
Using NCL with VAPOR to Visualize WRF-ARW data

January 2012

Web links updated April 2013

Version 2.1

Introduction

This document is a tutorial on the combined use of two NCAR-developed visualization tools, VAPOR (Visualization and Analysis Platform for atmospheric, Oceanic, and solar Research) and NCL (NCAR Command Language). NCL is used to construct 2D data plots from WRF data, and then these 2D plots can be inserted in an appropriate location in a VAPOR 3D scene of the WRF data.

A basic familiarity with NCL and VAPOR may be helpful to the reader, but not required for this tutorial.

This tutorial makes use of a WRF-ARW simulation of typhoon Jangmi, a typhoon with sustained winds of 160mph, passing over the Philippines, Taiwan, and China, during September and October 2008. To limit the size of the data to download for this exercise, we focus on the first 10 hours of the day of September 28, when the typhoon passed over Taiwan.

Several related VAPOR documents may be useful with this tutorial:

- The [VAPOR/WRF Data and Image Preparation Guide](#). Sections 1-3 of this guide provide background information on the procedures of this tutorial.
- [VAPOR User's Guide for WRF Typhoon Research](#) shows how to use VAPOR to visualize the same dataset (typhoon Jangmi) that is used in this tutorial.
- The document [Visualization of WRF Data using VAPOR: A Georgia Weather Case Study](#) is a tutorial that shows how to use VAPOR to visualize WRF data, including images such as those generated here.

The reader should follow the following 5 steps of image preparation and usage. Each step requires completion of the preceding steps.

Using VAPOR with NCL on WRF data: A short tutorial

1. [Obtaining the needed data and software for this tutorial.](#)
2. [Converting the WRF output data to a VAPOR dataset.](#)
3. [Creating horizontal and vertical plots of the WRF output data using NCL.](#) These images are initially produced as postscript files; then, using the NCL script “wrf2geotiff.ncl”, the output images are converted to a geo-referenced tiff (geo-tiff) file.
4. [Obtaining a geo-referenced terrain image for a region containing the WRF simulation output.](#) This uses the VAPOR shell script “getWMSImage.sh”.
5. [Inserting the NCL plots and the terrain image into a 3D scene, and visualize the images together with the WRF 3D weather data in VAPOR.](#)

At the end of this document is provided, for reference, an additional section, namely the [Appendix](#), containing the NCL scripts used in section 3.

1 Prepare software and data

To work through these examples you will need the following:

1. VAPOR 1.5 or later

Vapor installers can be obtained from the VAPOR download site, <http://www.vapor.ucar.edu/page/vapor-download#Binary>. In most cases the binary installers work well; it is not necessary to compile VAPOR. Installation instructions are at <http://www.vapor.ucar.edu/docs/vapor-installation/vapor-installation>. It is best to install VAPOR on a computer that has a good graphics card; in particular, most nVidia and ATI cards work well with VAPOR.

2. NCL version 5.1 or later: If NCL is not already installed on your system, follow the download and installation instructions provided at: <http://www.ncl.ucar.edu/Download/>

NOTE: You must have the file “.hluresfile” installed in your home directory, for these examples to work properly. You can download an example of this file from <http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml>.

3. WRF output data files of typhoon Jangmi on September 28, 2008. This data was provided by Dr. Bill Kuo and Dr. Wei Wang of NCAR, and Dr. Minsu Joh of KISTI.

These WRF output files are for 10 hours of the day that typhoon Jangmi passed over Taiwan. The domain is a moving D02 domain nest that tracks the typhoon. These WRF output files are provided as either a zip file or as a gzip/tar file. The WRF output files

have been renamed without colons (:) in the name (the trailing :00:00 has been removed from the name) because Windows does not allow colons in file names. Create a new directory to hold these WRF output files. Download and uncompress one of the following files. (Note that you will need about 850MB of storage to hold the unzipped data files; the files to download are about 450MB. Shorter versions (containing only two time steps, around 80MB) are also available for users lacking sufficient storage space.):

- Ten time-step files: [jangmiWrfout.tar.gz](#) and [jangmiWrfout.zip](#)
 - Shorter two time-step files: [jangmiWrfout_small.tar.gz](#) and [jangmiWrfout_small.zip](#)
4. Two additional (UNIX) utilities are needed; check to be sure they are available on your system. If you are using Windows, these must be available in the [Cygwin-X](#) environment:
 - convert (from [ImageMagick](#) , version 6.2 or later is best)
 - psplit (this comes with NCL)
 5. The three NCL scripts we use are included in this document, and versions of these are also available at the [NCL/WRF](#) website. Download each of the following scripts to the same directory where you have put the WRF output files:
 - [wrf_Height.ncl](#) (this plots humidity, temperature, pressure and wind at a fixed elevation)
 - [wrf_Precip.ncl](#) (this plots precipitation tendency with pressure isobars)
 - [wrf_crossSection2.ncl](#) (this is a vertical plot of relative humidity)

Note that numerous other NCL scripts that generate plots of WRF data are available at the NCL/WRF website; we have chosen the above three to illustrate various options for using NCL with VAPOR. The other NCL scripts on the WRF/NCL examples page can similarly be converted to produce plots that display in VAPOR. In the VAPOR examples/NCL directory, five such converted example NCL scripts are provided.

2 (Optional) Convert the WRF output data to a VAPOR dataset

Starting with VAPOR 2.0, it is no longer necessary to convert WRF output files to VAPOR data format: the VAPOR user interface (vaporgui) can read WRF output files directly. If your dataset is very large then it is still useful to perform the data conversion, because a Vapor Data Collection (VDC) will perform better on such large data; however, for the purposes of this tutorial you may skip the conversion step, and, rather than load the VAPOR data, you can import the WRF output files directly.

We do provide the data conversion here, as an optional exercise as follows: This conversion is performed in two steps. The conversion requires that you have already installed VAPOR. Before the commands are run on UNIX platforms, you should ensure VAPOR has been set up in your current shell, by sourcing the command `vapor-setup.csh` (for c-shell) or `vapor-setup.sh` (for the Bourne shell)

1. **Run wrfvdfcreate.** This application scans the WRF output files, and constructs a VAPOR metadata file that describes the dataset. It also provides some useful information about the coordinate extents of the WRF domain. For this exercise, the default options suffice. Other options are described in the wrfvdfcreate [man pages](#).
CD to the directory in which you have downloaded and unzipped the WRF output files. Then issue the following command:

```
wrfvdfcreate wrfout_d02_2008-09-28* jangmi-09-28.vdf
```

On completion of this command you will see that the file “jangmi-09-28.vdf” has been created. The console output of this command will look something like the following:

```
$ wrfvdfcreate wrfout_d02_2008-09-28* jangmi-09-28.vdf
Created UDF file:
  Num time steps : 10
  3D Variable names : U U W PH PHB T P PB QUAPOR QCLOUD QRAIN QICE QSNOW E
LEUATION
  2D Variable names : LU_INDEX MU MUB NEST_POS SR POTEUP SNOPCX SOILTB Q2
T2 TH2 PSFC U10 U10 LANDMASK SEAICE XICEM SFROFF UDROFF IUGTYP ISLTYP VEGFRA GRD
FLX SNOW SNOWH RHOSN CANWAT SST MAPFAC_M MAPFAC_U MAPFAC_U MAPFAC_MX MAPFAC_MY M
APFAC_UX MAPFAC_UY MAPFAC_UX MF_UX_INU MAPFAC_UY F E SINALPHA COSALPHA HGT HGT_S
HAD TSK RAINC RAINNC PRATEC RAINCU SNOWNC GRAUPELNC EDT_OUT SWDOWN GLW OLR XLAT
XLONG XLAT_U XLONG_U XLAT_U XLONG_U ALBEDO ALBBCK EMISS TMN XLAND UST PBLH HFX Q
FX LH SNOWC
  Coordinate extents : 0 0 30.7281 800000 800000 25400.8
  Min Longitude and Latitude of domain corners: 117.053 18.7224
  Max Longitude and Latitude of domain corners: 126.302 27.4392
```

Console output resulting from running wrfvdfcreate on typhoon data

Note the latitude and longitude extents of the data; these will be useful later when obtaining a terrain image for this domain.

2. **Run wrf2vdf:** This application reads all the 2D and 3D variables of the WRF output files, and converts them the format that VAPOR uses, namely a VDC (Vapor Data Collection).

Issue the command:

```
wrf2vdf jangmi-09-28.vdf wrfout_d02_2008-09-28*
```

The command output will list all the variables and time steps that are converted.

This command will take a minute or so to complete. On completion, the VDC data is in a subdirectory jangmi-09-28_data of the current directory. The jangmi-09-28_data directory will contain a subdirectory for each variable, representing each WRF variable in a multi-resolution (wavelet) form.

The first few lines of the console output of this command are shown below.

```
$ wrf2vdf jangmi-09-28.vdf wrfout_d02_2008-09-28*
Processing file : wrfout_d02_2008-09-28_00
    Transforming variables : U U W PH PHB T P PB QUAPOR QCLOUD QRAIN QICE QS
NOW ELEVATION LU_INDEX MU MUB NEST_POS SR POTEUP SNOPCX SOILTB Q2 T2 TH2 PSFC U1
0 U10 LANDMASK SEAICE XICEM SFROFF UDROFF IUGTYP ISLTYP UEGFRA GRDFLX SNOW SNOWH
    RHOSN CANWAT SST MAPFAC_M MAPFAC_U MAPFAC_U MAPFAC_MX MAPFAC_MY MAPFAC_UX MAPFA
C_UY MAPFAC_UX MF_UX_INU MAPFAC_UY F E SINALPHA COSALPHA HGT HGT_SHAD TSK RAINC
RAINNC PRATEC RAINCU SNOWNC GRAUPELNC EDT_OUT SWDOWN GLW OLR XLAT XLONG XLAT_U X
LONG_U XLAT_U XLONG_U ALBEDO ALBBCK EMISS TMN XLAND UST PBLH HFX QFX LH SNOWC
    Time step : 0 (UDC), 0 (WRF)
Processing file : wrfout_d02_2008-09-28_01
    Transforming variables : U U W PH PHB T P PB QUAPOR QCLOUD QRAIN QICE QS
NOW ELEVATION LU_INDEX MU MUB NEST_POS SR POTEUP SNOPCX SOILTB Q2 T2 TH2 PSFC U1
0 U10 LANDMASK SEAICE XICEM SFROFF UDROFF IUGTYP ISLTYP UEGFRA GRDFLX SNOW SNOWH
    RHOSN CANWAT SST MAPFAC_M MAPFAC_U MAPFAC_U MAPFAC_MX MAPFAC_MY MAPFAC_UX MAPFA
C_UY MAPFAC_UX MF_UX_INU MAPFAC_UY F E SINALPHA COSALPHA HGT HGT_SHAD TSK RAINC
RAINNC PRATEC RAINCU SNOWNC GRAUPELNC EDT_OUT SWDOWN GLW OLR XLAT XLONG XLAT_U X
LONG_U XLAT_U XLONG_U ALBEDO ALBBCK EMISS TMN XLAND UST PBLH HFX QFX LH SNOWC
    Time step : 1 (UDC), 0 (WRF)
Processing file : wrfout_d02_2008-09-28_02
    Transforming variables : U U W PH PHB T P PB QUAPOR QCLOUD QRAIN QICE QS
NOW ELEVATION LU_INDEX MU MUB NEST_POS SR POTEUP SNOPCX SOILTB Q2 T2 TH2 PSFC U1
0 U10 LANDMASK SEAICE XICEM SFROFF UDROFF IUGTYP ISLTYP UEGFRA GRDFLX SNOW SNOWH
    RHOSN CANWAT SST MAPFAC_M MAPFAC_U MAPFAC_U MAPFAC_MX MAPFAC_MY MAPFAC_UX MAPFA
C_UY MAPFAC_UX MF_UX_INU MAPFAC_UY F E SINALPHA COSALPHA HGT HGT_SHAD TSK RAINC
RAINNC PRATEC RAINCU SNOWNC GRAUPELNC EDT_OUT SWDOWN GLW OLR XLAT XLONG XLAT_U X
LONG_U XLAT_U XLONG_U ALBEDO ALBBCK EMISS TMN XLAND UST PBLH HFX QFX LH SNOWC
    Time step : 2 (UDC), 0 (WRF)
Processing file : wrfout_d02_2008-09-28_03
    Transforming variables : U U W PH PHB T P PB QUAPOR QCLOUD QRAIN QICE QS
NOW ELEVATION LU_INDEX MU MUB NEST_POS SR POTEUP SNOPCX SOILTB Q2 T2 TH2 PSFC U1
0 U10 LANDMASK SEAICE XICEM SFROFF UDROFF IUGTYP ISLTYP UEGFRA GRDFLX SNOW SNOWH
    RHOSN CANWAT SST MAPFAC_M MAPFAC_U MAPFAC_U MAPFAC_MX MAPFAC_MY MAPFAC_UX MAPFA
C_UY MAPFAC_UX MF_UX_INU MAPFAC_UY F E SINALPHA COSALPHA HGT HGT_SHAD TSK RAINC
RAINNC PRATEC RAINCU SNOWNC GRAUPELNC EDT_OUT SWDOWN GLW OLR XLAT XLONG XLAT_U X
LONG_U XLAT_U XLONG_U ALBEDO ALBBCK EMISS TMN XLAND UST PBLH HFX QFX LH SNOWC
    Time step : 3 (UDC), 0 (WRF)
Processing file : wrfout_d02_2008-09-28_04
    Transforming variables : U U W PH PHB T P PB QUAPOR QCLOUD QRAIN QICE QS
NOW ELEVATION LU_INDEX MU MUB NEST_POS SR POTEUP SNOPCX SOILTB Q2 T2 TH2 PSFC U1
0 U10 LANDMASK SEAICE XICEM SFROFF UDROFF IUGTYP ISLTYP UEGFRA GRDFLX SNOW SNOWH
```

The first few lines of the console output resulting from running wrf2vdf

3 Make NCL scripts to plot data as georeferenced tiff images

We start with three different NCL scripts as provided on the WRF/NCL Web page, and illustrate the changes that are necessary to convert these scripts, with sufficient detail that the user should be able to apply the same steps to other NCL scripts to obtain geo-referenced output images. The three scripts are: [wrf_Height.ncl](#), [wrf_Precip.ncl](#), and [wrf_CrossSection2.ncl](#). These three scripts were chosen because the resulting images are

appropriate for three different kinds of display in VAPOR:

- The `wrf_Height.ncl` script produces an image that presents the data on a particular horizontal plane. The resulting images can appropriately be positioned in the VAPOR scene at the height of that plane.
- The `wrf_Precip.ncl` script produces a plot that presents precipitation tendencies and sea-level pressure. This plot can appropriately be positioned on any horizontal plane in the VAPOR scene, or can be applied to the terrain surface.
- The `wrf_CrossSection2.ncl` script produces a vertical plot of temperature and relative humidity along a particular plane, parallel to the X-Z axis. The resulting images should be positioned to coincide with the specified vertical plane in the VAPOR scene.

Each of these three scripts reads in a WRF dataset and produces a number of NCL plots to a console window (as X11 output). We shall convert these to produce Postscript (.ps) output files, and the Postscript output will then be converted to georeferenced TIFF files. In the [Appendix](#) we provide three versions of each of these scripts: The initial version (obtained from the WRF/NCL page), a first modification (customized to produce a series of plots, exactly one per timestep, from the sample WRF data set), and a final version that produces a georeferenced TIFF file.

The conversion process involves the following steps:

1. Modify the script so that it generates only one image at each time step. (If you want to use multiple images at the same time steps then you need to run a separate script for each of the different images)
2. Modify this script so that it will loop over all time steps in the WRF files. This may involve producing an outer loop that loops over several files, and an inner loop that handles the time steps in each file.
3. Modify the script so that the output is a postscript (.ps) file
4. Insert the following new lines into your NCL script:
 - Near the top, insert a line to load `wrf2geotiff.ncl` (This script and other NCL examples are by default installed in the subdirectory `share/vapor-x.x.x/examples/NCL`, of the `$(VAPOR_HOME)` directory where vapor is installed, where “x.x.x” should be replaced by “1.5.2” or whatever version of VAPOR is being used.

(N.B. for Windows users: On Windows, the file `wrf2geotiff.ncl` is installed in `$(VAPOR_HOME)/share/examples/NCL/`. However, NCL does not allow blanks in the file path for the NCL load instruction. To ensure there are no blanks in the path, copy the `wrf2geotiff.ncl` file to your current directory, and then change the load instruction to:

```
load "wrf2geotiff.ncl"
```

- After the NCL workstation “wks” is created, call the function `wrf2geotiff_create(wks)`. In these examples we name the returned opaque object “wrf2gtiff”.
- If this is a vertical plot, disable georeferencing by calling:

- `wrf2geotiff_disableGeoTags (wrf2gtiff)`
 - Set the plot resource `pltres@gsnFrame=False`. This enables manual control of the frame advance.
 - Each time a plot is created, insert two lines:
 - Call `wrf2geotiff_write()`, to cause the plot to be written to file. The final argument to `wrf2geotiff_write()` determines whether or not the image will be cropped.
 - Call `frame(wks)` to advance the frame.
 - At the end of the script, insert a call to `wrf2geotiff_close()`, which will cause all the images to be combined into one geo-tiff file with appropriate geo-referencing.
- 5. When you run this new script, make sure you execute it from a shell where you have already sourced `vapor_setup.csh` or `vapor_setup.sh`, to set up the correct execution environment. The result will be one georeferenced tiff file with all the images (one for each time step), ready to be loaded into a VAPOR 3D scene.

3.1 Conversion of `wrf_Height.ncl`

We follow the above steps in some detail using the first example script, [wrf_Height.ncl](#). For the other two scripts, we just illustrate the changes that are made for this conversion.

First, make sure that your environment is set up to run the `wrf_Height.ncl` script. To see that the `wrf_Height.ncl` script operates correctly, find the “addfile” command at line 14 in the script, and edit it by providing the path to the first WRF file, as follows:

```
a = addfile("wrfout_d02_2008-09-28_00.nc", "r")
```

Note that we assume that the script is in the same directory as the `wrfout` file. If that is not the case, you need to include the path with the name of the file in the above line. Issue the command:

```
ncl wrf_Height.ncl
```

You should see an image of humidity, temperature, pressure, and wind, at elevation 0.25 km, followed by an image at 2.0 km.

Next, we modify this script so that it produces one image at each time step, and at a higher elevation: i.e. we make an image of humidity, temperature, pressure and wind at 5.0km. Edit the loop over levels, by replacing

```
do level = 0, nlevels-1 ; LOOP OVER LEVELS
with:
level = 1
```

change the line (above the TIME LOOP):

```
height_levels = (/250., 2000./)
```

to

```
height_levels = (/250., 5000./)
```

Remove the line:

```
end do ; END OF LEVEL LOOP
```

Again run the script. You should see only one image, a plot at height 5km.

To make a plot at all ten times, we must modify the script to loop over the various WRF output files. Replace the line that we previously edited:

```
a = addfile("wrfout_d02_2008-09-28_00.nc","r")
```

with the following lines:

```
wrffiles = systemfunc("ls wrfout_d02_2008-09-28*")
numFiles = dimsizes(wrffiles)
do i = 0, numFiles-1
  wrffiles(i) = wrffiles(i) + ".nc"
end do
inpFiles = addfiles(wrffiles,"r")
```

These lines will create an array of filenames ("inpFiles"), one for each wrf output file. To loop over these filenames, insert the following two lines:

```
do ifile = 0, numFiles-1
  a = inpFiles[ifile]
```

Immediately above the line

```
times = wrf_user_list_times(a) ;get times in the file
```

Also, to terminate this loop over files, insert the line:

```
end do ;END OF FILE LOOP
```

right after the line:

```
end do ;END OF TIME LOOP
```

Note that the original script was producing images every other time step. To get an image for each time step, change the line:

```
do it = 0, ntimes-1, 2 ; TIME LOOP
```


To:

```
do it = 0, ntimes-1      ; TIME LOOP
```

(This won't affect our current plots, because there is only one time step per file in the Jangmi dataset; however this change would be needed for datasets with multiple times in a file).

Again execute the command:

```
ncl wrf_Height.ncl
```

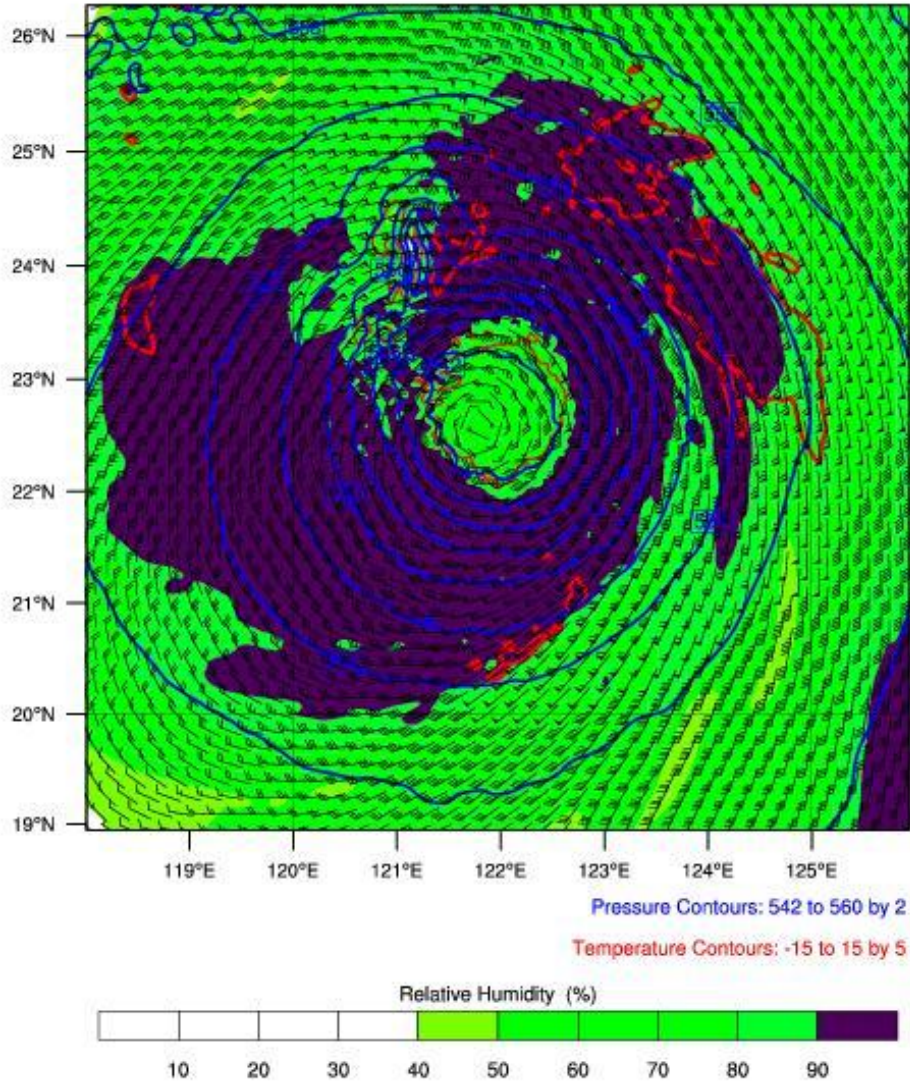
and you will see ten images, one for each of the ten time steps in the WRF data.

This file is saved as “[wrf_Height_FirstMod.ncl](#)” in the [Appendix](#) for your reference. The third of the resulting images is the following:

REAL-TIME WRF

Init: 2006-09-26_12:00:00
Valid: 2006-09-28_02:00:00

Relative Humidity (%) at 5 km
Temperature (C) at 5 km
Pressure (hPa) at 5 km
Wind (kts) at 5 km



Currently the wrf_Height.ncl script produces X11 graphics to your screen. We next modify it so that it produces a geo-referenced tiff file as output. This requires the following changes

to the script:

At or near the top, insert as one line:

```
load "$VAPOR_HOME/share/examples/NCL/wrf2geotiff.ncl"
```

(Windows users should change the load instruction to:

```
load "wrf2geotiff.ncl"
```

and copy the file `wrf2geotiff.ncl` from `"$VAPOR_HOME/share/examples/NCL/` into the current directory, as discussed above.)

The above line loads the library of NCL commands that will be used later in the file. `wrf2geotiff.ncl` is an auxiliary library that is shipped in the NCL examples directory of the VAPOR distribution.

Next, comment out the following line (precede with a semicolon ";"):

```
Type = "x11"
```

And un-comment (remove the preceding ";"):

```
; type = "ps"
```

This makes the output a postscript file instead of x11. Postscript files are needed for the conversion to tiff.

After the line

```
wks = gsn_open_wks(type,"plt_HeightLevel")
```

Insert the line:

```
wrf2gtiff = wrf2geotiff_open(wks)
```

This creates an opaque pointer that is used to track the geotiff capture process

Note: For vertical plots (such as the [wrf_CrossSection2_Final.ncl](#) example) it is necessary to insert a statement here to disable the georeferencing capability. The `wrf_Height.ncl` plot is a horizontal plot, not a vertical plot; but, if it were vertical, at this point we would insert the statement `"wrf2geotiff_disableGeoTags(wrf2gtiff)"`.

After the line:

```
pltres = True
```

Insert the line:

```
pltres@gsnFrame = False
```

This `gsnFrame` resource needs to be `False` for us to control the timing when a new page is output.

The plot is created by the call to `wrf_map_overlays`

```
plot = wrf_map_overlays(a,wks,...,pltres,mpres)
```

after that line, insert the two lines:

```
wrf2geotiff_write(wrf2gtiff, a, times(it),wks,plot,False)
frame(wks)
```

The above two lines cause information about the image to be saved (time, georeferencing), so that it will later be output. The final argument to `wrf2geotiff_write()` determines whether or not the output image will be cropped to the WRF domain extents. By setting it to “False” we will obtain an image that extends beyond the plot area, with annotation, colorbar, legends, etc. displayed outside the plot area. If you would like to not display the annotation, set this final argument to `True`. The annotation can also be cropped when it is displayed in VAPOR. The “`frame(wks)`” command is required (along with the “`pltres@gsnFrame=False`” resource definition inserted previously) in order to cause a new frame to be started after the georeferencing is saved.

Finally, immediately before the end at the end of the script, insert the line:

```
wrf2geotiff_close(wrf2gtiff,wks)
```

This last statement triggers the necessary file conversions needed to convert the postscript images to an output geo-tiff file.

Run the script. You will note several messages to the console. When the script completes, you should find a file “`plt_HeightLevel.tiff`” in the current directory. This is a geo-tiff file that includes the ten plots at height 2km together with geo-referencing information.

This script is saved as [wrf_Height_Final.ncl](#) in the [Appendix](#).

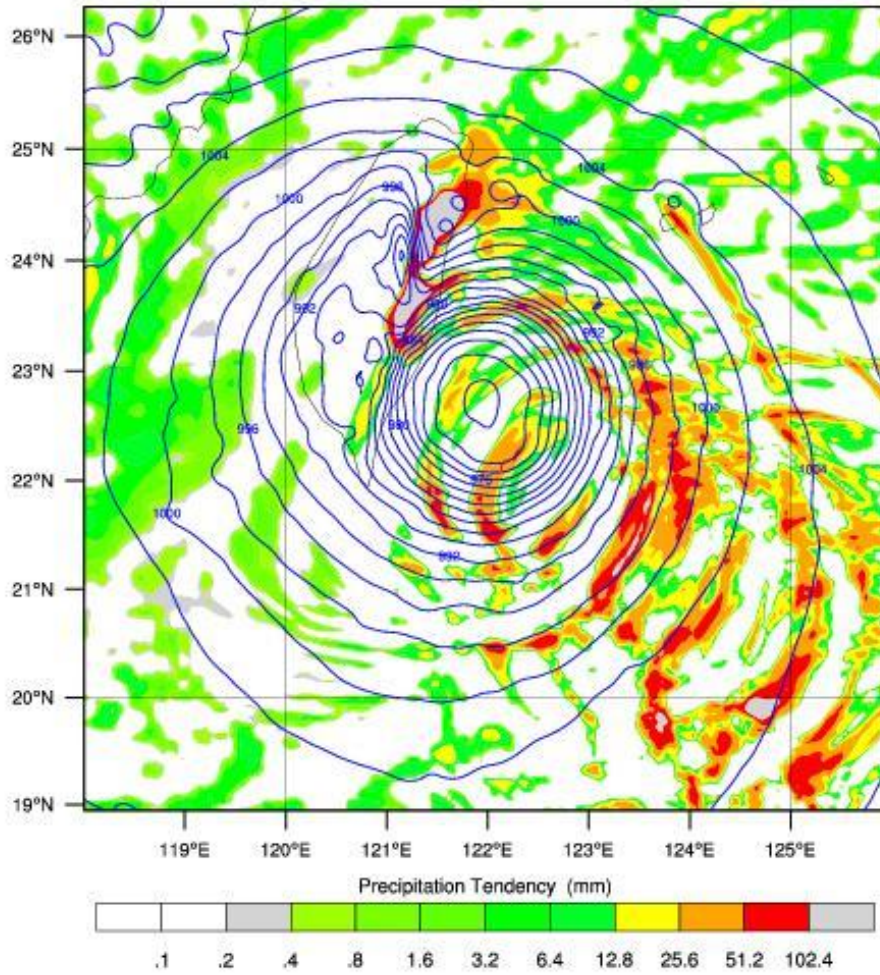
3.2 Conversion of `wrf_Precip.ncl`

This script provides three plots. Here we only use the second of the three, which plots total precipitation tendency with sea level pressure isobars. We perform similar changes as were made in the previous example, to produce one image at each time step. This example, unlike the first example, is not associated with a particular height in the scene. Note that there will be several warning messages at the first time step, since total precipitation tendency requires two times to be calculated. The modified script, [wrf_Precip_FirstMod.ncl](#), has been modified to produce one plot at each timestep. The following is the image at the third time step:

REAL-TIME WRF

Init: 2008-09-26_12:00:00
Valid: 2008-09-28_02:00:00

Precipitation Tendency from 2008-09-28_01:00:00 to 2008-09-28_02:00:00 (mm)
Sea Level Pressure (hPa)



OUTPUT FROM WRF V3.0.1.1 MODEL
WE = 202 ; SN = 202 ; Levels = 36 ; Dis = 4km ; Phys Opt = 4 ; PBL Opt = 1 ; Cu Opt = 0

To produce a geo-referenced tiff output, follow the same steps that were followed in section 3.1, resulting in the ncl script [wrf_Precip_Final.ncl](#). When you run this script, the output will be “plt_Precip.tiff”.

3.3 Conversion of wrf_CrossSection2.ncl

This script provides a vertical slice through the domain, plotting relative humidity and temperature. The original script produces 3 vertical plots, the first one at a constant x coordinate, the second at a constant y coordinate, and the third a diagonal slice with both x and y changing.

The conversion of this script is very much like the two preceding scripts, but the following should be noted:

- Since VAPOR only supports axis-aligned image planes, we use only the first plot (ip = 1). To adapt this to the Jangmi data (which is on a 200x200 horizontal grid) we set the starting and ending (x,y) coordinates to be (0,84) and (200,84). We could easily modify this to be aligned with the YZ plane instead of the XZ plane.
- Because this is a vertical plot, we do not use georeferencing. The output tiff file will contain the WRF date/time stamps, but not the georeferencing from the WRF output. To turn off georeferencing, call:

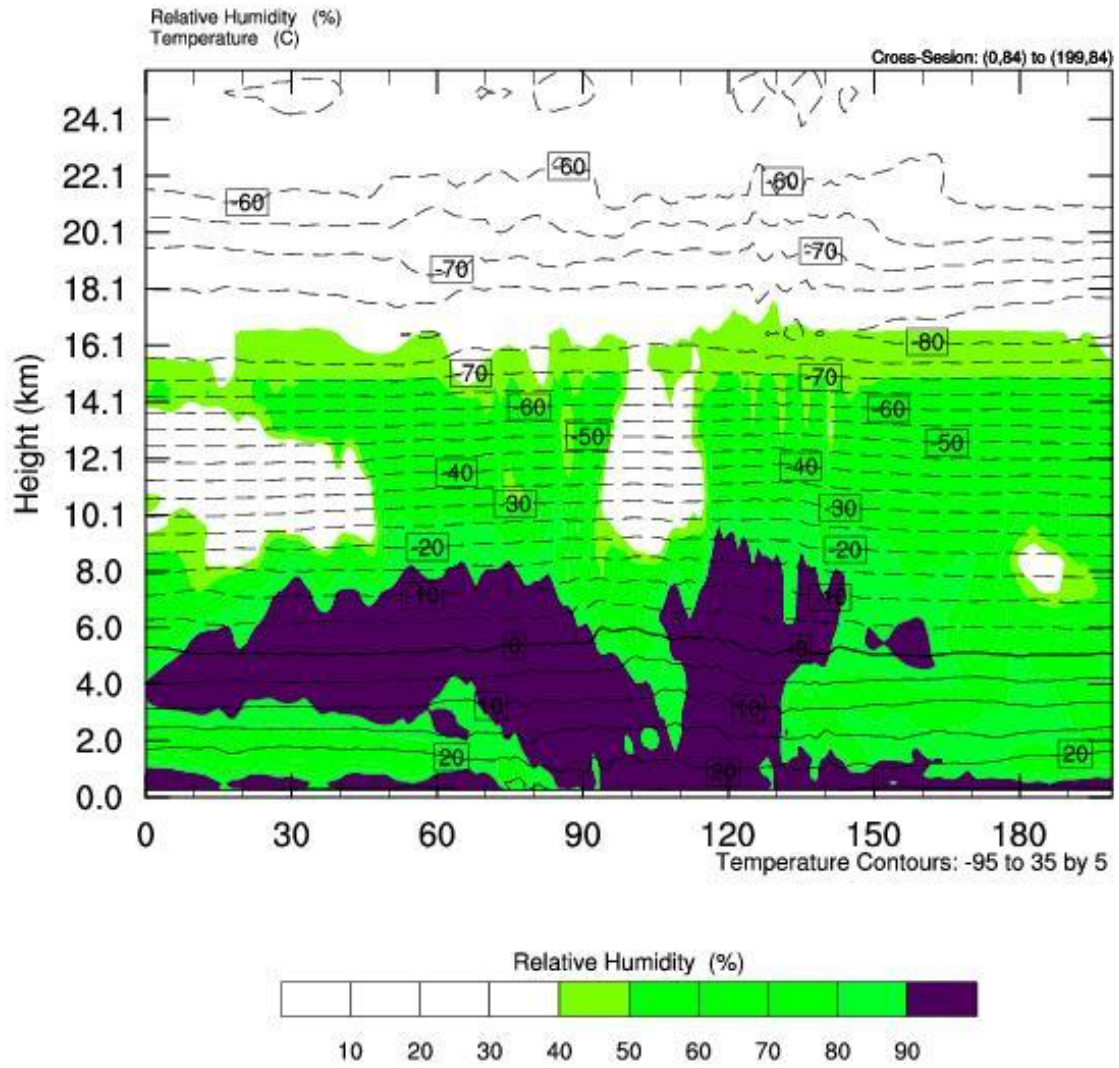
```
wrf2geotiff_disableGeoTags(wrf2gtiff)
```

```
immediately after the call to wrf2geotiff_open(wks).
```

All other changes are similar to the changes discussed in the above two sections. Below is the figure that results from the second time step of this series:

REAL-TIME WRF

Init: 2008-09-26_12:00:00
Valid: 2008-09-26_02:00:00



Refer to the [Appendix](#), which includes:

- the original script:

- [wrf CrossSection2.ncl](#)
- The version that produces one plot per timestep:
[wrf CrossSection2 FirstMod.ncl](#)
- and the final version that produces a geotiff:
[wrf CrossSection2 Final.ncl](#)

4 Obtain a terrain image for the domain of the simulation

When we display the NCL/WRF plots in VAPOR, it will be useful to have a geo-referenced terrain image, showing us the location of the typhoon relative to the local geography (e.g. the island of Taiwan). We explain here how to easily obtain an image using the VAPOR utility “`getWMSImage.sh`”.

First, we must establish the latitude and longitude range for this terrain image. Fortunately, this information was already provided when we executed `wrfvdfcreate` in [Section 2](#). The output of that command indicated that the longitude range was 117.053 to 126.302 degrees, and the latitude was from 18.7224 to 27.4392 degrees. We choose a slightly larger range to provide a geographic context.

The VAPOR shell command `getWMSImage.sh` can easily retrieve a geo-referenced image for this space. (In fact, the default behavior of this command is to obtain a satellite image for the specified rectangle.) Additional options are described in the [man page](#). For example, you can also use this command to obtain maps of political boundaries or rivers.

To obtain an image for the specified coordinates, you must be connected to the Internet, and the current directory must be writable. On Linux or Mac, you must have sourced `vapor-setup.csh`. Issue the command:

```
getWMSImage.sh -o jangmiTerrain.tiff 115 15 130 30
```

On completion, a terrain image “`jangmiTerrain.tiff`” will be written to the current directory. The latitude goes from 15 to 30 degrees North and the longitude goes from 115 to 130 degrees East.

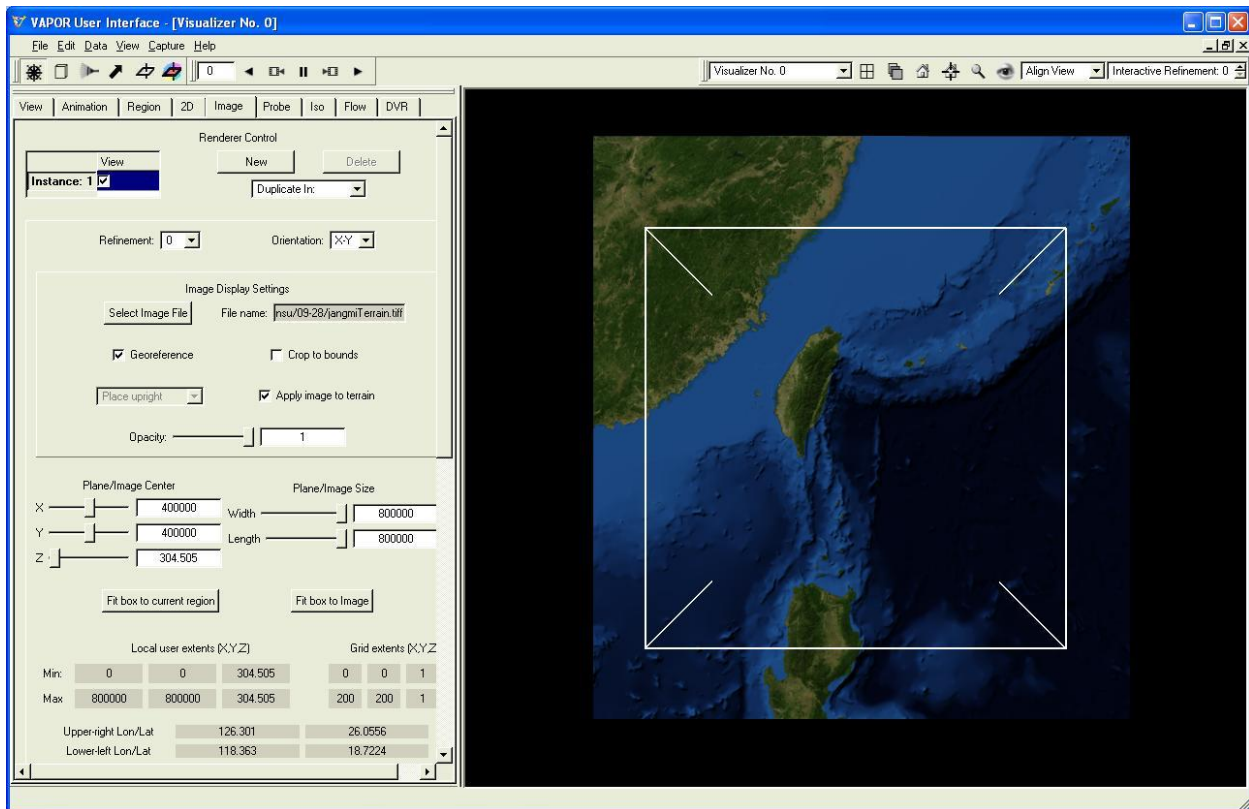
5 Display these images in the VAPOR scene

We can now use VAPOR to perform a 3D visualization of typhoon Jangmi, using the images we have created. If you are not familiar with VAPOR, we recommend that you first read the first section of the tutorial “[Visualization of WRF Data Using VAPOR: A Georgia Weather Case Study](#)”. This section will explain in detail the operations that we are about to perform.

1. Launch `vaporgui`
2. From the Data menu, Import the `wrf` output files into the current session. Alternately, if you converted the data to a VDC (as described in [Section 2](#)), then load the file “`jangmi-09-28.vdf`” into default session. You should see a white rectangular shape in the visualizer window.
3. From the Edit menu, select “Edit Visualizer Features”. Next to “Scene Stretch Factors X, Y, Z” set the values to 1, 1, 30. This will result in the 3D scene being stretched a

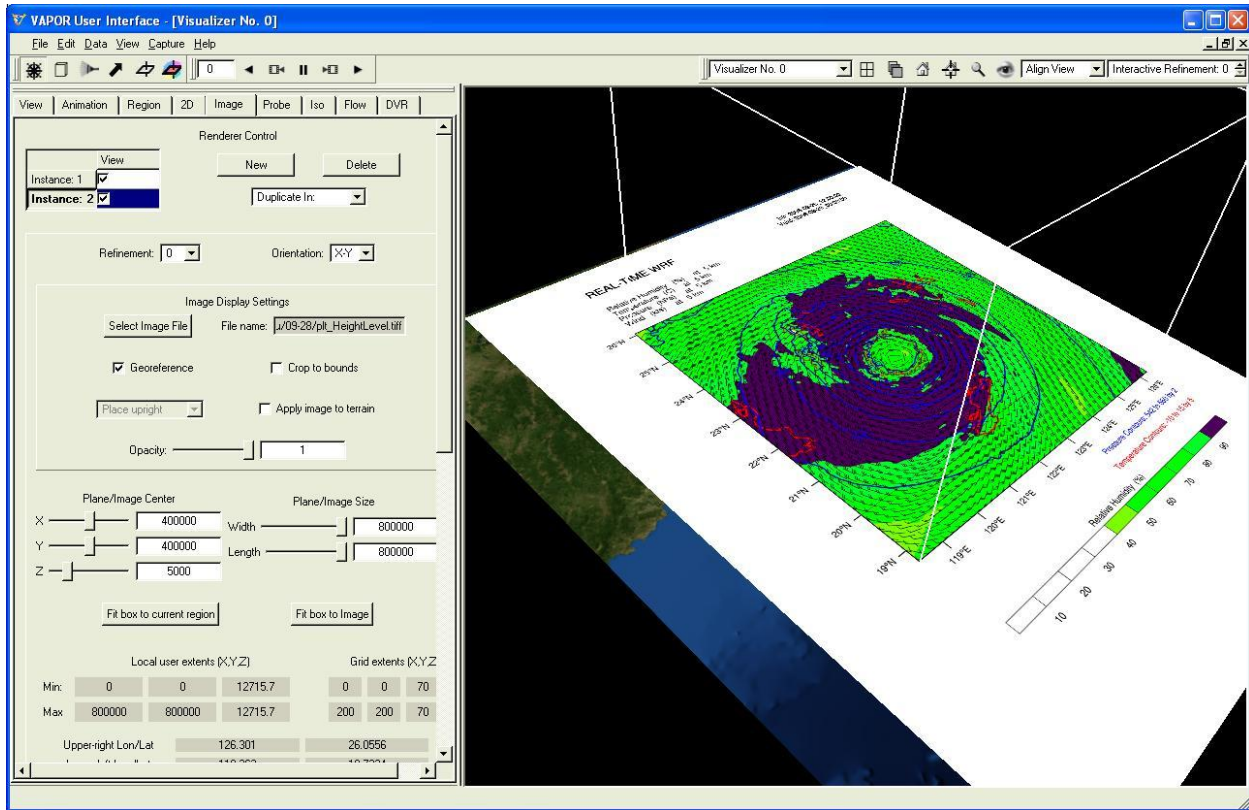
factor of 30 in the vertical (Z) direction, after you click the “OK” button. You will see that the white rectangular shape is now box-shaped.

- Click on the “Image” tab. We first display the terrain image that we created in Section 4. Click the button “Select Image File” and select the image “jangmiTerrain.tiff” that we generated. Check the “Instance 1:” checkbox in the Image tab to display the terrain image. Also check the box “Apply Image to Terrain” so that this image will appear on the earth’s surface in the scene. The VAPOR scene should look as follows:



Terrain image displayed in VAPOR Image panel

- Next, we insert the image sequence we calculated using the wrf_Height.ncl script. We place these horizontal images at elevation 5000m (this was the elevation that was used for calculation of the image by the NCL script). This is done as follows:
At the top of the Image panel, under Renderer Control, click the “New” button to create a new image renderer. Click on “Instance: 2” to select the second instance, then click the “Select Image File” button. Select the file “plt_HeightLevel.tiff” and then check the “Instance: 2” checkbox to view this image. To place it at elevation 5000m, set the Z coordinate under “Plane/Image Center” to 5000, by typing 5000 in the corresponding text box and then pressing the Enter key. Reposition the viewpoint by dragging with the left mouse button in the scene (you may also want to translate or zoom, using the middle and right buttons) so that you can see both the terrain and the data plot, as shown below:

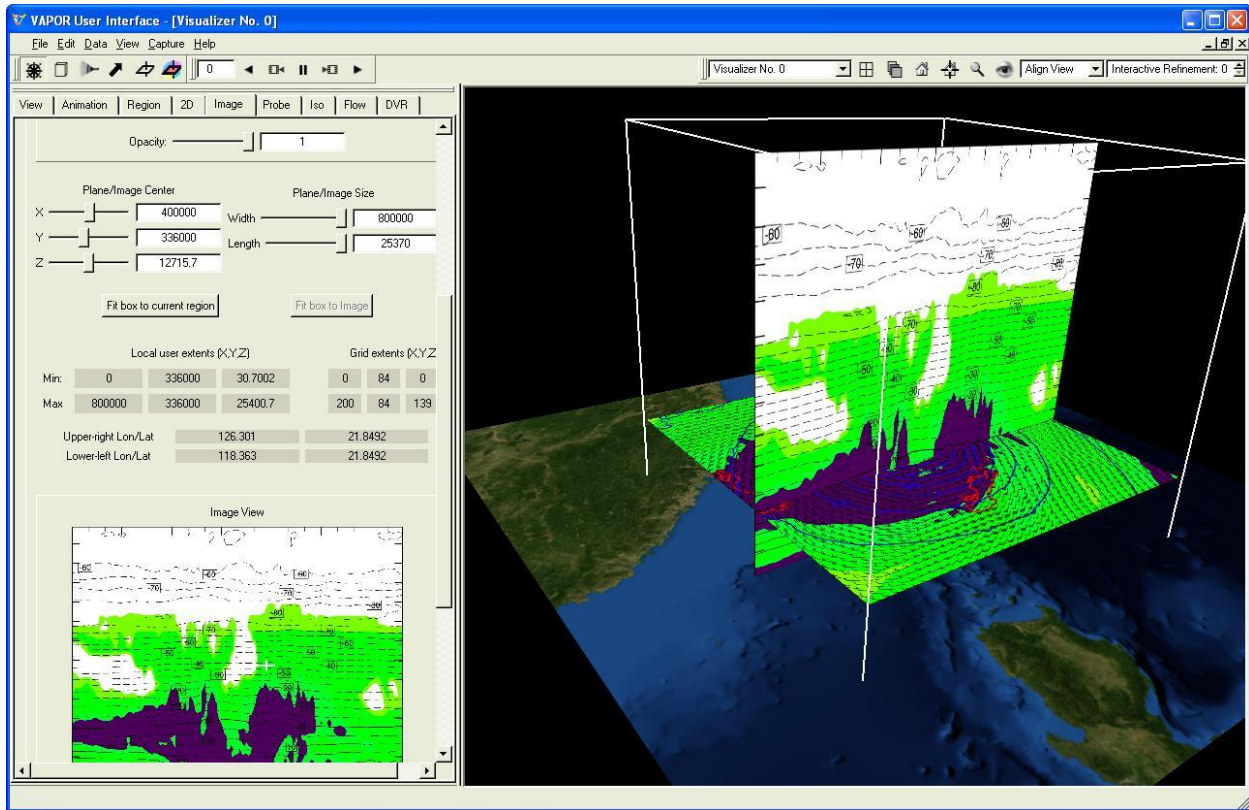


Plot of humidity, temperature, etc. at elevation 5000, with annotation

Note that the annotation in the plot is shown in the VAPOR scene. To crop this annotation, check the box “Crop to bounds” in the Image panel.

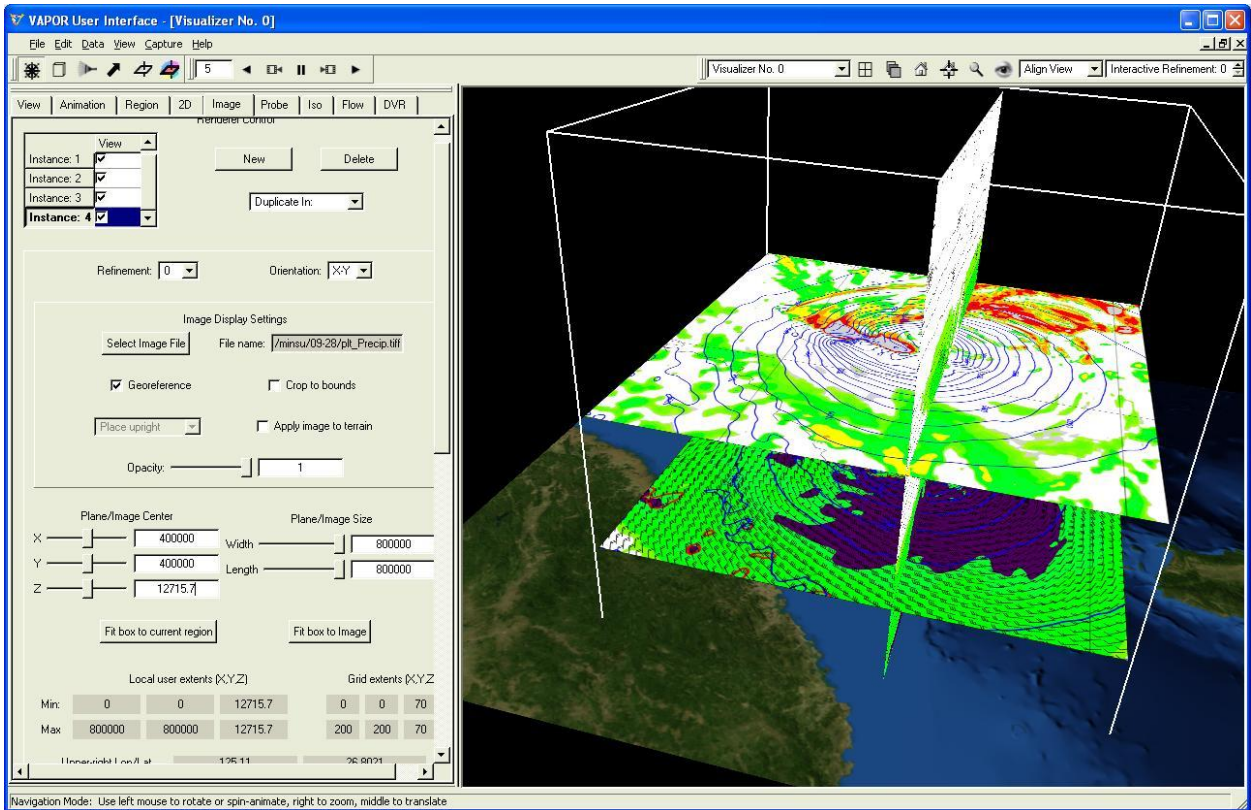
- Next we insert the vertical image calculated using `wrf_CrossSection2.ncl`. Press the “New” button under Renderer Control in the Image tab, and then click “Instance: 3” to select the new Image instance. Click the button “Select Image File” and select the file “`plt_CrossSection2.tiff`”. In the “Orientation” selector, above the file name, choose “X-Z”, since this is a vertical slice parallel to the X-Z plane. Then check the “Instance: 3” box to display the vertical slice.

By default the vertical slice is positioned in the middle of the scene, at y coordinate 400000, which corresponds to a grid coordinate of 100. The correct y-coordinate should be 84 (this y-coordinate was set in the NCL script). To move the plane to the correct value, click repeatedly on the left side of the Y slider for Plane/Image center, until the y-coordinate (displayed under Grid extents (X,Y,Z)) is 84. Alternatively just type in the value 336000 ($800000 \times 84 / 200$) for the Y coordinate of Plane/Image center, and press the Enter key. You will see the following image (note that the `wrf_Height` plot has been cropped):



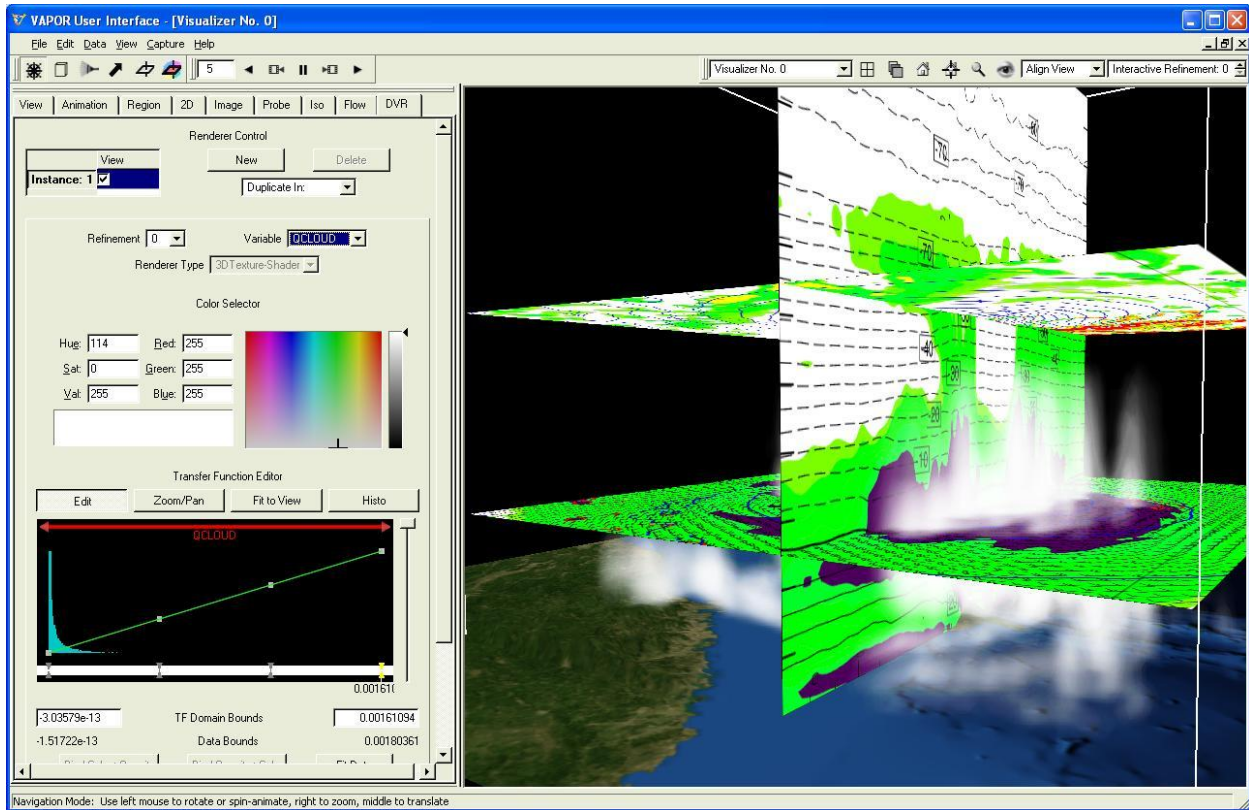
Vertical plot of humidity and temperature inserted at y coordinate 84

- Next we insert the precipitation images calculated using `wrf_Precip.ncl`. Again click on the “New” button under Renderer Control, then select “Instance: 4”, and then choose the image file “`plt_Precip.tiff`”. When you check the “Instance:4” checkbox you will see the first image of the sequence displayed at the vertical middle height of 12715m. The first image is not very interesting, so click the “play” button (▶) in the animation toolbar at the top of the VAPOR window, and play the animation forward. VAPOR display will be similar to the following at time step 5:



Precipitation plot inserted at timestep 5

8. We can use other visualization capabilities of VAPOR in conjunction with the images we have created in NCL. The 3D extent of typhoon Jangmi can be illustrated by a volume rendering of the QCLOUD variable. From the DVR panel, select the QCLOUD variable and check the “Instance:1” checkbox to enable a volume rendering. Then set the four color control points in the DVR transfer function editor to white. The resulting visualization shows the location of clouds relative to the variables plotted in the NCL images:



Volume rendering of QCLOUD combined with NCL plots and terrain image

Other VAPOR visualization tools can be used to improve visual understanding of typhoon Jangmi, such as flow integration, probe/contour planes, isosurfaces, and 2D variable display.

6 Appendix

In this are included the original NCL scripts used in section 3, as well as the modifications that were applied. In each case, the modifications (from the previous version) are indicated in red.

1. [wrf_Height.ncl](#) (original)
2. [wrf_Height_FirstMod.ncl](#)
3. [wrf_Height_Final.ncl](#)

1. [wrf_Precip.ncl](#) (original)
2. [wrf_Precip_FirstMod.ncl](#)
3. [wrf_Precip_Final.ncl](#)

1. [wrf_CrossSection2.ncl](#) (original)
2. [wrf_CrossSection2_FirstMod.ncl](#)
3. [wrf_CrossSection2_Final.ncl](#)

wrf_Height.ncl:

```

; The following NCL script was copied from the WRF/NCL site:
; http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_examples.htm

; Example script to produce plots for a WRF real-data run,
; with the ARW coordinate dynamics option.
; Interpolating to specified height levels

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

begin
;
; The WRF ARW input file.
; This needs to have a ".nc" appended, so just do it.
  a = addfile("./wrfout_d01_2000-01-24_12:00:00.nc","r")

; We generate plots, but what kind do we prefer?
  type = "x11"
; type = "pdf"
; type = "ps"
; type = "ncgm"
  wks = gsn_open_wks(type,"plt_HeightLevel")

; Set some basic resources
  res = True
  res@MainTitle = "REAL-TIME WRF"
  res@Footer = False

  pltres = True
  mpres = True
  mpres@mpGeophysicalLineColor = "Black"
  mpres@mpNationalLineColor    = "Black"
  mpres@mpUSStateLineColor     = "Black"
  mpres@mpGridLineColor        = "Black"
  mpres@mpLimbLineColor        = "Black"
  mpres@mpPerimLineColor       = "Black"

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; What times and how many time steps are in the data set?
  times = wrf_user_list_times(a) ; get times in the file
  ntimes = dimsizes(times)      ; number of times in the file

; The specific height levels that we want the data interpolated to.
  height_levels = (/ 250., 2000./) ; height levels to plot - in meter
  nlevels       = dimsizes(height_levels) ; number of height levels

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

do it = 0,ntimes-1,2 ; TIME LOOP

  print("Working on time: " + times(it) )
  res@TimeLabel = times(it) ; Set Valid time to use on plots

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; First get the variables we will need

tc = wrf_user_getvar(a,"tc",it)          ; T in C
u   = wrf_user_getvar(a,"ua",it)        ; u averaged to mass points
v   = wrf_user_getvar(a,"va",it)        ; v averaged to mass points
p   = wrf_user_getvar(a, "pressure",it) ; pressure is our vertical coordinate
z   = wrf_user_getvar(a, "z",it)        ; grid point height
rh  = wrf_user_getvar(a,"rh",it)        ; relative humidity

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

do level = 0,nlevels-1                  ; LOOP OVER LEVELS

    height = height_levels(level)

    p_plane = wrf_user_intrp3d( p,z,"h",height,0.,False)
    tc_plane = wrf_user_intrp3d(tc,z,"h",height,0.,False)
    rh_plane = wrf_user_intrp3d(rh,z,"h",height,0.,False)
    u_plane  = wrf_user_intrp3d( u,z,"h",height,0.,False)
    v_plane  = wrf_user_intrp3d( v,z,"h",height,0.,False)

    u_plane = u_plane*1.94386           ; kts
    v_plane = v_plane*1.94386           ; kts
    u_plane@units = "kts"
    v_plane@units = "kts"

; Plotting options for T
opts = res
opts@cnLineColor = "Red"
opts@ContourParameters = (/ 5.0 /)
opts@cnInfoLabelOrthogonalPosF = 0.07 ; offset second label info
opts@gsnContourLineThicknesesScale = 2.0
contour_tc = wrf_contour(a,wks,tc_plane,opts)
delete(opts)

; Plotting options for Pressure
opts = res
opts@cnLineColor = "Blue"
opts@gsnContourLineThicknesesScale = 3.0
contour_p = wrf_contour(a,wks,p_plane,opts)
delete(opts)

; Plotting options for RH
opts = res
opts@cnFillOn = True
opts@ContourParameters = (/ 10., 90., 10./)
opts@cnFillColors = (/ "White", "White", "White", \
                      "White", "Chartreuse", "Green", \
                      "Green3", "Green4", \
                      "ForestGreen", "PaleGreen4"/)
contour_rh = wrf_contour(a,wks,rh_plane,opts)
delete(opts)

```



```

; Plotting options for Wind Vectors
opts = res
opts@FieldTitle = "Wind" ; overwrite Field Title
opts@NumVectors = 47 ; wind barb density
vector = wrf_vector(a,wks,u_plane,v_plane,opts)
delete(opts)

; MAKE PLOTS
plot = wrf_map_overlays(a,wks,(/contour_rh,contour_tc,contour_p, \
vector/),pltres,mpres)

end do ; END OF LEVEL LOOP

;;;;;;;;;;;;;

end do ; END OF TIME LOOP

end

```

wrf_Height_FirstMod.ncl:

```

; Example script to produce plots for a WRF real-data run,
; with the ARW coordinate dynamics option.
; Interpolating to specified height levels

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

begin
;
; The WRF ARW input file.
; This needs to have a ".nc" appended, so just do it.

; Create a list of all the WRF output files we are using
; from the typhoon Jangmi simulation:
wrffiles = systemfunc("ls wrfout_d02_2008-09-28*")
numFiles = dimsizes(wrffiles)
do i = 0, numFiles-1
wrffiles(i) = wrffiles(i) + ".nc"
end do
inpFiles = addfiles(wrffiles,"r")

; We generate plots, but what kind do we prefer?
type = "x11"
; type = "pdf"
; type = "ps"
; type = "ncgm"
wks = gsn_open_wks(type,"plt_HeightLevel")

```

```

; Set some basic resources
res = True
res@MainTitle = "REAL-TIME WRF"
res@Footer = False

pltres = True
mpres = True
mpres@mpGeophysicalLineColor = "Black"
mpres@mpNationalLineColor    = "Black"
mpres@mpUSStateLineColor     = "Black"
mpres@mpGridLineColor        = "Black"
mpres@mpLimbLineColor        = "Black"
mpres@mpPerimLineColor       = "Black"

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; What times and how many time steps are in the data set?
do ifile = 0, numFiles-1      ; LOOP OVER FILES
  a = inpFiles[ifile]
  times = wrf_user_list_times(a) ; get times in the file
  ntimes = dimsizes(times)      ; number of times in the file

; The specific height levels that we want the data interpolated to.

; We use height of 5000m in the plot for VAPOR
; That height is near the top of the typhoon
height_levels = (/ 250., 5000./) ; height levels to plot - in meter
nlevels = dimsizes(height_levels) ; number of height levels

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Modify the time loop to include all time steps, not just every odd one
do it = 0, ntimes-1          ; TIME LOOP

  print("Working on time: " + times(it) )
  res@TimeLabel = times(it) ; Set Valid time to use on plots

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; First get the variables we will need

  tc = wrf_user_getvar(a, "tc", it) ; T in C
  u = wrf_user_getvar(a, "ua", it) ; u averaged to mass points
  v = wrf_user_getvar(a, "va", it) ; v averaged to mass points
  p = wrf_user_getvar(a, "pressure", it) ; pressure is our vertical coordinate
  z = wrf_user_getvar(a, "z", it) ; grid point height
  rh = wrf_user_getvar(a, "rh", it) ; relative humidity

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Modify loop over level to just do one level = 1
level = 1 ; just one level for the georeferenced plot

  height = height_levels(level)

  p_plane = wrf_user_intrp3d( p, z, "h", height, 0., False)

```

```

tc_plane = wrf_user_intrp3d(tc,z,"h",height,0.,False)
rh_plane = wrf_user_intrp3d(rh,z,"h",height,0.,False)
u_plane  = wrf_user_intrp3d( u,z,"h",height,0.,False)
v_plane  = wrf_user_intrp3d( v,z,"h",height,0.,False)

u_plane = u_plane*1.94386      ; kts
v_plane = v_plane*1.94386      ; kts
u_plane@units = "kts"
v_plane@units = "kts"

; Plotting options for T
opts = res
opts@cnLineColor = "Red"
opts@ContourParameters = (/ 5.0 /)
opts@cnInfoLabelOrthogonalPosF = 0.07 ; offset second label info
opts@gsnContourLineThicknesesScale = 2.0
contour_tc = wrf_contour(a,wks,tc_plane,opts)
delete(opts)

; Plotting options for Pressure
opts = res
opts@cnLineColor = "Blue"
opts@gsnContourLineThicknesesScale = 3.0
contour_p = wrf_contour(a,wks,p_plane,opts)
delete(opts)

; Plotting options for RH
opts = res
opts@cnFillOn = True
opts@ContourParameters = (/ 10., 90., 10./)
opts@cnFillColors = (/ "White", "White", "White", \
                      "White", "Chartreuse", "Green", \
                      "Green3", "Green4", \
                      "ForestGreen", "PaleGreen4"/)
contour_rh = wrf_contour(a,wks,rh_plane,opts)
delete(opts)

; Plotting options for Wind Vectors
opts = res
opts@FieldTitle = "Wind" ; overwrite Field Title
opts@NumVectors = 47 ; wind barb density
vector = wrf_vector(a,wks,u_plane,v_plane,opts)
delete(opts)

; MAKE PLOTS
plot = wrf_map_overlays(a,wks,(/contour_rh,contour_tc,contour_p, \
vector/),pltres,mpres)

; comment out the end of the level loop:
; end do ; END OF LEVEL LOOP

```

```

;;;;;;;;;;;;;

end do          ; END OF TIME LOOP

; End loop over files that we introduced:
end do        ; END OF LOOP OVER FILES

end

```

wrf_Height_Final.ncl:

```

; Example script to produce plots for a WRF real-data run,
; with the ARW coordinate dynamics option.
; Interpolating to specified height levels

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

; Load the wrf2geotiff library.
; The wrf2geotiff library is installed with VAPOR:
load "$VAPOR_HOME/share/examples/NCL/wrf2geotiff.ncl"

begin
;
; The WRF ARW input file.
; This needs to have a ".nc" appended, so just do it.

wrffiles = systemfunc("ls wrfout_d02_2008-09-28*")
numFiles = dimsizes(wrffiles)
do i = 0, numFiles-1
    wrffiles(i) = wrffiles(i) + ".nc"
end do
inpFiles = addfiles(wrffiles,"r")

; We generate plots, but what kind do we prefer?

; When using wrf2geotiff, type must be ps
; type = "x11"
; type = "pdf"
type = "ps"
; type = "ncgm"
wks = gsn_open_wks(type,"plt_HeightLevel")
; Create the wrf2gtiff opaque pointer, which will
; be referenced in all geotiff changes
wrf2gtiff = wrf2geotiff_open(wks)

; Set some basic resources
res = True
res@MainTitle = "REAL-TIME WRF"
res@Footer = False

; For wrf2geotiff we need to control frame advance:
pltres = True

```

```

pltres@gsnFrame = False

mpres = True
mpres@mpGeophysicalLineColor = "Black"
mpres@mpNationalLineColor   = "Black"
mpres@mpUSStateLineColor    = "Black"
mpres@mpGridLineColor       = "Black"
mpres@mpLimbLineColor        = "Black"
mpres@mpPerimLineColor      = "Black"

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; What times and how many time steps are in the data set?
do ifile = 0, numFiles-1      ; LOOP OVER FILES
  a = inpFiles[ifile]
  times = wrf_user_list_times(a) ; get times in the file
  ntimes = dimsizes(times)      ; number of times in the file

; The specific height levels that we want the data interpolated to.
height_levels = (/ 250., 5000./) ; height levels to plot - in meter
nlevels = dimsizes(height_levels) ; number of height levels

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

do it = 0, ntimes-1          ; TIME LOOP

  print("Working on time: " + times(it) )
  res@TimeLabel = times(it)   ; Set Valid time to use on plots

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; First get the variables we will need

  tc = wrf_user_getvar(a,"tc",it) ; T in C
  u = wrf_user_getvar(a,"ua",it) ; u averaged to mass points
  v = wrf_user_getvar(a,"va",it) ; v averaged to mass points
  p = wrf_user_getvar(a, "pressure",it) ; pressure is our vertical coordinate
  z = wrf_user_getvar(a, "z",it) ; grid point height
  rh = wrf_user_getvar(a,"rh",it) ; relative humidity

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

level = 1 ; just one level for the georeferenced plot
height = height_levels(level)

p_plane = wrf_user_intrp3d( p,z,"h",height,0.,False)
tc_plane = wrf_user_intrp3d( tc,z,"h",height,0.,False)
rh_plane = wrf_user_intrp3d( rh,z,"h",height,0.,False)
u_plane = wrf_user_intrp3d( u,z,"h",height,0.,False)
v_plane = wrf_user_intrp3d( v,z,"h",height,0.,False)

u_plane = u_plane*1.94386 ; kts
v_plane = v_plane*1.94386 ; kts
u_plane@units = "kts"
v_plane@units = "kts"

```



```

; Plotting options for T
opts = res
opts@cnLineColor = "Red"
opts@ContourParameters = (/ 5.0 /)
opts@cnInfoLabelOrthogonalPosF = 0.07 ; offset second label information
opts@gsnContourLineThicknessesScale = 2.0
contour_tc = wrf_contour(a,wks,tc_plane,opts)
delete(opts)

; Plotting options for Pressure
opts = res
opts@cnLineColor = "Blue"
opts@gsnContourLineThicknessesScale = 3.0
contour_p = wrf_contour(a,wks,p_plane,opts)
delete(opts)

; Plotting options for RH
opts = res
opts@cnFillOn = True
opts@ContourParameters = (/ 10., 90., 10./)
opts@cnFillColors = (/ "White", "White", "White", \
                      "White", "Chartreuse", "Green", \
                      "Green3", "Green4", \
                      "ForestGreen", "PaleGreen4"/)
contour_rh = wrf_contour(a,wks,rh_plane,opts)
delete(opts)

; Plotting options for Wind Vectors
opts = res
opts@FieldTitle = "Wind" ; overwrite Field Title
opts@NumVectors = 47 ; wind barb density
vector = wrf_vector(a,wks,u_plane,v_plane,opts)
delete(opts)

; MAKE PLOTS
plot = wrf_map_overlays(a,wks,(/contour_rh,contour_tc,contour_p, \
vector/),pltres,mpres)

; capture plot georeferencing information
; then change the frame
; Don't crop (leave annotation intact)
wrf2geotiff_write(wrf2gtiff, a, times(it), wks, plot, False)
frame(wks)

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

end do ; END OF TIME LOOP
end do ; END OF LOOP OVER FILES

```

```
; Call wrf2geotiff_close at the end
wrf2geotiff_close(wrf2gtiff, wks)
```

end

wrf_Precip.ncl:

```
; The following NCL script was copied from the WRF/NCL site:
; http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_examples.htm
; Example script to produce plots for a WRF real-data run,
; with the ARW coordinate dynamics option.
```

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

```
begin
```

```
;
; The WRF ARW input file.
; This needs to have a ".nc" appended, so just do it.
a = addfile("./wrfout_d01_2000-01-24_12:00:00.nc","r")
```

```
; We generate plots, but what kind do we prefer?
type = "x11"
type = "pdf"
; type = "ps"
; type = "ncgm"
wks = gsn_open_wks(type,"plt_Precip")
```

```
; Set some basic resources
res = True
res@MainTitle = "REAL-TIME WRF"
```

```
pltres = True
mpres = True
mpres@mpGeophysicalLineColor = "Black"
mpres@mpNationalLineColor = "Black"
mpres@mpUSStateLineColor = "Black"
mpres@mpGridLineColor = "Black"
mpres@mpLimbLineColor = "Black"
mpres@mpPerimLineColor = "Black"
```

```
;;;;;;;;;;;;;
;;;;;;;;;;;;;
```

```
; What times and how many time steps are in the data set?
FirstTime = True
times = wrf_user_list_times(a) ; get times in the file
ntimes = dimsizes(times) ; number of times in the file
```

```
;;;;;;;;;;;;;
```

```

do it = 0, ntimes-1, 2 ; TIME LOOP

print("Working on time: " + times(it) )
if (FirstTime) then ; Save some times for tracking tendencies
  times_sav = times(it)
end if
res@TimeLabel = times(it) ; Set Valid time to use on plots

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; First get the variables we will need

slp = wrf_user_getvar(a, "slp", it) ; slp
wrf_smooth_2d( slp, 3 ) ; smooth slp

; Get non-convective, convective and total precipitation
; Calculate tendency values
rain_exp = wrf_user_getvar(a, "RAINNC", it)
rain_con = wrf_user_getvar(a, "RAINC", it)
rain_tot = rain_exp + rain_con
rain_tot@description = "Total Precipitation"

if( FirstTime ) then
  if ( it .eq. 0 ) then
    rain_exp_save = rain_exp
    rain_con_save = rain_con
    rain_tot_save = rain_tot
  else
    rain_exp_save = wrf_user_getvar(a, "RAINNC", it-1)
    rain_con_save = wrf_user_getvar(a, "RAINC", it-1)
    rain_tot_save = rain_exp_save + rain_con_save
    FirstTime = False
    times_sav = times(it-1)
  end if
end if

rain_exp_tend = rain_exp - rain_exp_save
rain_con_tend = rain_con - rain_con_save
rain_tot_tend = rain_tot - rain_tot_save
rain_exp_tend@description = "Explicit Precipitation Tendency"
rain_con_tend@description = "Param Precipitation Tendency"
rain_tot_tend@description = "Precipitation Tendency"

; Bookkeeping, just to allow the tendency at the next time step
rain_exp_save = rain_exp
rain_con_save = rain_con
rain_tot_save = rain_tot

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

if( .not. FirstTime ) then ; We will skip the first time

; Plotting options for Sea Level Pressure
opts_psl = res
opts_psl@ContourParameters = (/ 900., 1100., 2. /)
opts_psl@cnLineColor = "Blue"
opts_psl@cnInfoLabelOn = False
opts_psl@cnLineLabelFontHeightF = 0.01

```

```

opts_psl@cnLineLabelPerimOn = False
opts_psl@gsnContourLineThicknessesScale = 1.5
contour_psl = wrf_contour(a,wks,slp,opts_psl)
delete(opts_psl)

; Plotting options for Precipitation
opts_r = res
opts_r@UnitLabel           = "mm"
opts_r@cnLevelSelectionMode = "ExplicitLevels"
opts_r@cnLevels            = (/ .1, .2, .4, .8, 1.6, 3.2, 6.4, \
                             12.8, 25.6, 51.2, 102.4/)
opts_r@cnFillColors        = (/ "White", "White", "DarkOliveGreen1", \
                             "DarkOliveGreen3", "Chartreuse", \
                             "Chartreuse3", "Green", "ForestGreen", \
                             "Yellow", "Orange", "Red", "Violet"/)
opts_r@cnInfoLabelOn      = False
opts_r@cnConstFLabelOn    = False
opts_r@cnFillOn           = True

; Total Precipitation (color fill)
contour_tot = wrf_contour(a,wks, rain_tot, opts_r)

; Precipitation Tendencies
opts_r@SubFieldTitle = "from " + times_sav + " to " + times(it)
; total (color)

contour_tend = wrf_contour(a,wks, rain_tot_tend,opts_r)
contour_res = wrf_contour(a,wks,rain_exp_tend,opts_r) ; exp (color)
opts_r@cnFillOn = False
opts_r@cnLineColor = "Red4"
; con (red lines)

contour_prm = wrf_contour(a,wks,rain_con_tend,opts_r)
delete(opts_r)

; MAKE PLOTS

; Total Precipitation
plot = wrf_map_overlays(a,wks,contour_tot,pltres,mpres)

; Total Precipitation Tendency + SLP
plot = \
wrf_map_overlays(a,wks, (/contour_tend,contour_psl/),pltres,mpres)

; Non-Convective and Convective Precipitation Tendencies
plot = \
wrf_map_overlays(a,wks, (/contour_res,contour_prm/),pltres,mpres)

end if ; END IF FOR SKIPPING FIRST TIME

;;;;;;;;;;;;;;

times_sav = times(it)

```

```

        FirstTime = False
    end do          ; END OF TIME LOOP

end

```

wrf_Precip_FirstMod.ncl:

```

; Example script to produce plots for a WRF real-data run,
; with the ARW coordinate dynamics option.

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

begin
;
; The WRF ARW input file.
; This needs to have a ".nc" appended, so just do it.

; Instead of selecting one file, make a variable that has
; all the names of the WRF output files in it:
; a = addfile("wrfout_d02_2008-09-28_00.nc","r")
wrffiles = systemfunc("ls wrfout_d02_2008-09-28*")
numFiles = dimsizes(wrffiles)
do i = 0, numFiles-1
    wrffiles(i) = wrffiles(i)+".nc"
end do
inpFiles = addfiles(wrffiles,"r")

; We generate plots, but what kind do we prefer?
type = "x11"
; type = "pdf"
; type = "ps"
; type = "ncgm"
wks = gsn_open_wks(type,"plt_Precip")

; Set some basic resources
res = True
res@MainTitle = "REAL-TIME WRF"

pltres = True
mpres = True
mpres@mpGeophysicalLineColor = "Black"
mpres@mpNationalLineColor    = "Black"
mpres@mpUSStateLineColor     = "Black"
mpres@mpGridLineColor        = "Black"
mpres@mpLimbLineColor         = "Black"
mpres@mpPerimLineColor       = "Black"

;;;;;;;;;;;;;
;;;;;;;;;;;;;

; What times and how many time steps are in the data set?
FirstTime = True

```



```

; Make a loop over all the files:
do ifile = 0,numFiles -1
  a = inpFiles[ifile]

  times = wrf_user_list_times(a) ; get times in the file
  ntimes = dimsizes(times) ; number of times in the file

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Loop over all the times, not just the odd times:
; do it = 0,ntimes-1,2 ; TIME LOOP
do it = 0, ntimes-1 ; Modified to do every time step

  print("Working on time: " + times(it) )
  if (FirstTime) then ; Save some times for tracking tendencies
    times_sav = times(it)
  end if
  res@TimeLabel = times(it) ; Set Valid time to use on plots

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; First get the variables we will need

  slp = wrf_user_getvar(a,"slp",it) ; slp
  wrf_smooth_2d( slp, 3 ) ; smooth slp

; Get non-convective, convective and total precipitation
; Calculate tendency values
rain_exp = wrf_user_getvar(a,"RAINNC",it)
rain_con = wrf_user_getvar(a,"RAINC",it)
rain_tot = rain_exp + rain_con
rain_tot@description = "Total Precipitation"

if( FirstTime ) then
  if ( it .eq. 0 ) then
    rain_exp_save = rain_exp
    rain_con_save = rain_con
    rain_tot_save = rain_tot
  else
    rain_exp_save = wrf_user_getvar(a,"RAINNC",it-1)
    rain_con_save = wrf_user_getvar(a,"RAINC",it-1)
    rain_tot_save = rain_exp_save + rain_con_save
    FirstTime = False
    times_sav = times(it-1)
  end if
end if

rain_exp_tend = rain_exp - rain_exp_save
rain_con_tend = rain_con - rain_con_save
rain_tot_tend = rain_tot - rain_tot_save
rain_exp_tend@description = "Explicit Precipitation Tendency"
rain_con_tend@description = "Param Precipitation Tendency"
rain_tot_tend@description = "Precipitation Tendency"

; Bookkeeping, just to allow the tendency at the next time step
rain_exp_save = rain_exp
rain_con_save = rain_con

```

```

rain_tot_save = rain_tot

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Don't skip the first time: we need an image at each time stamp
; This will result in a warning message on the first plot
; if( .not. FirstTime ) then      ; We don't skip the first time

; Plotting options for Sea Level Pressure
opts_psl = res
opts_psl@ContourParameters = (/ 900., 1100., 2. /)
opts_psl@cnLineColor      = "Blue"
opts_psl@cnInfoLabelOn    = False
opts_psl@cnLineLabelFontHeightF = 0.01
opts_psl@cnLineLabelPerimOn = False
opts_psl@gsnContourLineThicknessesScale = 1.5
contour_psl = wrf_contour(a,wks,slp,opts_psl)
delete(opts_psl)

; Plotting options for Precipitation
opts_r = res
opts_r@UnitLabel          = "mm"
opts_r@cnLevelSelectionMode = "ExplicitLevels"
opts_r@cnLevels           = (/ .1, .2, .4, .8, 1.6, 3.2, 6.4, \
                             12.8, 25.6, 51.2, 102.4/)
opts_r@cnFillColors       = (/ "White", "White", "DarkOliveGreen1", \
                             "DarkOliveGreen3", "Chartreuse", \
                             "Chartreuse3", "Green", "ForestGreen", \
                             "Yellow", "Orange", "Red", "Violet"/)
opts_r@cnInfoLabelOn      = False
opts_r@cnConstFLabelOn    = False
opts_r@cnFillOn           = True

; Total Precipitation (color fill)
contour_tot = wrf_contour(a,wks, rain_tot, opts_r)

; Precipitation Tendencies
opts_r@SubFieldTitle = "from " + times_sav + " to " + times(it)
; total (color)
contour_tend = wrf_contour(a,wks, rain_tot_tend,opts_r)
contour_res = wrf_contour(a,wks,rain_exp_tend,opts_r) ; exp (color)
opts_r@cnFillOn = False
opts_r@cnLineColor = "Red4"
; con (red lines)
contour_prm = wrf_contour(a,wks,rain_con_tend,opts_r)
delete(opts_r)

; MAKE PLOTS

; Total Precipitation
; Don't do this plot for geotiff:
; plot = wrf_map_overlays(a,wks,contour_tot,pltres,mpres)

```

```

        ; Total Precipitation Tendency + SLP
        ; Do just this one plot:
        plot = \
wrf_map_overlays(a,wks, (/contour_tend,contour_psl/),pltres,mpres)

        ; Non-Convective and Convective Precipitation Tendencies

        ; Don't do this plot either:
        ; plot =
wrf_map_overlays(a,wks, (/contour_res,contour_prm/),pltres,mpres)

;   end if   ; END IF FOR SKIPPING FIRST TIME - commented out

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        times_sav = times(it)
        FirstTime = False
    end do      ; END OF TIME LOOP
end do      ; END OF LOOP OVER FILES

end

```

wrf_Precip_Final.ncl:

```

;   Example script to produce plots for a WRF real-data run,
;   with the ARW coordinate dynamics option.

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

; load the wrf2geotiff library from the VAPOR distribution:
load "$VAPOR_HOME/share/examples/NCL/wrf2geotiff.ncl"

begin
;
; The WRF ARW input file.
; This needs to have a ".nc" appended, so just do it.
; a = addfile("wrfout_d02_2008-09-28_00.nc","r")
wrf_files = systemfunc("ls wrfout_d02_2008-09-28*")
numFiles = dimsizes(wrf_files)
do i = 0, numFiles-1
    wrf_files(i) = wrf_files(i)+".nc"
end do
inpFiles = addfiles(wrf_files,"r")

; We generate plots, but what kind do we prefer?
; For wrf2geotiff, the type MUST be ps
; type = "x11"
; type = "pdf"
type = "ps"
; type = "ncgm"
wks = gsn_open_wks(type,"plt_Precip")

```

```

; wrf2geotiff_open() creates the opaque pointer wrf2gtiff:
wrf2gtiff = wrf2geotiff_open(wks)

; Set some basic resources
res = True
res@MainTitle = "REAL-TIME WRF"

; Following are needed for control of frame advance:
pltres = True
pltres@gsnFrame = False

mpres = True
mpres@mpGeophysicalLineColor = "Black"
mpres@mpNationalLineColor = "Black"
mpres@mpUSStateLineColor = "Black"
mpres@mpGridLineColor = "Black"
mpres@mpLimbLineColor = "Black"
mpres@mpPerimLineColor = "Black"

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; What times and how many time steps are in the data set?
FirstTime = True
do ifile = 0,numFiles -1
a = inpFiles[ifile]
times = wrf_user_list_times(a) ; get times in the file
ntimes = dimsizes(times) ; number of times in the file

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; do it = 0,ntimes-1,2 ; TIME LOOP
do it = 0, ntimes-1 ; Modified to do every time step

print("Working on time: " + times(it) )
if (FirstTime) then ; Save some times for tracking tendencies
times_sav = times(it)
end if
res@TimeLabel = times(it) ; Set Valid time to use on plots

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; First get the variables we will need

slp = wrf_user_getvar(a,"slp",it) ; slp
wrf_smooth_2d( slp, 3 ) ; smooth slp

; Get non-convective, convective and total precipitation
; Calculate tendency values
rain_exp = wrf_user_getvar(a,"RAINNC",it)
rain_con = wrf_user_getvar(a,"RAINC",it)
rain_tot = rain_exp + rain_con
rain_tot@description = "Total Precipitation"

if( FirstTime ) then
if ( it .eq. 0 ) then
rain_exp_save = rain_exp

```

```

        rain_con_save = rain_con
        rain_tot_save = rain_tot
    else
        rain_exp_save = wrf_user_getvar(a,"RAINNC",it-1)
        rain_con_save = wrf_user_getvar(a,"RAINC",it-1)
        rain_tot_save = rain_exp_save + rain_con_save
        FirstTime = False
        times_sav = times(it-1)
    end if
end if

rain_exp_tend = rain_exp - rain_exp_save
rain_con_tend = rain_con - rain_con_save
rain_tot_tend = rain_tot - rain_tot_save
rain_exp_tend@description = "Explicit Precipitation Tendency"
rain_con_tend@description = "Param Precipitation Tendency"
rain_tot_tend@description = "Precipitation Tendency"

; Bookkeeping, just to allow the tendency at the next time step
rain_exp_save = rain_exp
rain_con_save = rain_con
rain_tot_save = rain_tot

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Don't skip the first time: we need an image at each time stamp
; if( .not. FirstTime ) then

; Plotting options for Sea Level Pressure
opts_psl = res
opts_psl@ContourParameters = (/ 900., 1100., 2. /)
opts_psl@cnLineColor      = "Blue"
opts_psl@cnInfoLabelOn   = False
opts_psl@cnLineLabelFontHeightF = 0.01
opts_psl@cnLineLabelPerimOn = False
opts_psl@gsnContourLineThicknessesScale = 1.5
contour_psl = wrf_contour(a,wks,slp,opts_psl)
delete(opts_psl)

; Plotting options for Precipitation
opts_r = res
opts_r@UnitLabel          = "mm"
opts_r@cnLevelSelectionMode = "ExplicitLevels"
opts_r@cnLevels           = (/ .1, .2, .4, .8, 1.6, 3.2, 6.4, \
                             12.8, 25.6, 51.2, 102.4/)
opts_r@cnFillColors       = (/ "White", "White", "DarkOliveGreen1", \
                             "DarkOliveGreen3", "Chartreuse", \
                             "Chartreuse3", "Green", "ForestGreen", \
                             "Yellow", "Orange", "Red", "Violet"/)
opts_r@cnInfoLabelOn     = False
opts_r@cnConstFLabelOn  = False
opts_r@cnFillOn          = True

; Total Precipitation (color fill)
contour_tot = wrf_contour(a,wks, rain_tot, opts_r)

```

```

; Precipitation Tendencies
opts_r@SubFieldTitle = "from " + times_sav + " to " + times(it)
; total (color):
contour_tend = wrf_contour(a,wks, rain_tot_tend,opts_r)
; exp (color):
contour_res = wrf_contour(a,wks,rain_exp_tend,opts_r)
opts_r@cnFillOn = False
opts_r@cnLineColor = "Red4"
; con (red lines)
contour_prm = wrf_contour(a,wks,rain_con_tend,opts_r)
delete(opts_r)

; MAKE PLOTS

; Total Precipitation: Don't plot
; plot = wrf_map_overlays(a,wks,contour_tot,pltres,mpres)

; Total Precipitation Tendency + SLP : Do plot
plot =
wrf_map_overlays(a,wks,(/contour_tend,contour_psl/),pltres,mpres)

; Then put it in geotiff
; Do crop the image to the domain bounds
wrf2geotiff_write(wrf2gtiff, a, times(it), wks, plot, True)
; The frame is needed between time steps after the wrf2geotiff_write
frame(wks)

; Non-Convective and Convective Precipitation Tendencies: Don't plot
; plot =
wrf_map_overlays(a,wks,(/contour_res,contour_prm/),pltres,mpres)

; end if ; END IF FOR SKIPPING FIRST TIME - commented out

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

times_sav = times(it)
FirstTime = False
end do ; END OF TIME LOOP
end do ; END OF LOOP OVER FILES

; At the end, close the wrf2geotiff
; This is where the tiff writing takes place:
wrf2geotiff_close(wrf2gtiff,wks)

end

```

wrf_CrossSection2.ncl:

```

; The following NCL script was copied from the WRF/NCL site:
; http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL\_examples.htm

```

```

; Example script to produce plots for a WRF real-data run,
; with the ARW coordinate dynamics option.
; Plot data on a cross section
; This script will plot data from a a given point A to point B
; Vertical coordinate is height

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

begin
;
; The WRF ARW input file.
; This needs to have a ".nc" appended, so just do it.
  a = addfile("./wrfout_d01_2000-01-24_12:00:00.nc","r")

; We generate plots, but what kind do we prefer?
  type = "x11"
; type = "pdf"
; type = "ps"
; type = "ncgm"
  wks = gsn_open_wks(type,"plt_CrossSection2")

; Set some basic resources
  res = True
  res@MainTitle = "REAL-TIME WRF"

  pltres = True

;-----
FirstTime = True
times = wrf_user_list_times(a) ; get times in the file
ntimes = dimsizes(times) ; number of times in the file

mdims = getfilevardimsizes(a,"P") ; get some dimension sizes for the file
nd = dimsizes(mdims)

;-----
do it = 0,ntimes-1,2 ; TIME LOOP

  print("Working on time: " + times(it) )
  res@TimeLabel = times(it) ; Set Valid time to use on plots

  tc = wrf_user_getvar(a,"tc",it) ; T in C
  rh = wrf_user_getvar(a,"rh",it) ; relative humidity
  z = wrf_user_getvar(a,"z",it) ; grid point height

  if ( FirstTime ) then ; get height info for labels
    zmin = 0.
    zmax = max(z)/1000.
    nz = floattoint(zmax/2 + 1)
    FirstTime = False

```



```
end if
```

```
;-----
```

```
do ip = 1, 3 ; we are doing 3 plots, specifying different start and end points
```

```
    opts = True ; setting start and end times  
    plane = new(4,float)
```

```
    if(ip .eq. 1) then  
        plane = (/ 40,81, 259,81 /) ; start x;y & end x;y point  
    end if
```

```
    if(ip .eq. 2) then  
        plane = (/ 130,1, 130,162 /) ; start x;y & end x;y point  
    end if
```

```
    if(ip .eq. 3) then  
        plane = (/ 49,1, 210,162 /) ; start x;y & end x;y point  
    end if
```

```
    rh_plane = wrf_user_intrp3d(rh,z,"v",plane,0.,opts)  
    tc_plane = wrf_user_intrp3d(tc,z,"v",plane,0.,opts)
```

```
    dim = dimsizes(rh_plane) ; Find the data span - for use in labels  
    zspan = dim(0)
```

```
; Options for XY Plots
```

```
    opts_xy = res  
    opts_xy@tiYAxisString = "Height (km)"  
    opts_xy@AspectRatio = 0.75  
    opts_xy@cnMissingValPerimOn = True  
    opts_xy@cnMissingValFillColor = 0  
    opts_xy@cnMissingValFillPattern = 11  
    opts_xy@tmYLMode = "Explicit"  
    ; Create tick marks  
    opts_xy@tmYLValues = fspan(0,zspan,nz)  
; Create labels  
    opts_xy@tmYLLabels = sprintf("%.1f",fspan(zmin,zmax,nz))  
    opts_xy@tiXAxisFontHeightF = 0.020  
    opts_xy@tiYAxisFontHeightF = 0.020  
    opts_xy@tmXBMajorLengthF = 0.02  
    opts_xy@tmYLMajorLengthF = 0.02  
    opts_xy@tmYLLabelFontHeightF = 0.015  
    opts_xy@PlotOrientation = tc_plane@Orientation
```

```
; Plotting options for RH
```

```
    opts_rh = opts_xy  
    opts_rh@pmLabelBarOrthogonalPosF = -0.07  
    opts_rh@ContourParameters = (/ 10., 90., 10. /)  
    opts_rh@cnFillOn = True  
    opts_rh@cnFillColors = (/ "White","White","White", \  
        "White","Chartreuse","Green", \  
        "Green3","Green4", \  
        "ForestGreen","PaleGreen4"/)
```

```

; Plotting options for Temperature
opts_tc = opts_xy
opts_tc@cnInfoLabelOrthogonalPosF = 0.00
opts_tc@ContourParameters = (/ 5. /)

; Get the contour info for the rh and temp
contour_tc = wrf_contour(a,wks,tc_plane,opts_tc)
contour_rh = wrf_contour(a,wks,rh_plane,opts_rh)

; MAKE PLOTS
plot = wrf_overlays(a,wks,(/contour_rh,contour_tc/),pltres)

; Delete options and fields, so we don't have carry over
delete(opts_tc)
delete(opts_rh)
delete(tc_plane)
delete(rh_plane)

end do ; make next cross section

;;;;;;;;;;;;;

end do ; END OF TIME LOOP

end

```

wrf_CrossSection2_FirstMod.ncl:

```

; Example script to produce plots for a WRF real-data run,
; with the ARW coordinate dynamics option.
; Plot data on a cross section
; This script will plot data from a a given point A to point B
; Modified to produce only one vertical plot, in the X-Z plane
; at grid y-coordinate 84
; Vertical coordinate is height

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

begin

; The WRF ARW input files.
; Use all the Jangmi typhoon files:
wrffiles = systemfunc("ls wrfout_d02_2008-09-28*")
numFiles = dimsizes(wrffiles)
do i = 0, numFiles -1
    wrffiles(i) = wrffiles(i)+".nc"
end do
inpFiles = addfiles(wrffiles,"r")

; We generate plots, but what kind do we prefer?
type = "x11"
; type = "pdf"

```

```

; type = "ps"
; type = "ncgm"
wks = gsn_open_wks(type,"plt_CrossSection2")

; Set some basic resources
res = True
res@MainTitle = "REAL-TIME WRF"

pltres = True

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

FirstTime = True
do ifile = 0, numFiles-1
a = inpFiles[ifile]
times = wrf_user_list_times(a) ; get times in the file
ntimes = dimsizes(times) ; number of times in the file

mdims = getfilevardimsizes(a,"P") ; get some dimension sizes for the file
nd = dimsizes(mdims)

;-----
; Modify time loop to loop over all times in the files:
do it = 0,ntimes-1 ; TIME LOOP

print("Working on time: " + times(it) )
res@TimeLabel = times(it) ; Set Valid time to use on plots

tc = wrf_user_getvar(a,"tc",it) ; T in C
rh = wrf_user_getvar(a,"rh",it) ; relative humidity
z = wrf_user_getvar(a, "z",it) ; grid point height

if ( FirstTime ) then ; get height info for labels
zmin = 0.
zmax = max(z)/1000.
nz = floattoint(zmax/2 + 1)
FirstTime = False
end if

;-----

; Modify loop over plots to just do one plot
ip = 1 ; Just do the one (constant y coord) plot

opts = True ; setting start and end times
plane = new(4,float)

if(ip .eq. 1) then

; Modify start and end x coordinates to match WRF D02 grid size:
; The jangmi data is on a grid from 0 to 200
plane = (/ 0,84, 200,84 /) ; start x;y & end x;y point

end if
if(ip .eq. 2) then

```

```

    plane = (/ 130,1, 130,162 /) ; start x;y & end x;y point
end if
if(ip .eq. 3) then
    plane = (/ 49,1, 210,162 /) ; start x;y & end x;y point
end if

rh_plane = wrf_user_intrp3d(rh,z,"v",plane,0.,opts)
tc_plane = wrf_user_intrp3d(tc,z,"v",plane,0.,opts)

dim = dimsizes(rh_plane) ; Find the data span - for use in labels
zspan = dim(0)

; Options for XY Plots
opts_xy = res
opts_xy@tiYAxisString = "Height (km)"
opts_xy@AspectRatio = 0.75
opts_xy@cnMissingValPerimOn = True
opts_xy@cnMissingValFillColor = 0
opts_xy@cnMissingValFillPattern = 11
opts_xy@tmYLMMode = "Explicit"
opts_xy@tmYLValues = fspan(0,zspan,nz)
; Create tick marks
opts_xy@tmYLLLabels = sprintf("%.1f",fspan(zmin,zmax,nz))
; Create labels
opts_xy@tiXAxisFontHeightF = 0.020
opts_xy@tiYAxisFontHeightF = 0.020
opts_xy@tmXBMajorLengthF = 0.02
opts_xy@tmYLMajorLengthF = 0.02
opts_xy@tmYLLLabelFontHeightF = 0.015
opts_xy@PlotOrientation = tc_plane@Orientation

; Plotting options for RH
opts_rh = opts_xy
opts_rh@pmLabelBarOrthogonalPosF = -0.07
opts_rh@ContourParameters = (/ 10., 90., 10. /)
opts_rh@cnFillOn = True
opts_rh@cnFillColors = (/ "White","White","White", \
                          "White","Chartreuse","Green", \
                          "Green3","Green4", \
                          "ForestGreen","PaleGreen4"/)

; Plotting options for Temperature
opts_tc = opts_xy
opts_tc@cnInfoLabelOrthogonalPosF = 0.00
opts_tc@ContourParameters = (/ 5. /)

; Get the contour info for the rh and temp
contour_tc = wrf_contour(a,wks,tc_plane,opts_tc)
contour_rh = wrf_contour(a,wks,rh_plane,opts_rh)

; MAKE PLOTS
plot = wrf_overlays(a,wks,(/contour_rh,contour_tc/),pltres)

```

```

; Delete options and fields, so we don't have carry over
delete(opts_tc)
delete(opts_rh)
delete(tc_plane)
delete(rh_plane)

; Comment out the loop over ip:
; end do ; make next cross section

;;;;;;;;;;;;;

end do ; END OF TIME LOOP

end do ; END OF FILE LOOP

end

```

wrf_CrossSection2_Final.ncl:

```

; Example script to produce plots for a WRF real-data run,
; with the ARW coordinate dynamics option.
; Plot data on a cross section
; This script will plot data from a a given point A to point B
; Vertical coordinate is height

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

; Load the wrf2geotiff.ncl library from VAPOR installation:
load "$VAPOR_HOME/share/examples/NCL/wrf2geotiff.ncl"

begin
;
; The WRF ARW input files.
; Make a list of all the WRF output files we are using:
wrffiles = systemfunc("ls wrfout_d02_2008-09-28*")
numFiles = dimsizes(wrffiles)
do i = 0, numFiles -1
    wrffiles(i) = wrffiles(i)+".nc"
end do
inpFiles = addfiles(wrffiles,"r")

; We generate plots, but what kind do we prefer?

; For the geotiff plots we need to use ps:
; type = "x11"
; type = "pdf"
type = "ps"
; type = "ncgm"

wks = gsn_open_wks(type,"plt_CrossSection2")

; Create the opaque pointer for wrf2geotiff:
wrf2gtiff = wrf2geotiff_open(wks)

```

```

; Since this is a vertical plot, turn off georeferencing:
wrf2geotiff_disableGeoTags(wrf2gtiff)

; Set some basic resources
res = True
res@MainTitle = "REAL-TIME WRF"

pltres = True

; Control the frame advance manually:
pltres@gsnFrame = False

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

FirstTime = True
do ifile = 0, numFiles-1
a = inpFiles[ifile]
times = wrf_user_list_times(a) ; get times in the file
ntimes = dimsizes(times) ; number of times in the file

mdims = getfilevardimsizes(a,"P") ; get some dimension sizes for the file
nd = dimsizes(mdims)

;-----

do it = 0,ntimes-1 ; TIME LOOP

print("Working on time: " + times(it) )
res@TimeLabel = times(it) ; Set Valid time to use on plots

tc = wrf_user_getvar(a,"tc",it) ; T in C
rh = wrf_user_getvar(a,"rh",it) ; relative humidity
z = wrf_user_getvar(a, "z",it) ; grid point height

if ( FirstTime ) then ; get height info for labels
zmin = 0.
zmax = max(z)/1000.
nz = floattoint(zmax/2 + 1)
FirstTime = False
end if

;-----

ip = 1 ; Just do the one (constant y coord) plot
opts = True ; setting start and end times
plane = new(4,float)

if(ip .eq. 1) then
; The jangmi data is on a grid from 0 to 200

plane = (/ 0,84, 200,84 /) ; start x;y & end x;y point
end if
if(ip .eq. 2) then
plane = (/ 130,1, 130,162 /) ; start x;y & end x;y point

```

```

end if
if(ip .eq. 3) then
  plane = (/ 49,1, 210,162 /) ; start x;y & end x;y point
end if

rh_plane = wrf_user_intrp3d(rh,z,"v",plane,0.,opts)
tc_plane = wrf_user_intrp3d(tc,z,"v",plane,0.,opts)

dim = dimsizes(rh_plane) ; Find the data span - for use in labels
zspan = dim(0)

; Options for XY Plots
opts_xy = res
opts_xy@tiYAxisString = "Height (km)"
opts_xy@AspectRatio = 0.75
opts_xy@cnMissingValPerimOn = True
opts_xy@cnMissingValFillColor = 0
opts_xy@cnMissingValFillPattern = 11
opts_xy@tmYLMMode = "Explicit"
; Create tick marks
opts_xy@tmYLValues = fspan(0,zspan,nz)
; Create labels
opts_xy@tmYLLLabels = sprintf("%.1f",fspan(zmin,zmax,nz))
opts_xy@tiXAxisFontHeightF = 0.020
opts_xy@tiYAxisFontHeightF = 0.020
opts_xy@tmXBMajorLengthF = 0.02
opts_xy@tmYLMajorLengthF = 0.02
opts_xy@tmYLLLabelFontHeightF = 0.015
opts_xy@PlotOrientation = tc_plane@Orientation

; Plotting options for RH
opts_rh = opts_xy
opts_rh@pmLabelBarOrthogonalPosF = -0.07
opts_rh@ContourParameters = (/ 10., 90., 10. /)
opts_rh@cnFillOn = True
opts_rh@cnFillColors = (/ "White","White","White", \
                          "White","Chartreuse","Green", \
                          "Green3","Green4", \
                          "ForestGreen","PaleGreen4"/)

; Plotting options for Temperature
opts_tc = opts_xy
opts_tc@cnInfoLabelOrthogonalPosF = 0.00
opts_tc@ContourParameters = (/ 5. /)

; Get the contour info for the rh and temp
contour_tc = wrf_contour(a,wks,tc_plane,opts_tc)
contour_rh = wrf_contour(a,wks,rh_plane,opts_rh)

; MAKE PLOTS
plot = wrf_overlays(a,wks,(/contour_rh,contour_tc/),pltres)

; save the info for the geotiff, and end the frame.

```



```

; Do crop to bounds
wrf2geotiff_write(wrf2gtiff, a, times(it), wks, plot, True)
frame(wks)

; Delete options and fields, so we don't have carry over
delete(opts_tc)
delete(opts_rh)
delete(tc_plane)
delete(rh_plane)

; end do ; make next cross section (loop over ip was removed)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
end do ; END OF TIME LOOP

end do ; END OF FILE LOOP

; Close wrf2geotiff:
wrf2geotiff_close(wrf2gtiff,wks)

end

```

