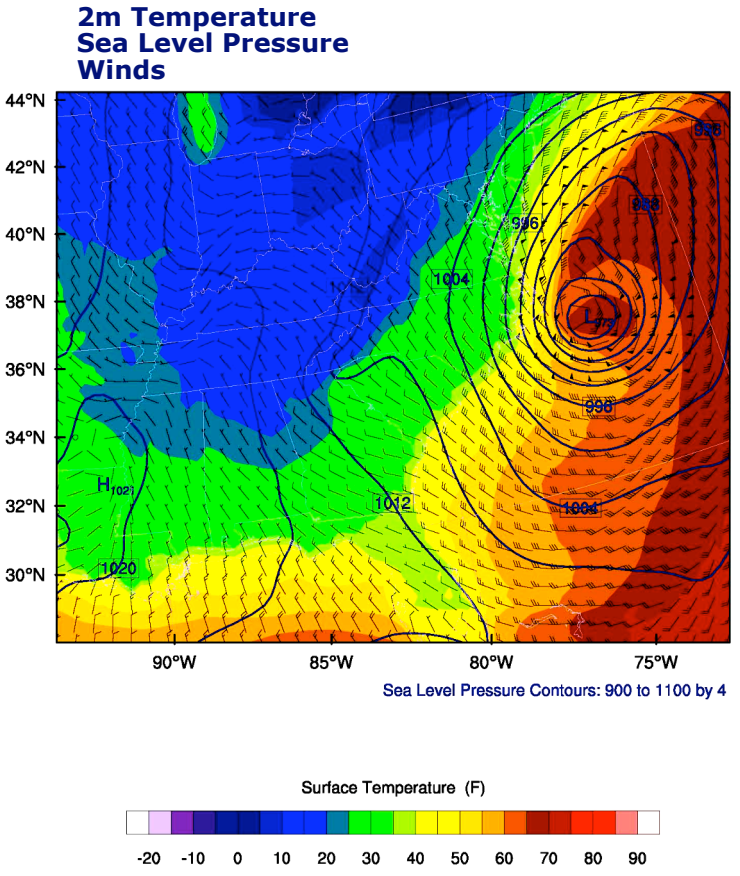
Generate Plots

```
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)
```



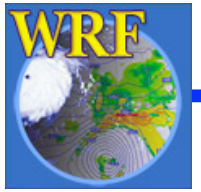
NCL and WRF_NCL

- **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)
```

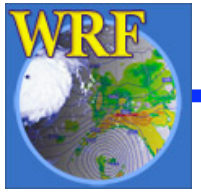


Special WRF Functions

- **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)

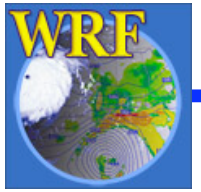
  avo/pvo: Absolute/Potential Vorticity,
  cape_2d: 2D mcape/mcin/lcl/lfc, **cape_3d**: 3D cape/cin,
  dbz/mdbz: Reflectivity (3D and max),
  geopt/geopotential: Geopotential,
  p/pres/pressure: Pressure, **rh/rh2**: Relative Humidity (3D and 2m),
  slp: Sea Level Pressure, **td/td2**: Dew Point Temperature (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)



Diagnostics

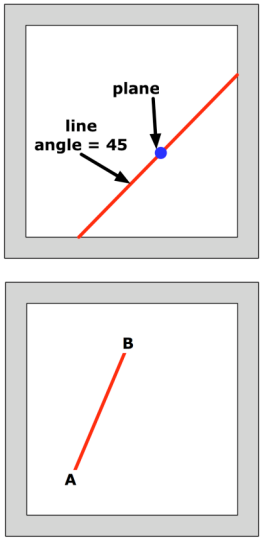
```
PH      = nc_file->PH(5,:,:,:)
PHB     = nc_file->PHB(5,:,:,:)
   PH     = ( PH + PHB ) / 9.81
   z = wrf_user_unstagger(PH,PH@stagger)

P       = nc_file->P(5,:,:,:)
PB      = nc_file->PB(5,:,:,:)
   P = P + PB

T       = nc_file->T(5,:,:,:)
   T = T + 300.
   tk = wrf_tk( P , T )

QVAPOR = nc_file->QVAPOR(5,:,:,:)
   QVAPOR = QVAPOR > 0.000


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

```
slp = wrf_user_getvar(nc_file,"slp",5)
```

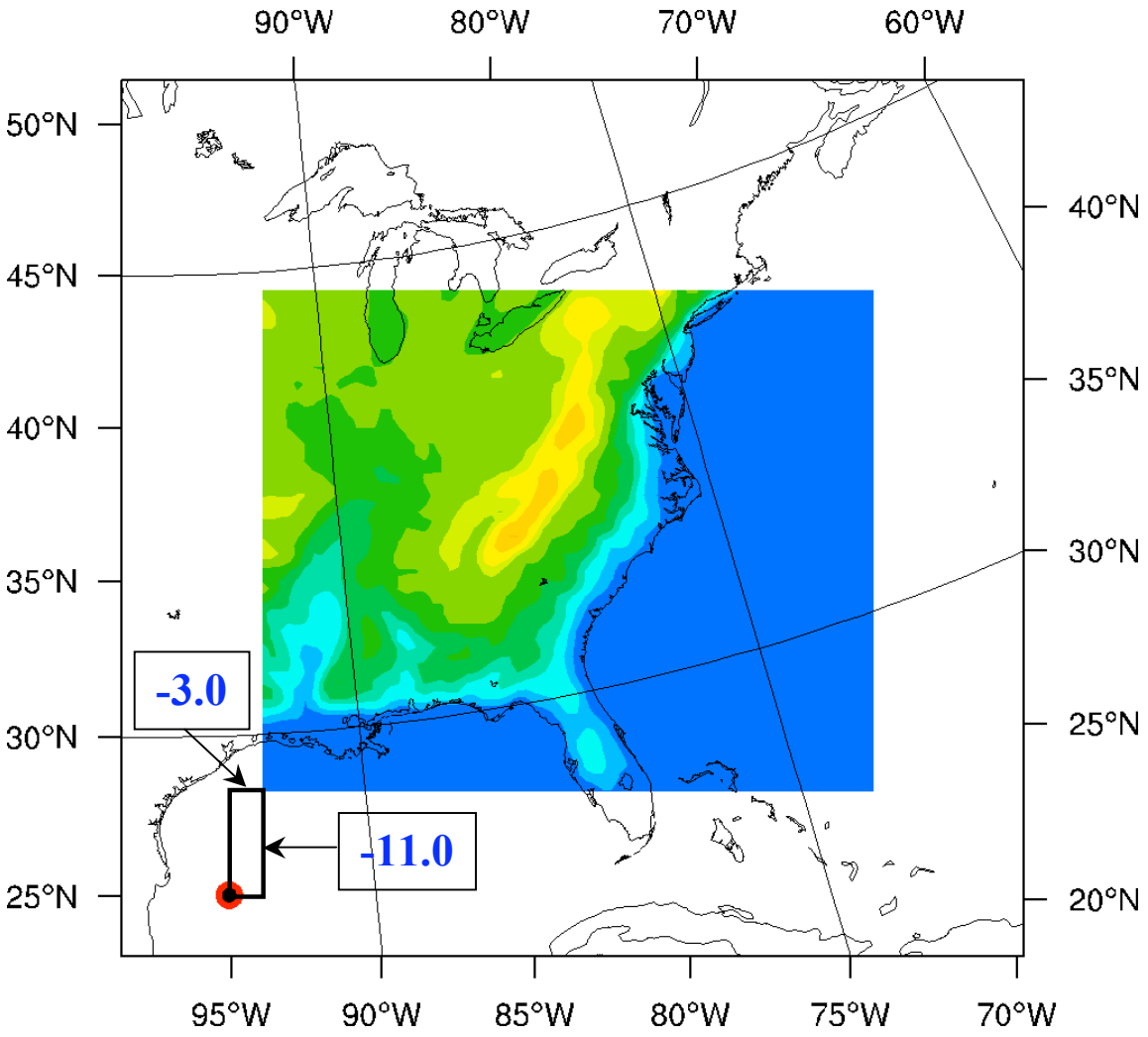


Special WRF Functions

- **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

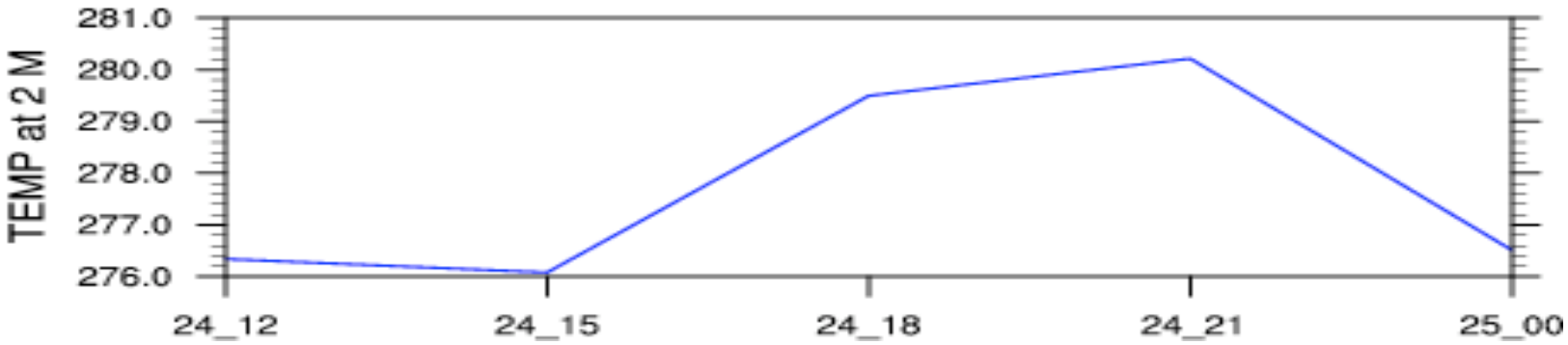


wrf_user_ll_to_ij



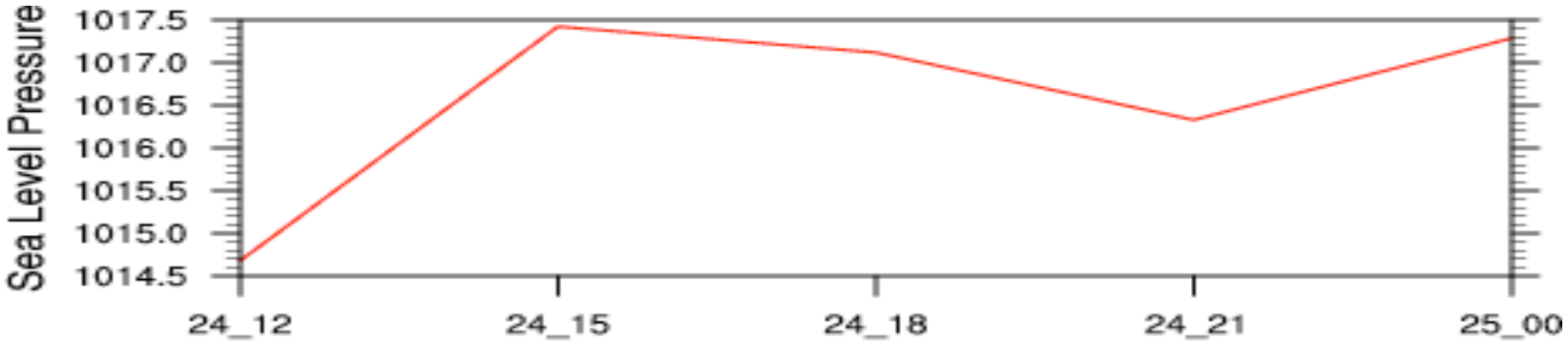
Time Series

```
   locij = wrf_user_ll_to_ij(a, -87., 32.5, llres)
locij = locij - 1
locX = locij(0)
locY = locij(1)

t2_point  = a->T2(:,locY,locX)
t2_plot   = gsn_csm_xy(wks,taus,t2_point,t2_res)
```



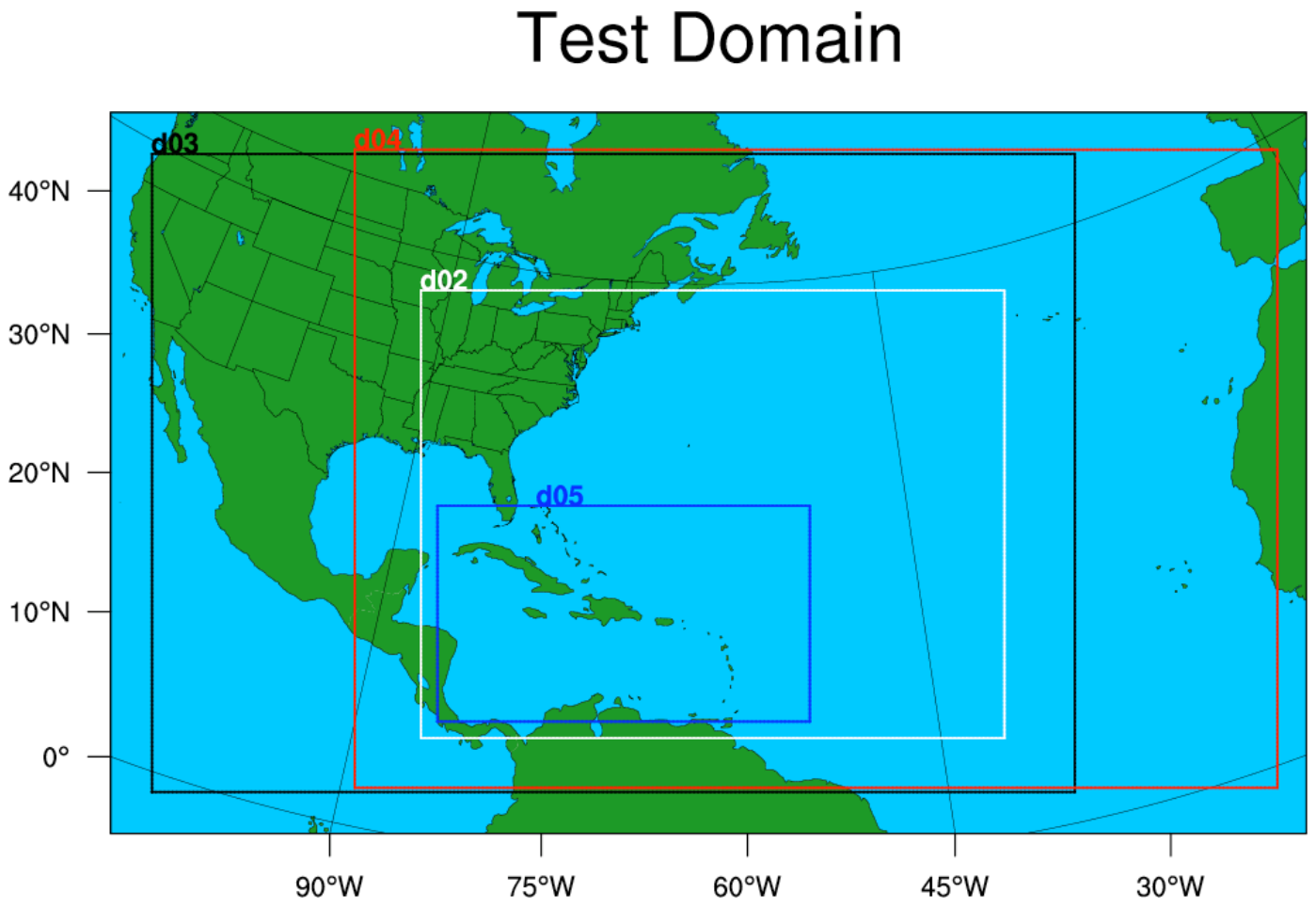
Time Series

```
   locij = wrf_user_ll_to_ij(a, -87., 32.5, llres)
locij = locij - 1
locX = locij(0)
locY = locij(1)

slp   = wrf_user_getvar(a,"slp",-1)
slp_point  = slp(:,locY,locX)
slp_plot  = gsn_csm_xy(wks,taus,slp_point,t2_res)
```

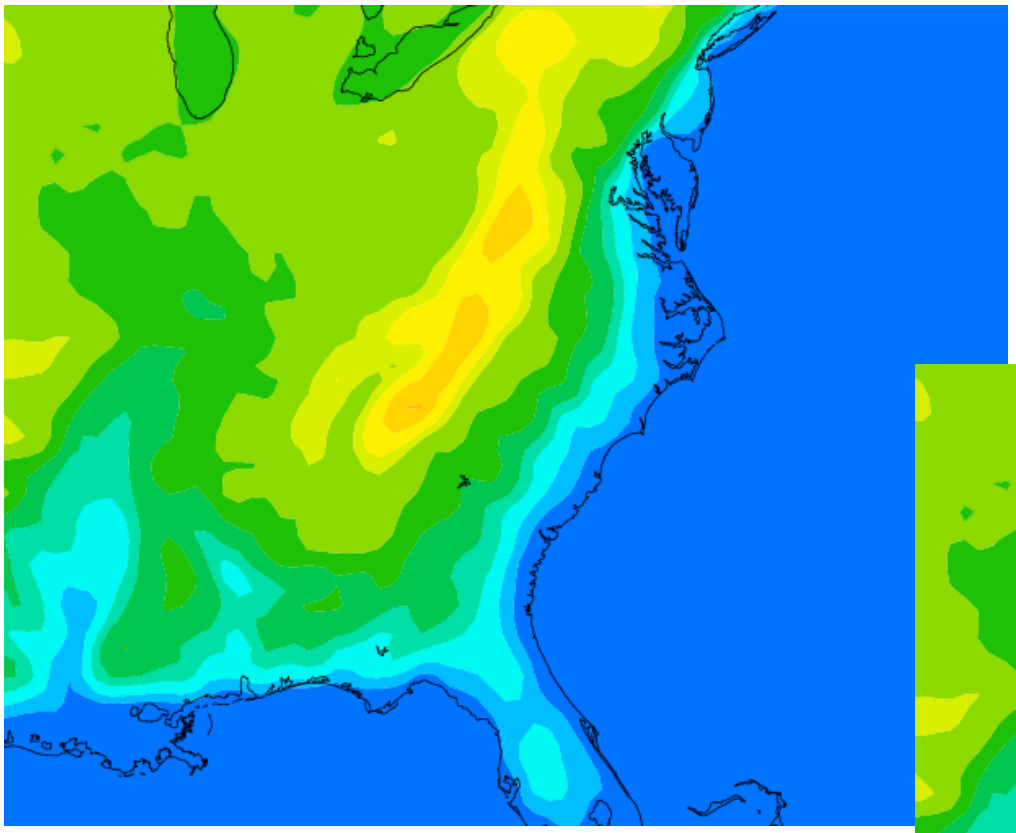


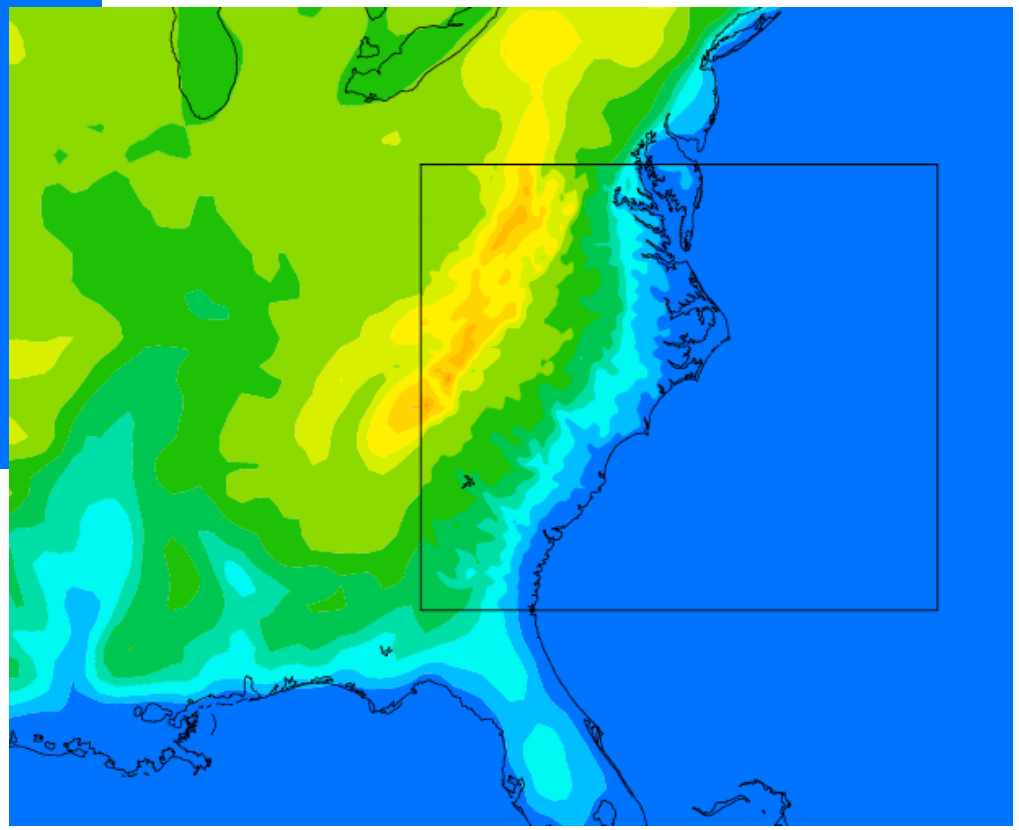
Domain Design

mp = wrf_wps_dom (wks, mpres, lnres, txres)

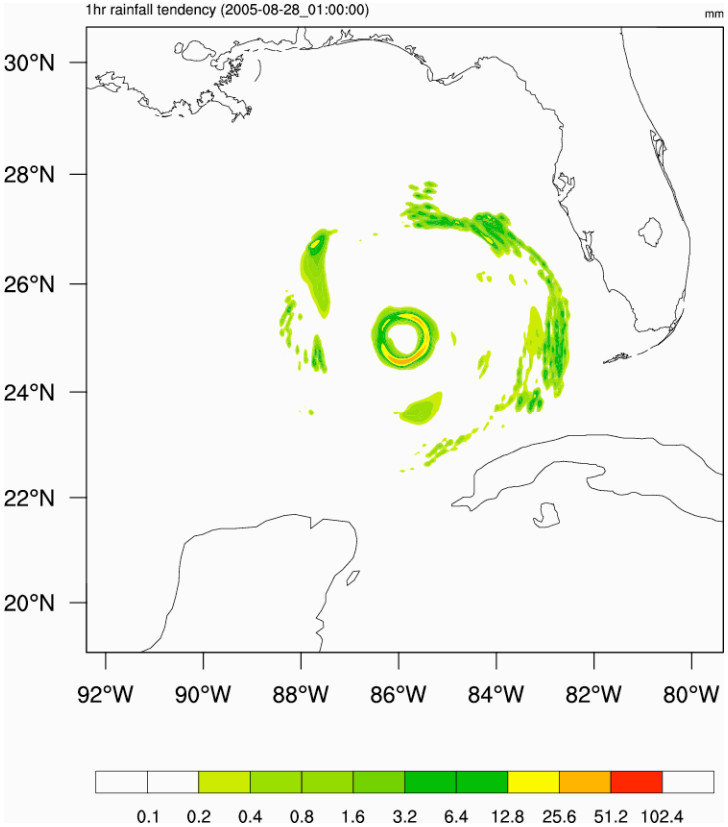


Over Domains

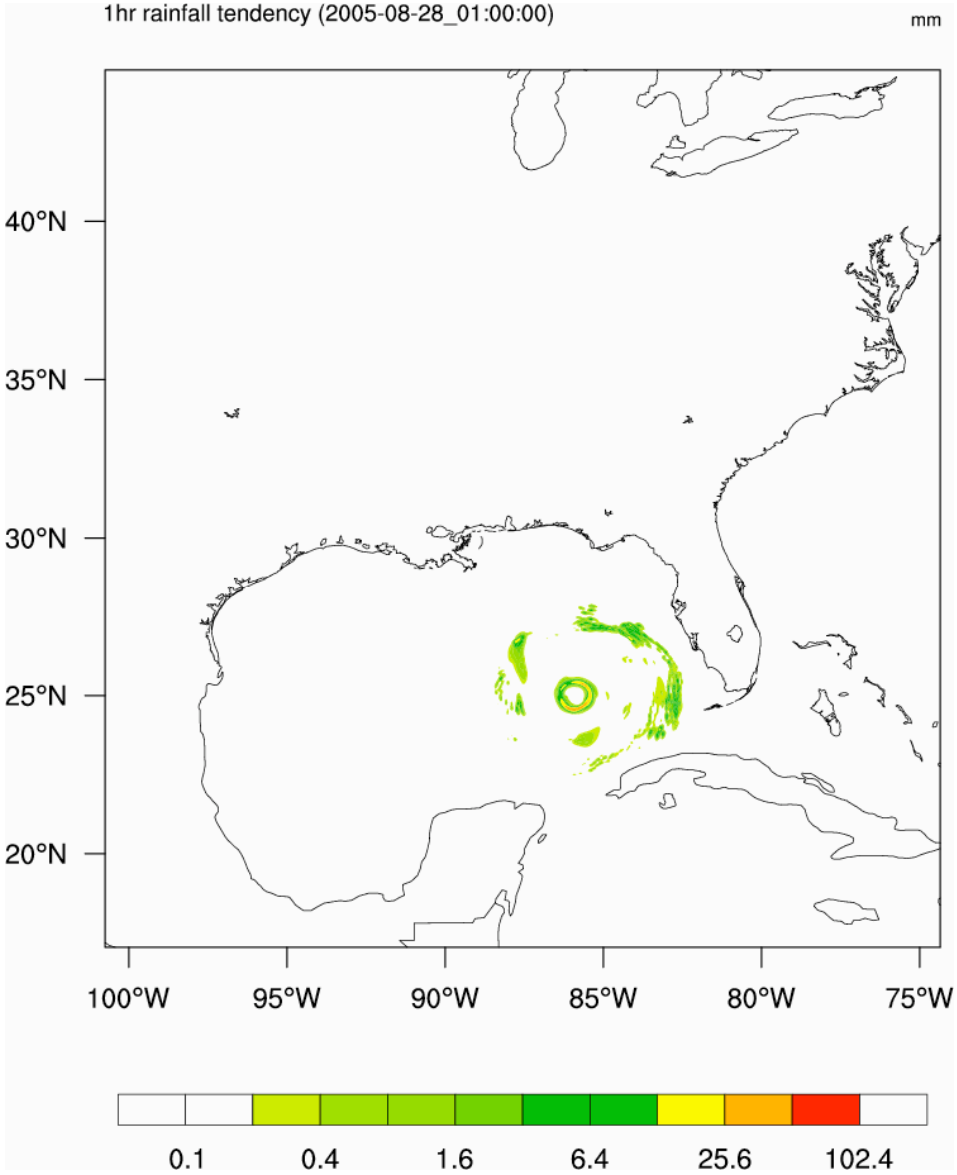




Moving Nests



Moving Nests

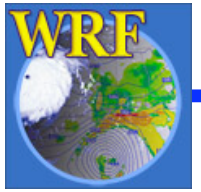


Resources

- **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)



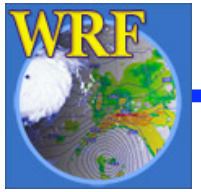
Calling FORTRAN code from NCL

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

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

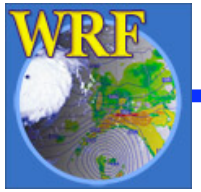
```
C NCLFORTSTART
C NCLEND
```

- **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 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
```

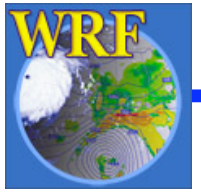


myTK.so - *Create & use in NCL script*

```
% WRAPIT myTK.f
```

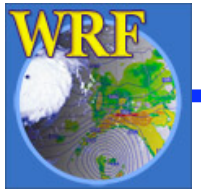
This will create a "myTK.so" file

```
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

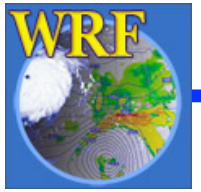
- **Can use simple FORTRAN 90 code**
- **Your FORTRAN 90 program may not contain any of the following features:**
  - pointers or structures as arguments,
  - missing/optional arguments,
  - keyword arguments, or
  - recursive procedure.



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
```



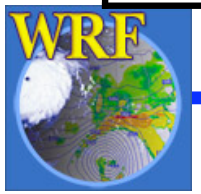
FORTRAN 90 code

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
```