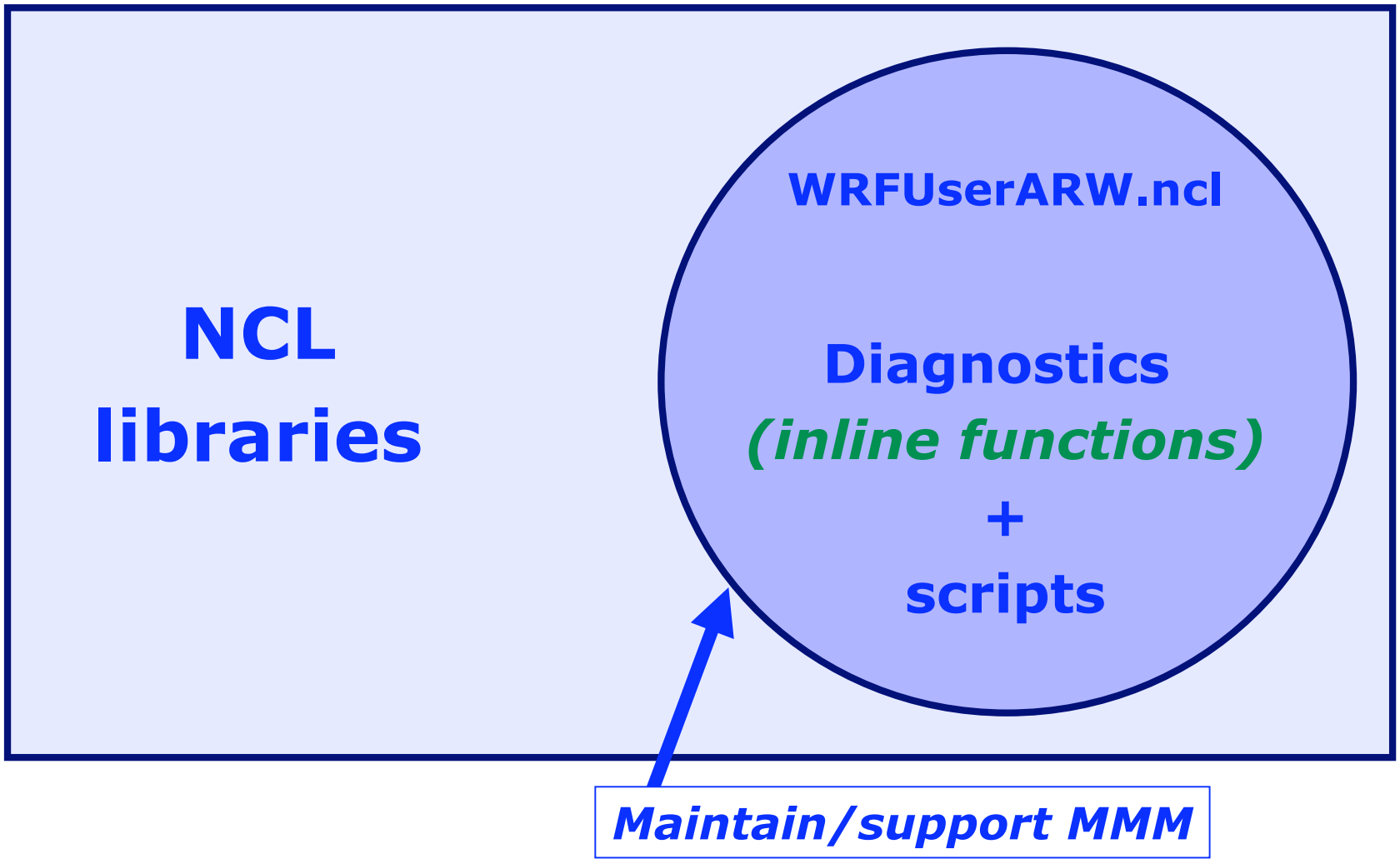
Post-processing Tools: NCL
(WRF-ARW Only)

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*

NCL & WRF


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

Download NCL

- ***http://www.ncl.ucar.edu/Download***
  - Fill out short registration form *(there is a short waiting period)*
  - Read and agree to OSI-based license
  - Download binaries
- **NCARG_ROOT environment variable**
  - *setenv NCARG_ROOT /usr/local/ncl*
- ***NCL version 5.1.0***

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

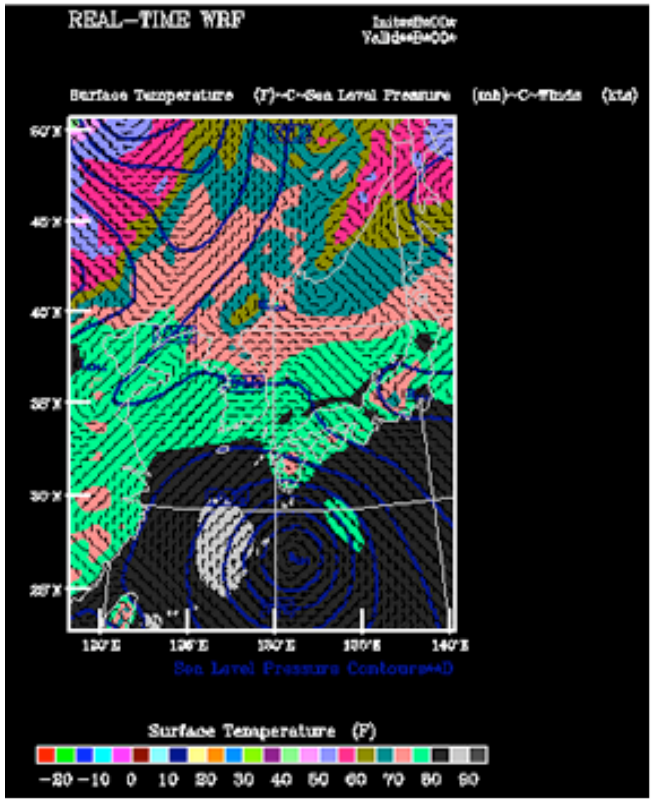
- ***http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml***

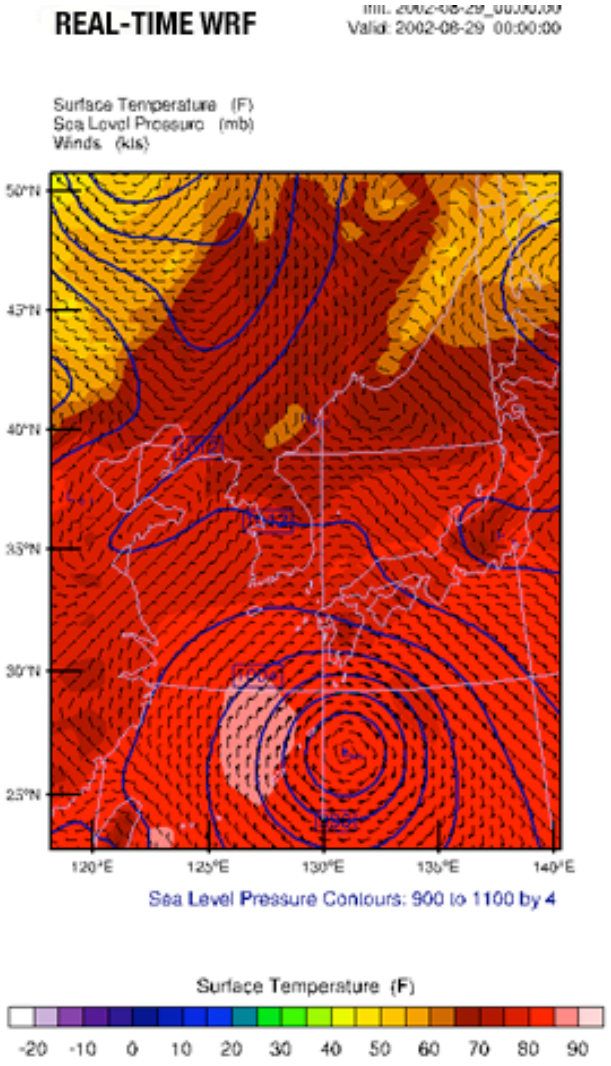
~/.hluresfile

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

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

~/.hluresfile


REAL-TIME WRF
Surface Temperature (F)~C~Sea Level Pressure (mb)~C~Winds (kts)
60°N
45°N
40°N
35°N
30°N
25°N
120°E
125°E
130°E
135°E
140°E
Sea Level Pressure Contours:900 to 1100 by 4
Surface Temperature (F)
-20 -10 0 10 20 30 40 50 60 70 80 90

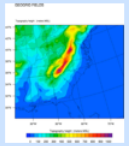
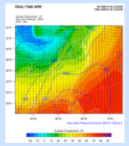
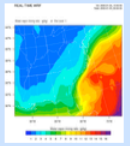
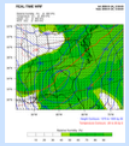
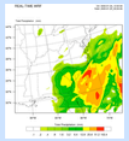
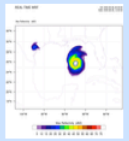
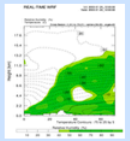
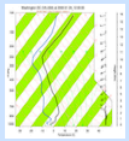
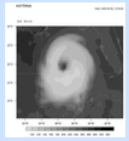
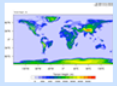
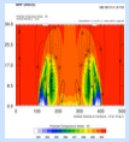

REAL-TIME WRF
Valid: 2002-08-29 00:00:00
Surface Temperature (F)
Sea Level Pressure (mb)
Winds (kts)
50°N
45°N
40°N
35°N
30°N
25°N
120°E
125°E
130°E
135°E
140°E
Sea Level Pressure Contours: 900 to 1100 by 4
Surface Temperature (F)
-20 -10 0 10 20 30 40 50 60 70 80 90

Generate Plots

- **Create a script**
  - *wrf_real.ncl*
- **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

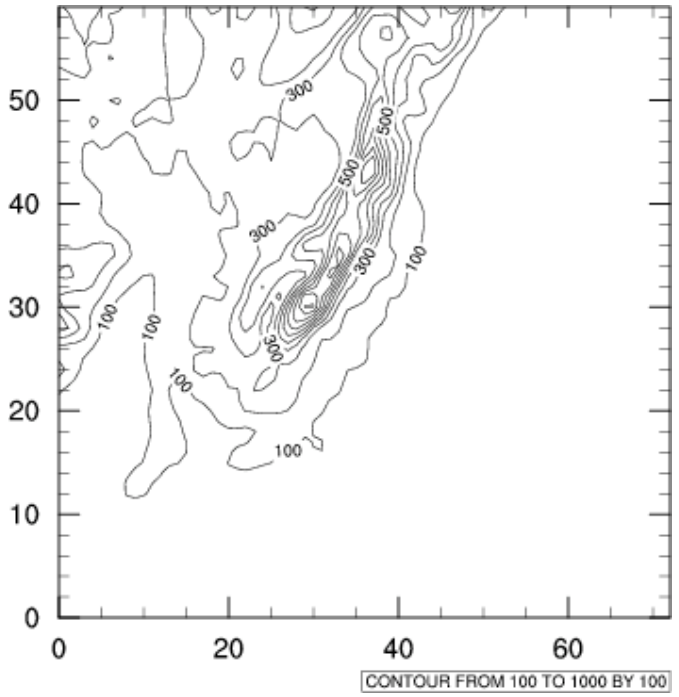
Basic Plots  Basic Plot Setup *(This series of examples takes users though same basic steps in generating plotting scripts.)* Get and plot a single field Multiple input files	**Basic Surface Plots**  Surface 1 Surface 3 Surface 2	**Plots on Model Levels**  Clouds Levels from wrfout files Levels from 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
```

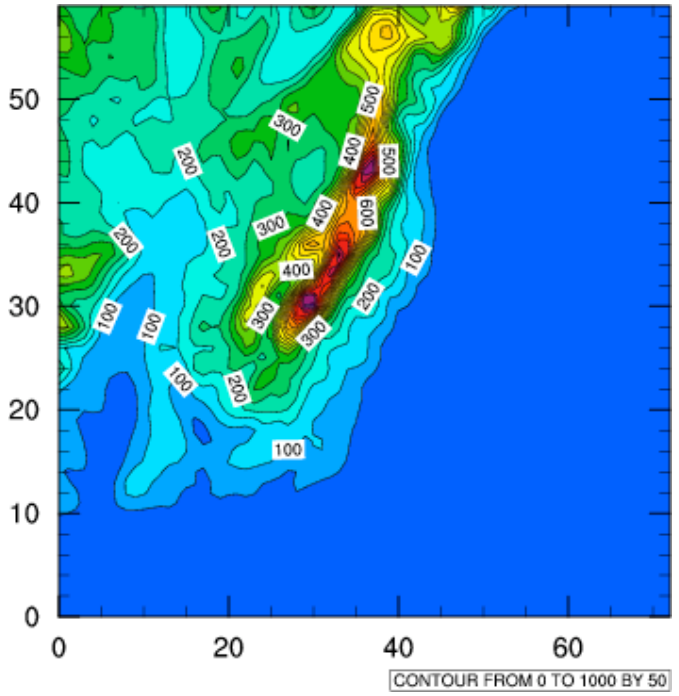
Generate Plots

```
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
```



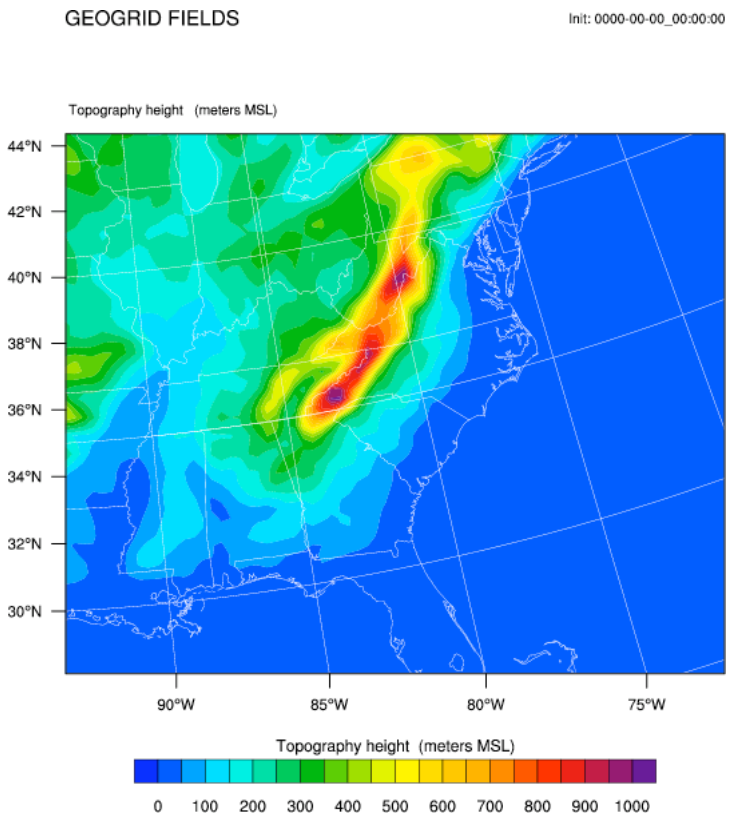
Generate Plots

```
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
```



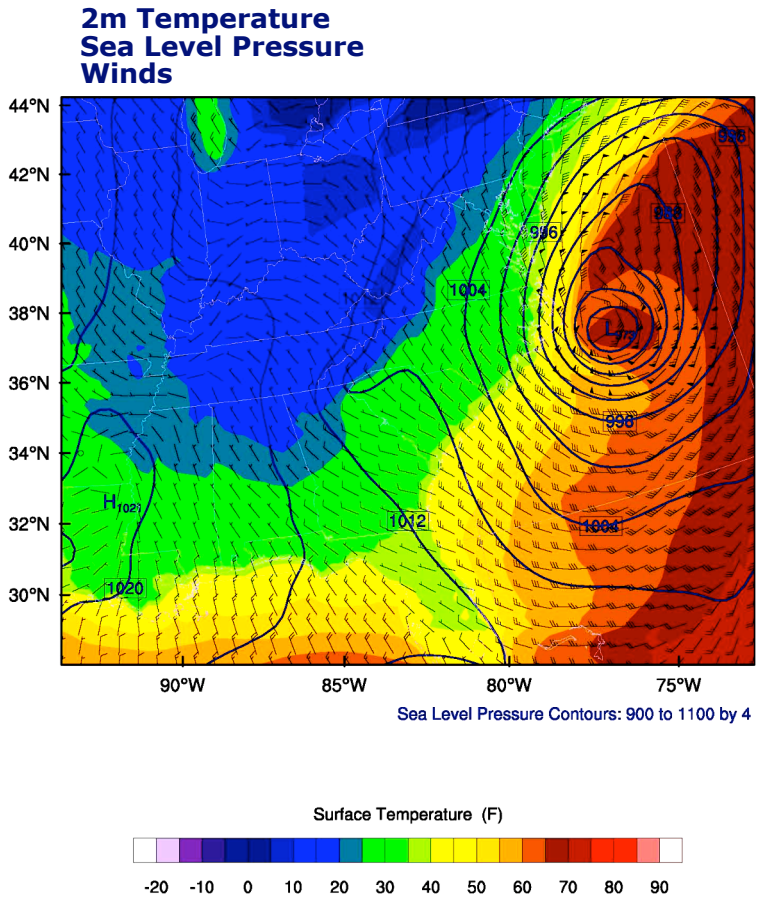
Generate Plots

```
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")
  opts = True
  opts@MainTitle = "GEOGRID FIELDS"

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



Creating a Plot : NCL script

```
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)
```



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

```
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 )
```

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)
```

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)*

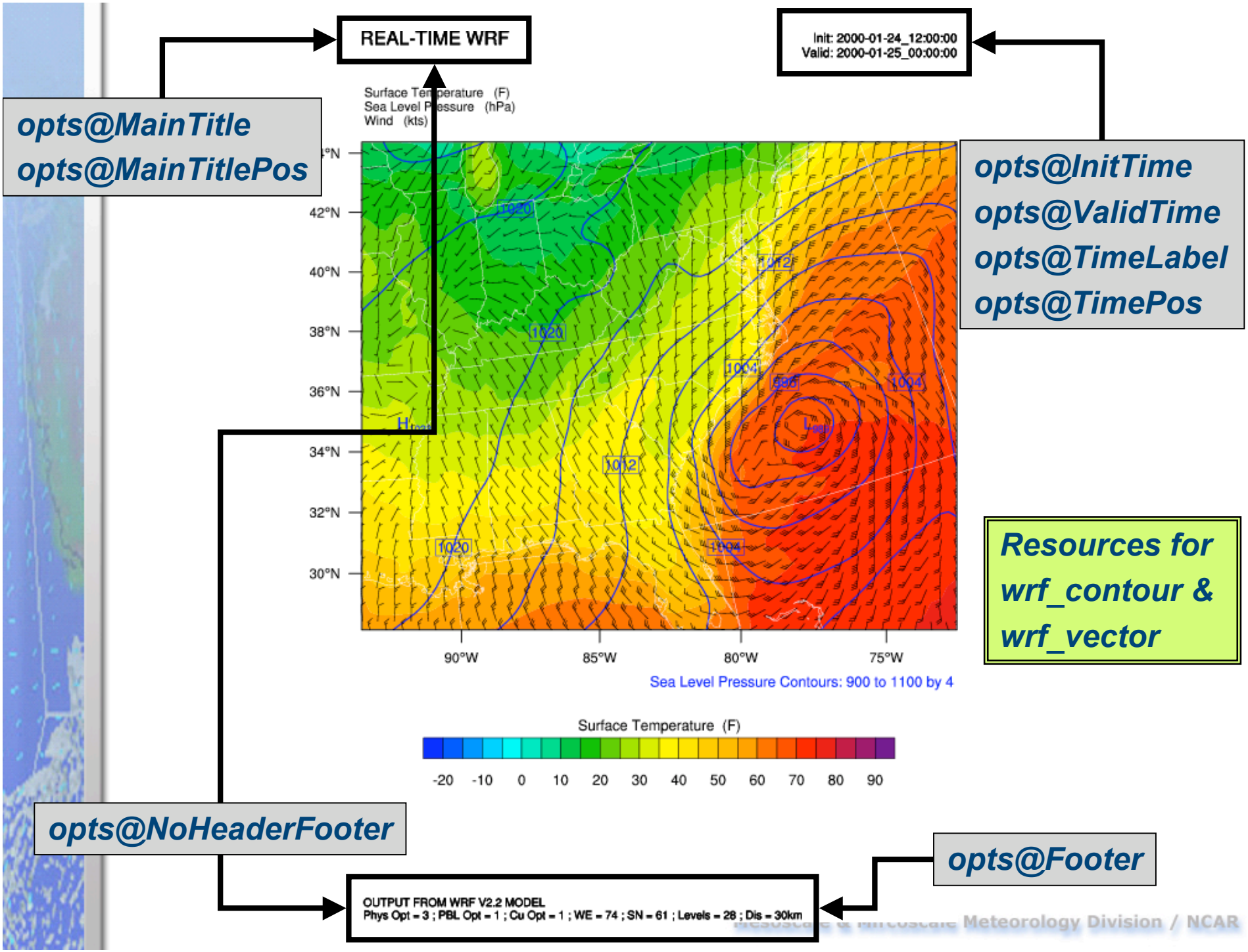
  Diagnostics
  avo/pvo: Absolute/Potential Vorticity, ***cape_2d***: 2D mcape/mcin/lcl/lfc, ***cape_3d***: 3D cape/cin, ***dbz/mdbz***: Reflectivity, ***geopt/geopotential***: Geopotential, ***p/pres/pressure***: Pressure, ***rh/rh2***: Relative Humidity, ***slp***: Sea Level Pressure, ***td/td2***: Dew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

Special WRF Functions

- **wrf_contour / wrf_vector**
  Create line/shaded & vector plots

  contour = wrf_contour(a, wks, ter, opts)

  opts@MainTitle : Main title on the plot.
  opts@MainTitlePos : Main title position
  opts@NoHeaderFooter : Switch 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 : Countour 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 vectors. *(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

Special WRF Functions

wrf_map_overlays / wrf_overlays

Overlay plots created with *wrf_contour* and *wrf_vector*

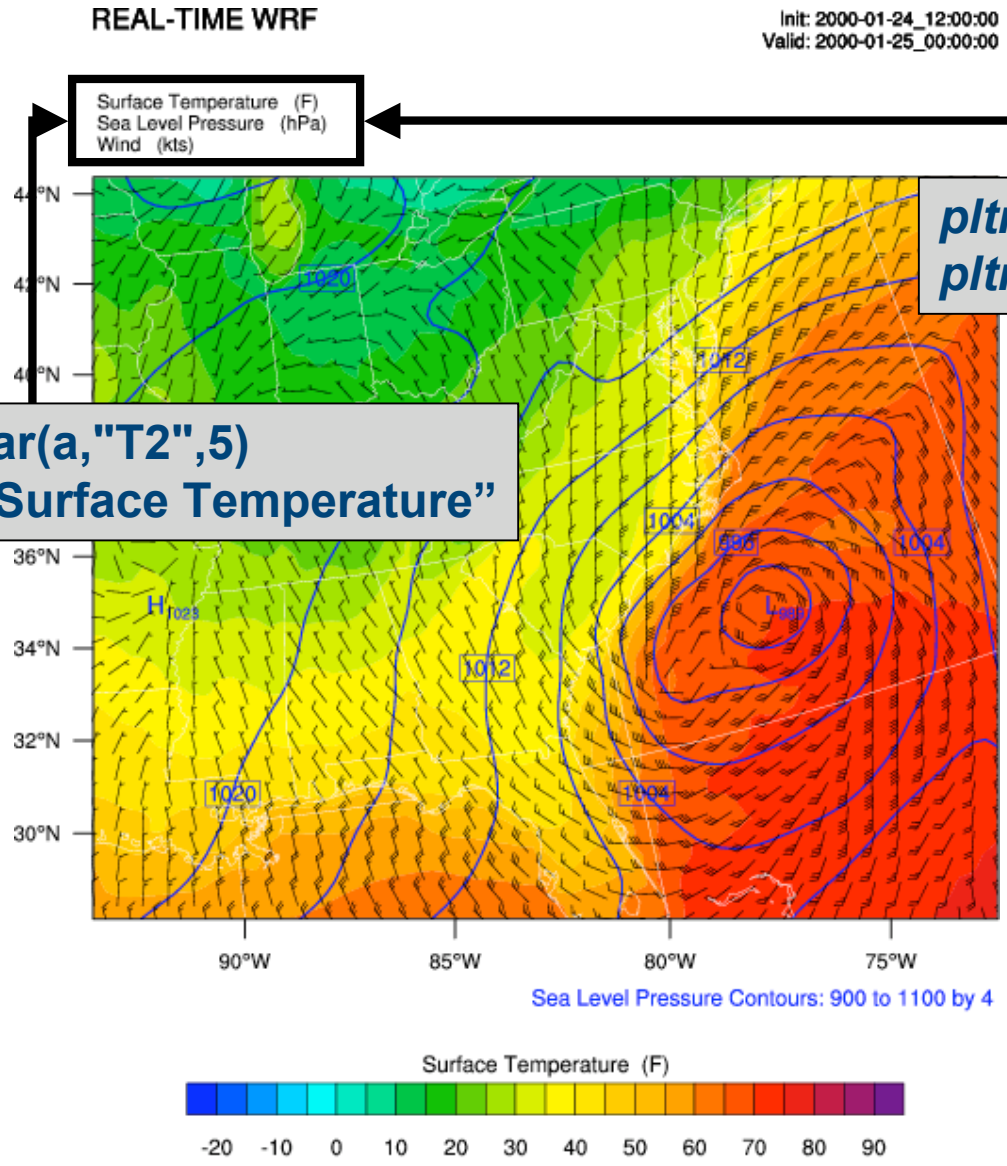
*plot = wrf_map_overlays (a, wks, *
(/contour,vector/), pltres, mpres)

plot = wrf_overlays (a, wks, (/contour,vector/), pltres)

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

To Zoom 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; pltres@CommonTitle; pltres@PlotTitle;
pltres@PanelPot; pltres@FramePlot



pltres@NoTitles
pltres@CommonTitle

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

Resources for wrf_overlay

Special WRF Functions

- **wrf_user_list_times**
  Get list if times available in input file

  times = wrf_user_list_times (a)

- **wrf_map**
  Create a map background - not used often

  map = wrf_map(a, wks, opts)

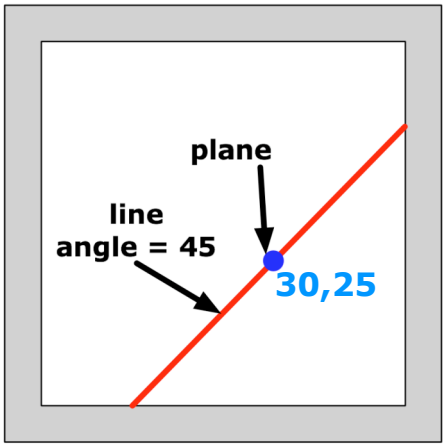
- **wrf_user_unstagger (varin, unstagDim)**
  Unstaggers an array

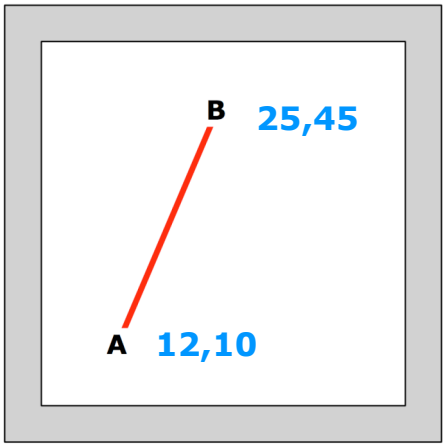
  ua = wrf_user_unstagger (U, "X")
  ua = wrf_user_getvar(a,"ua",time)

Special WRF 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)





Special WRF Functions

- **wrf_user_ll_to_ij / wrf_user_ij_to_ll**
  Convert: lat/lon ⬌ ij

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

  res@useTime - Default is 0
  Set if want the reference longitude/latitudes must come from a specific time - one will 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)*

WRF / Built-in Functions

- **wrf_user_ll_to_ij / wrf_user_ij_to_ll (*WRFUserARW*)**
  wrf_ll_to_ij / wrf_ij_to_ll (*built-in function*)

```
locll = wrf_user_ij_to_ll (a, (/10, 12/), res)
```

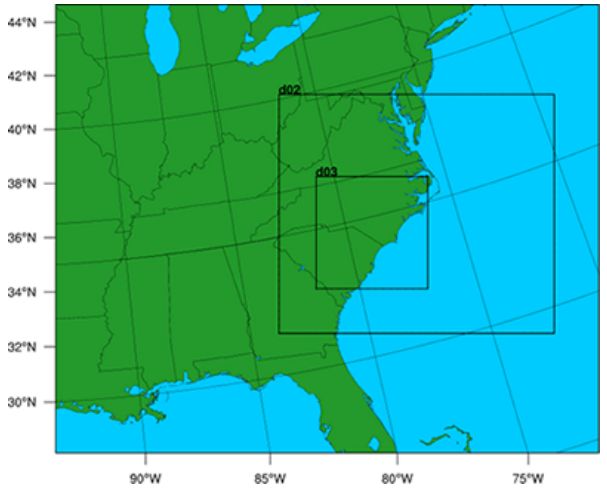
```
res             = True
res@MAP_PROJ    = 1   (lambert)
res@DX          = 30000.
res@DY          = 30000.
res@TRUELAT1    = 30.
res@TRUELAT2    = 60.
res@STAND_LON   = -98.
res@REF_LAT     = 34.83
res@REF_LON     = -81.03

res@KNOWNI      = 37.0
res@KNOWNJ      = 30.5
xx = 1.0
yy = 1.0
loc = wrf_ij_to_ll (xx,yy,res)
```

Center of domain {74 x 64}

Special WRF Functions

- **wrf_wps_dom**
  Experimental - preview domian location
  mp = wrf_wps_dom (wks, mpres, lnres, txres)

```
mpres@max_dom             = 3
mpres@parent_id           = (/ 1,    1,   2 /)
mpres@parent_grid_ratio   = (/ 1,    3,   3 /)
mpres@i_parent_start      = (/ 1,   31,  15 /)
mpres@j_parent_start      = (/ 1,   17,  20 /)
mpres@e_we                = (/ 74, 112, 133/)
mpres@e_sn                = (/ 61,  97, 133 /)
mpres@dx                  = 30000.
mpres@dy                  = 30000.
mpres@map_proj            = "lambert"
mpres@ref_lat             = 34.83
mpres@ref_lon             = -81.03
mpres@truelat1            = 30.0
mpres@truelat2            = 60.0
mpres@stand_lon           = -98.0
```



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

Adding FORTRAN code

- **Can link NCL to existing FORTRAN code**
- **Easiest to deal with F77 code**
- **Code must contain special *NCL START* and *END* markers**
- **Must create a shared object library from these routines in order for NCL to recognize the code.**

myTK.f

```
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)


  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.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

WRAPIT myTK.f

setenv NCARG_ROOT */usr/local/ncl*

/usr/local/ncl/bin/WRAPIT myTK.f

cp /usr/local/ncl/bin/WRAPIT .

edit your copy

./WRAPIT myTK.f

myTK.so - use in NCL script

```
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)

      dim = dimsizes(t)
      tk = new( dimsizes(t), typeof(t) )

      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, 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, 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

WRAPIT myTK90.stub myTK.f90

setenv NCARG_ROOT */usr/local/ncl*
/usr/local/ncl/bin/WRAPIT myTK.f
cp /usr/local/ncl/bin/WRAPIT . *edit your copy - link to a f90 compiler* ./WRAPIT myTK.f

myTK90.so - use in NCL script

```
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)

    dim = dimsizes(t)
    tk = new( dimsizes(t), typeof(t) )

    myTK90 :: compute_tk(tk,p,t,dim(2),dim(1),dim(0))

end
```