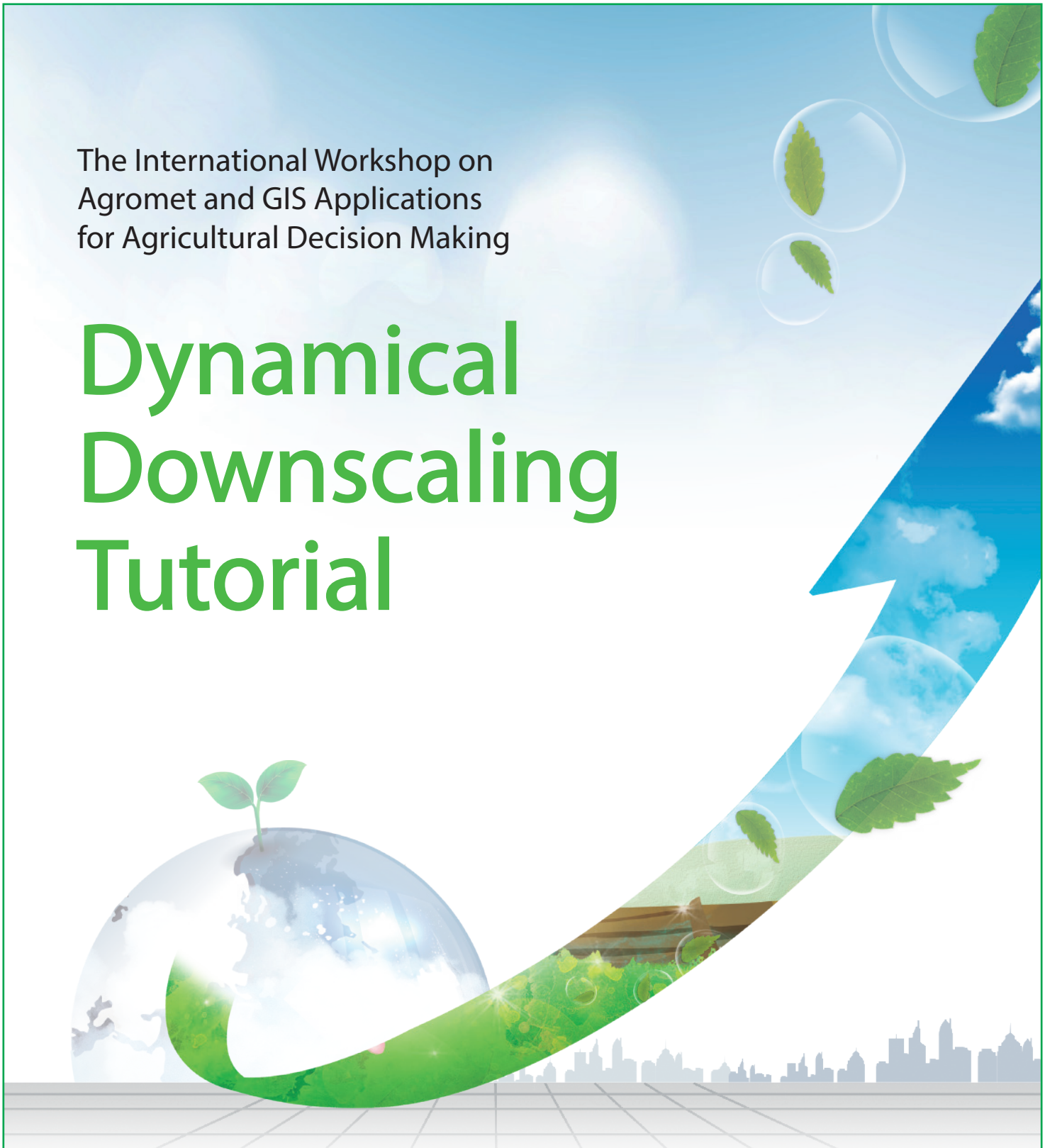
The International Workshop on
Agromet and GIS Applications
for Agricultural Decision Making

Dynamical Downscaling Tutorial



Date : December 5(Mon)~9(Fri), 2016
Place : MSTAY Hotel JEJU
Hosted by : Korea Meteorological Administration(KMA)
Organized by : National Institute of Meteorological Sciences(NIMS)
Sponsored by : WMO CAgM / NCAM / APCC / OSGeo / PKNU / DU

Korea Meteorological Administration

National Institute of Meteorological Sciences

contents

	Dynamical Downscaling Tutorial Organized by Dr. OH, Jaiho
Instructor	Dr. YANG, Shin Il and his assistants OH Jiwon, KIM Gaeun and CHOI Kyoungmin Super-Computing Center, Pukyong National University Department of Environmental Atmospheric Sciences, Pukyong National University Email : soho0427@gmail.com
Who is for	Anyone who is interested in downscaling methods from beginner to Intermediate level. This will be a good starting point for those who want to use downscaling methods in their own fields.
Prerequisite	The following items must be brought to the tutorial session : 1. His/her own notebook computer
Contents	Dynamical downscaling methods used by WRF 1. Introduction to WRF-ARW (80 min.) - WRF Preprocessing System (WPS) - WRF Dynamics and Physics 2. Hands on training I (80 min.) - Introduction to PKNU cluster for numerical practice - Installation WPS and WRF 3. Hands on training II (100 min.) - Running WPS and WRF with agricultural meteorological examples 4. Hands on training III (100 min.) - Practice post-processing utilities - Q & A
Remarks	The contents may be subject to change without notification.

Introduction to QGIS

- Using QGIS and ISCGM Global Map -

Introduction of WRF-ARW

December 8, 2016

Sin-Il Yang, Ji-Won Oh, Kyung-Min Choi, and Ga-Eun Kim
Pukyong National University, Busan, Korea

Scope of Tutorial

- ➔ **What's in the modeling system**
  - **– Pre-processing programs and model**
- ➔ **How to install and run the modeling system**
- ➔ **Hints on choosing options**
- ➔ **Advanced usage**
  - **– Adding your own input or output data**
  - **– Post-processing such as plotting tool**

What is WRF?

➔ **WRF: Weather Research and Forecasting Model**

– Used for both research and operational forecasting

➔ **It is a supported "community model"**

– a free and shared resource ;

– Distributed development and centralized support ;

➔ **Its development is led by NCAR, NOAA/ESRL and NOAA/NCEP/EMC with partnerships at AFWA, FAA, DOE/PNNL and collaborations with universities**

WRF Community Model

➔ **Version 1.0 WRF was released December 2000**

➔ **Version 2.0: May 2004 (add nesting)**

➔ **Version 3.0: April 2008 (add global ARW version)**

… (major releases in April, minor releases in summer)

➔ **Version 3.7: April 2015**

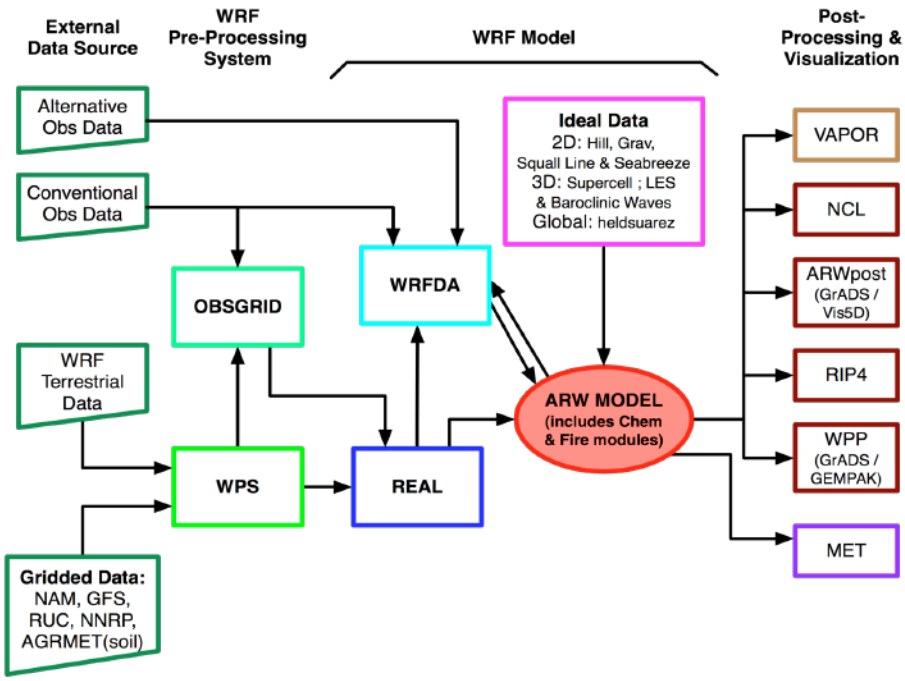
➔ **Version 3.7.1: August 2015**

➔ **Version 3.8: April 2016 (current version)**

What can WRF be used for?

- ➔ Atmospheric physics/parameterization research
- ➔ Case-study research
- ➔ Real-time NWP and forecast system research
- ➔ Data assimilation research
- ➔ Teaching dynamics and NWP
- ➔ Regional climate and seasonal time-scale research
- ➔ Coupled-chemistry applications
- ➔ Global simulations

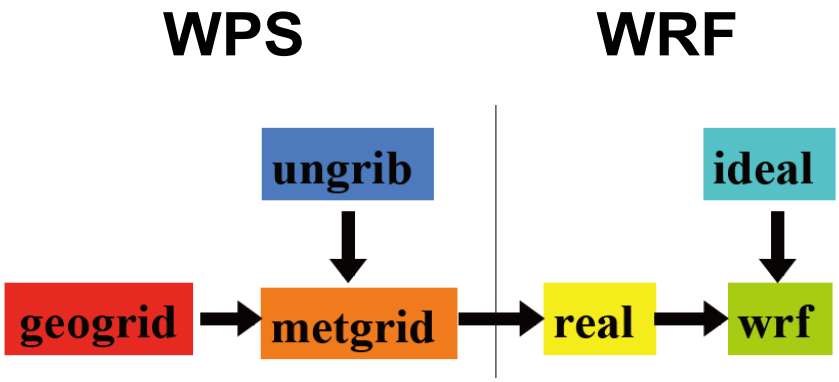
WRF Modeling System Flow Chart



Modeling System Components

- **WRF Pre-processing System**
  - Real-data interpolation for NWP runs (WPS)
  - Program for adding more observations to analysis (obsgrid)
- **WRF Model**
  - Initialization programs for real and (for ARW) idealized data (real.exe/ideal.exe)
  - Numerical integration program (wrf.exe)
- **Graphics and verification tools including MET**
- **WRFDA (separate tutorial)**
- **WRF-Chem (separate tutorial)**
- **WRF-Hydro – hydrology model coupled to WRF**
- **WRF-Fire – wildland model for forest fires**

WPS and WRF Program Flow



Real-Data Applications

- ➔ Numerical weather prediction
- ➔ Meteorological case studies
- ➔ Regional climate
- ➔ Applications: air quality, wind energy, hydrology, etc.

Real-Data Applications

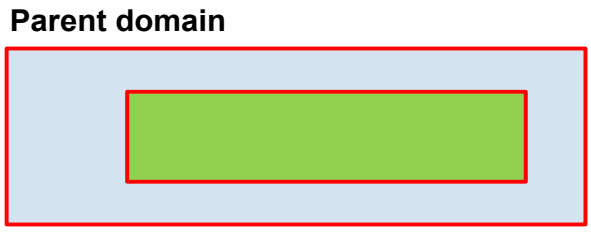
- ➔ Pre-processing for regional domains therefore needs multiple times for lateral boundary conditions during whole forecast period
  - – Note: Global models only need initial analysis
  - – Real-time regional NWP often uses global forecast for boundary conditions
- ➔ Long simulations also need lower boundary information on SST and sea ice to update them over periods of weeks, months, years

Nesting

➔ Running multiple domains with increasing resolution in nested areas

➔ Parent has specified boundary conditions from wrfbdy file

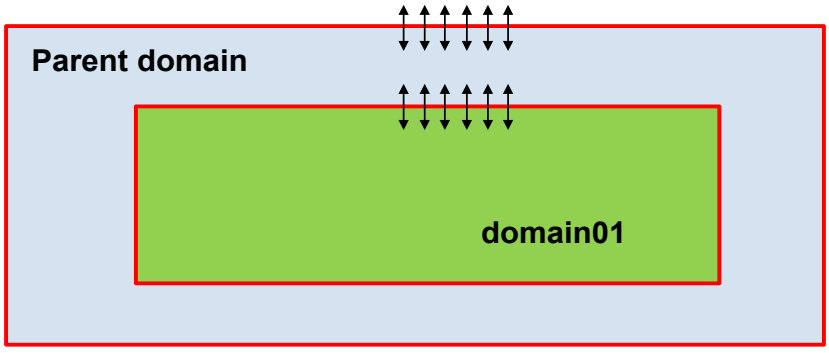
➔ Nested boundary conditions come from parent



Nesting : Two-Way nesting

➔ Lateral boundary condition is provided by parent domain at every parent step

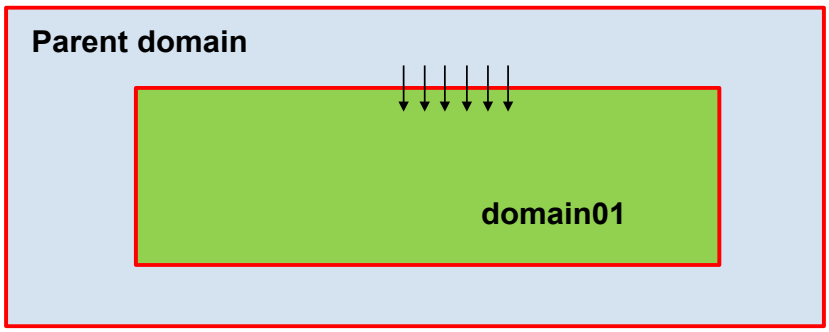
➔ Feedback: Interior of nest overwrites overlapped parent area



Nesting : One-Way nesting

➔ **As two-way nesting but no feedback**

– **Uses parent WRF run instead of analysis for initial and lateral boundary conditions**



WRF Preprocessing System (WPS)

➔ **Define Simulation domain area (and nests)**

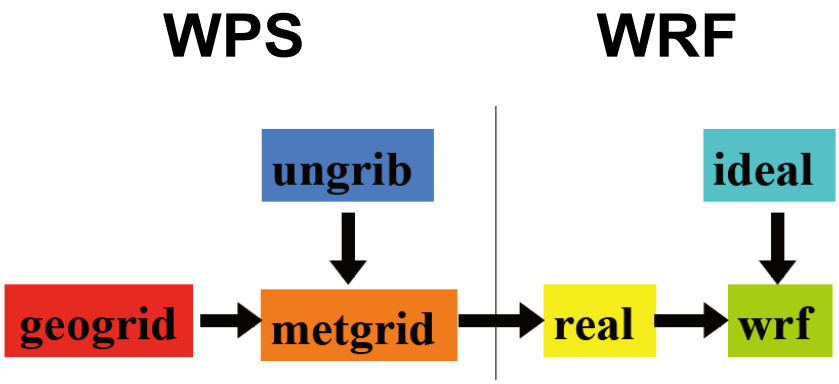
➔ **Produce terrain, land-use, soil type etc. on the simulation domain ("static" fields)**

➔ **De-grib GRIB files for meteorological data (u, v, T, q, surface pressure, soil data, snow data, SST, etc.)**

➔ **Interpolate meteorological data to WRF model grid (horizontally)**

➔ **Optionally add more observations to analysis (separate obsgrid program)**

WPS and WRF Program Flow



WRF: Key features

- ➔ 3rd-order Runge-Kutta time integration scheme
- ➔ High-order advection scheme
- ➔ Scalar-conserving (positive definite option)
- ➔ Complete Coriolis, curvature and mapping terms
- ➔ Two-way and one-way nesting
- ➔ Full physics options to represent atmospheric radiation, surface and boundary layer, and cloud and precipitation processes
- ➔ Grid-nudging and obs-nudging (FDDA)

Examples of WRF Forecasts

(1) A sharp drop in temperature case (October, 2016)

- ➔ **Nesting Korean peninsula domain (9km-3km-1km)**
- ➔ **Using GME 10day simulation (20km)**

Question?

or

Coffee Break

AM 10:20 ~ 10:40

Hands on training (1)

Hands on training (1)

December 8, 2016

Sin-Il Yang, Ji-Won Oh, Kyung-Min Choi, and Ga-Eun Kim
Pukyong National University, Busan, Korea

Introduction of PKNU cluster

➔ PKNU cluster for numerical practice

Name	6 nodes 'gomsol' cluster		
Structure	Master : 1EA	Client : 5EA	Storage : 0EA
CPU	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz / 2 Slot		
Memory / Processor	32 GB / (16+16) Core		

Programs

PuTTY

Free SSH, Telnet and Rlogin for Windows System

http://www.chiark.greenend.org.uk/~sgtatham/putty/

※ NOTICE ※

PuTTY does not provide graphic visualization, you should download and install Xming.

Xming

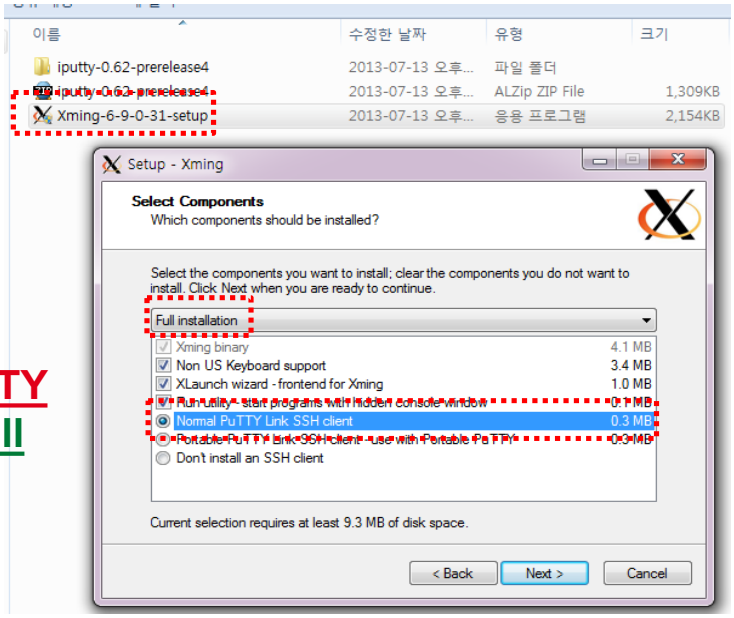
Xming is the leading X Windows System Server for Microsoft Windows.
You can display remote X clients directly on the Windows Desktop.

http://www.straightrunning.com/XmingNotes/

Installing Xming (1/2)

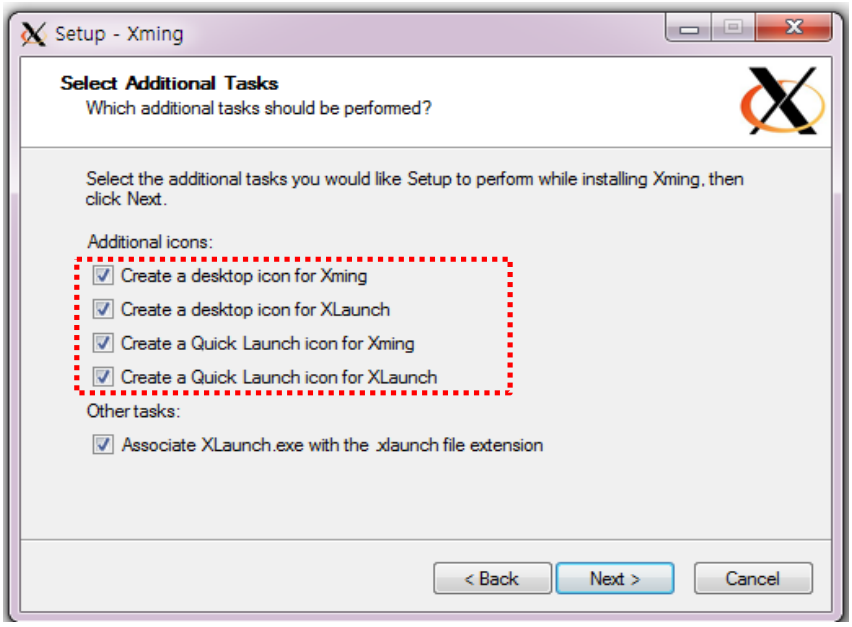
1) Run 'Xming-6-9-0-31-setup' to install Xming

2) Select 'Normal PuTTY Link SSH client' in 'Full installation'



Installing Xming (1/2)

3) Check all boxes for additional icons

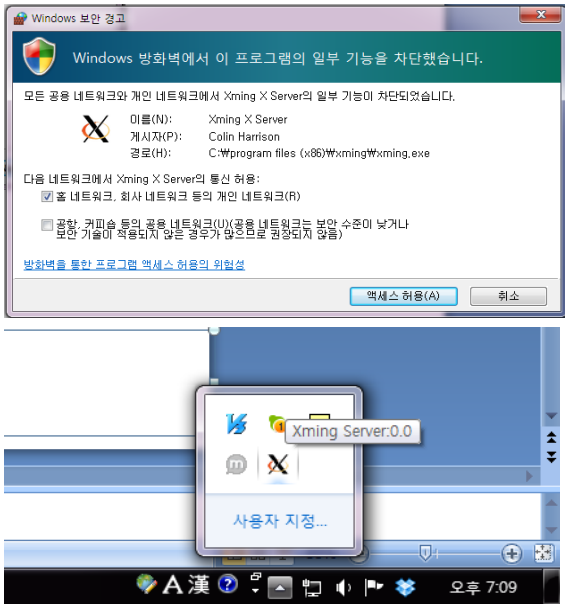


Run Xming

4) Run Xming

- Click ‘ **allow access (A)**’

If Xming icon does not appear, run Xming first.



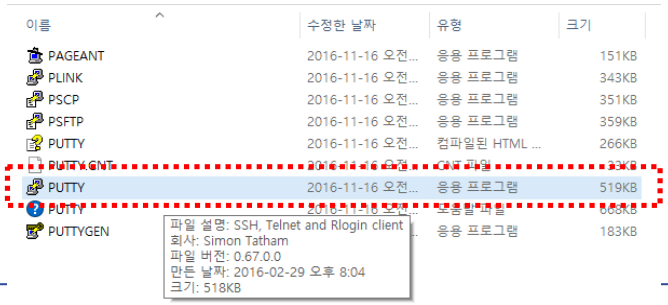
Set up and Run PuTTY (1/4)

Download:
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

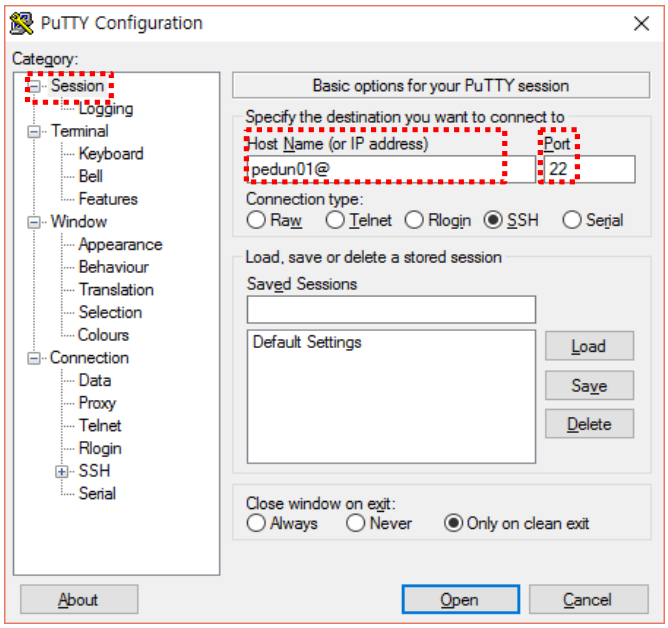
1) Download a .ZIP file containing all the binaries

1) Extract and run PuTTY

- Extract a .ZIP file
- No need to install



Set up and Run PuTTY (2/4)



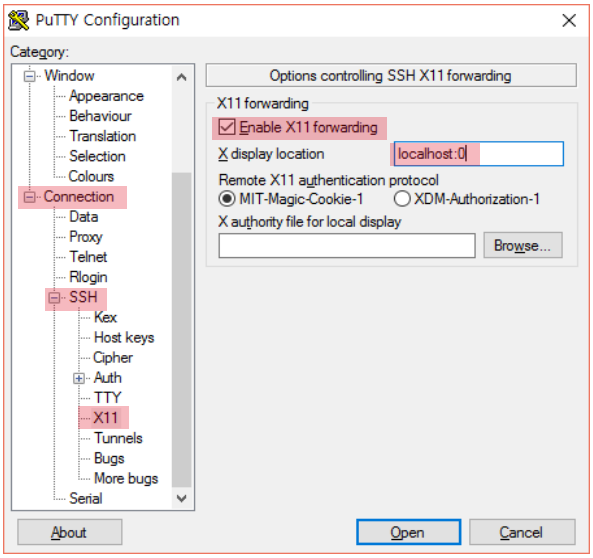
2) Type Host Name (or IP address)

- ex) pedun01@[your IP]

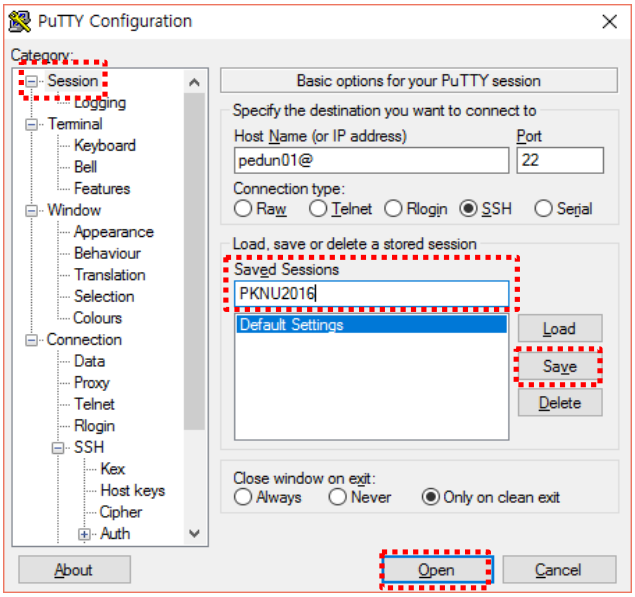
Set up and Run PuTTY (3/4)

3) Connect Xming & PuTTY

- Move to ‘Connection > SSH > X11’ menu in PuTTY configuration
- Check ‘Enable X11 forwarding’
- Type ‘localhost:0’ in X display location menu



Set up and Run PuTTY (4/4)



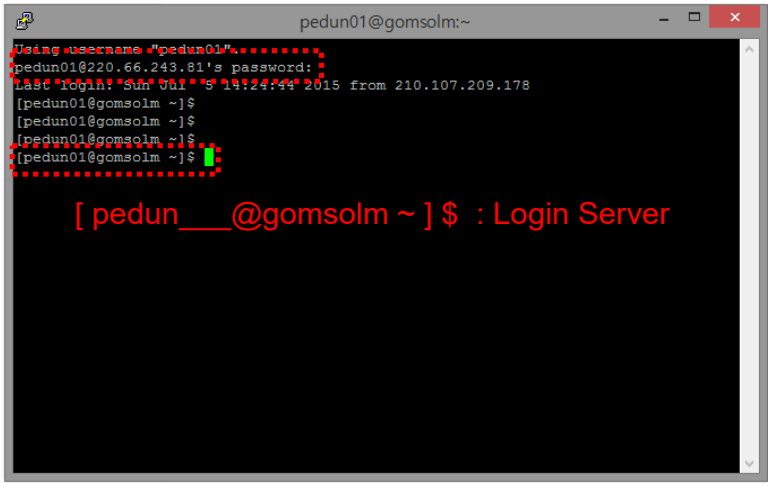
4) Save Configuration

- Move to ‘Session’ menu
- Enter ‘PKNU2016’ in ‘Saved Sessions’ to save the host configuration
- Click ‘Save’ and ‘Open’

Connect to gomsol cluster

1) Connect to gomsol-master

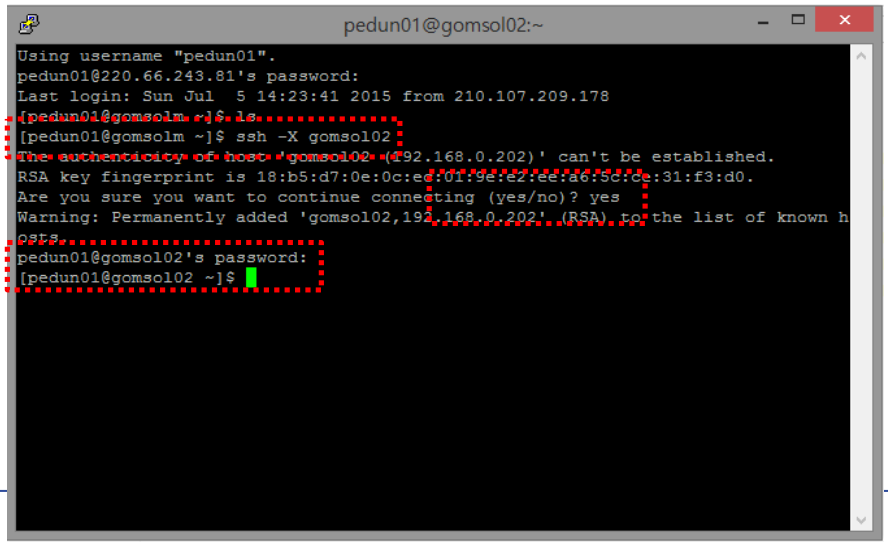
- ID for tutorial : pedun01 ~ pedun30
- Password : cagm2016!@#



Connect to gomsol cluster

2) Move to gomsol01~gomsol05

- Move to practical server gomsol01~gomsol05
- **ssh –X gomsol[01~05]**
- ex) ssh –X gomsol02



Installing Steps

- ➔ 1. Check system requirements
- ➔ 2. Installing libraries
- ➔ 3. Download source data
- ➔ 4. Compile WRFV3
- ➔ 5. Compile WPS
- ➔ 6. Download initial/boundary condition data

Check System Requirements

- ➔ On what kinds of systems will WRF run?
  - Generally any 32- or 64-bit hardware, running a UNIX-like OS
  - You may also use dual-booting into a UNIX-like OS
    (e.g., Windows with Linux built parallel)

- ➔ Examples of acceptable systems:
  - Laptops, desktops, and clusters running Linux
  - Laptops and desktops running MacOS X
  - Clusters running Unix-like: Linux, AIX

Installing Libraries

➔ NetCDF (needed by WRF and WPS)

➔ Optional libraries for GRIB2 meteorological data support

- JasPer (JPEG 2000 “lossy” compression library)
- PNG (“lossless” compression library)
- Zlib (compression library used by PNG)

➔ Optional MPI library (for building in parallel):

- MPICH2

Basic Software Requirement

➔ Fortran 90/95 compiler

- code uses standard f90 (very portable)

➔ C compiler

- automatic Fortran code generation

(for argument lists, declarations, nesting functions, I/O routines)

➔ netcdf library

- for I/O (other I/O formats semi-supported)

➔ Public domain mpich for MPI

- if using distributed memory option

Download source data: WRF & WPS Code

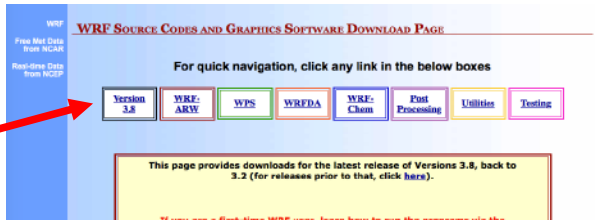
➔ **Download WRF & WPS source code from:**

http://www2.mmm.ucar.edu/wrf/users/download/get_source.html

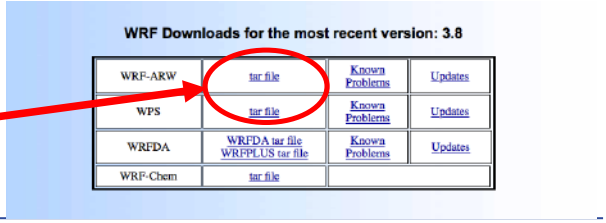
– **Click 'New User,' register and download, or**

– **Click 'Returning User,' enter your email, and download**

Step 1:
Click here for the latest released code (recommended)



Step 2:
Click on tar files to download

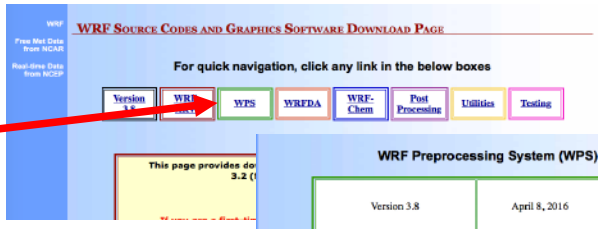


Download source data: Geographical Data (1/2)

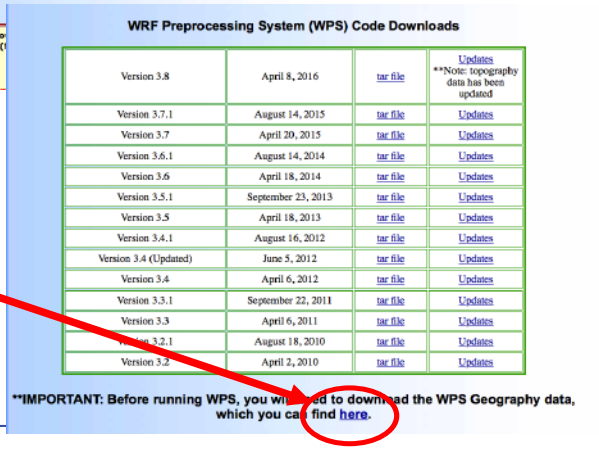
➔ **From the WRF Download page:**

http://www2.mmm.ucar.edu/wrf/users/download/get_source.html

Step 1:
Click 'WPS' box



Step 2:
Click 'here'
to get geography data



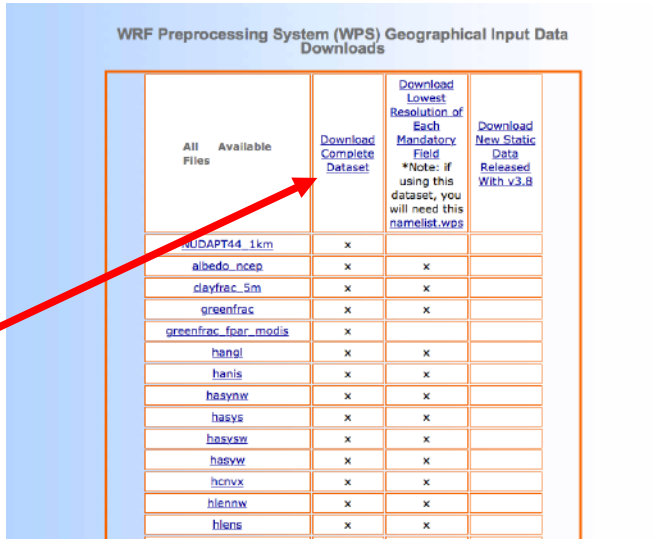
Download source data: Geographical Data (2/2)

➔ **Geographical Input and Data Download Page:**

http://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html

geog.tar.gz (~15 GB) when uncompressed

This is the one You want



WRF Preprocessing System (WPS) Geographical Input Data Downloads

All Available Files	Download Complete Dataset	Download Lowest Resolution of Each Mandatory Field *Note: if using this dataset, you will need this namelist.wps	Download New Static Data Released With v3.8
NUDAPT44_1km	x		
albedo_ncep	x	x	
clayfrac_5m	x	x	
greenfrac	x	x	
greenfrac_fpar_modis	x		
hangl	x	x	
hanis	x	x	
hasynw	x	x	
hasys	x	x	
hasysw	x	x	
hasyw	x	x	
hcnvx	x	x	
hlennw	x	x	
hlens	x	x	

Configure for WRFV3

➔ **Inside the WRFV3/ directory, type: ./configure**

```
checking for perl5... no
checking for perl... found /usr/bin/perl (perl)
Will use NETCDF in dir: /glade/apps/opt/netcdf/4.3.0/intel/12.1.5
PHDF5 not set in environment. Will configure WRF for use without.
Will use 'time' to report timing information
$JASPERLIB or $JASPERINC not found in environment, configuring to build without grib2 I/O...
------------------------------------------------------------------------
Please select from among the following Linux x86_64 options:

  1. (serial)   2. (smpar)   3. (dmpar)   4. (dm+sm)   PGI (pgf90/gcc)
  5. (serial)   6. (smpar)   7. (dmpar)   8. (dm+sm)   PGI (pgf90/pgcc): SGI MPT
  9. (serial)  10. (smpar)  11. (dmpar)  12. (dm+sm)   PGI (pgf90/gcc): PGI accelerator
 13. (serial)  14. (smpar)  15. (dmpar)  16. (dm+sm)   INTEL (ifort/icc)
                                         17. (dm+sm)   INTEL (ifort/icc): Xeon Phi (MIC architecture)
 18. (serial)  19. (smpar)  20. (dmpar)  21. (dm+sm)   INTEL (ifort/icc): Xeon (SNB with AVX mods)
 22. (serial)  23. (smpar)  24. (dmpar)  25. (dm+sm)   INTEL (ifort/icc): SGI MPT
 26. (serial)  27. (smpar)  28. (dmpar)  29. (dm+sm)   INTEL (ifort/icc): IBM POE
 30. (serial)               31. (dmpar)                PATHSCALE (pathf90/pathcc)
 32. (serial)  33. (smpar)  34. (dmpar)  35. (dm+sm)   GNU (gfortran/gcc)
 36. (serial)  37. (smpar)  38. (dmpar)  39. (dm+sm)   IBM (xlf90_r/cc_r)
 40. (serial)  41. (smpar)  42. (dmpar)  43. (dm+sm)   PGI (ftn/gcc): Cray XC CLE
 44. (serial)  45. (smpar)  46. (dmpar)  47. (dm+sm)   CRAY CCE (ftn/gcc): Cray XE and XC
 48. (serial)  49. (smpar)  50. (dmpar)  51. (dm+sm)   INTEL (ftn/icc): Cray XC
 52. (serial)  53. (smpar)  54. (dmpar)  55. (dm+sm)   PGI (pgf90/pgcc)
 56. (serial)  57. (smpar)  58. (dmpar)  59. (dm+sm)   PGI (pgf90/gcc): -f90=pgf90
 60. (serial)  61. (smpar)  62. (dmpar)  63. (dm+sm)   PGI (pgf90/pgcc): -f90=pgf90

Enter selection [1-63] :


Compile for nesting? (0=no nesting, 1=basic, 2=preset moves, 3=vortex following) [default 0]:
```

➔ **Output from configuration: a file called 'configure.wrf'**

Parallel Compile Option for WRFV3

➔ **To build WRF in parallel:**

– **setenv J "-j 2" for csh or export J="-j 2" for bash/sh**

# of Processors	Time to Compiler
1	22.8 Mins
2	14.92 Mins
3	9.33 Mins
4	**8.02 Mins**
5	7.23 Mins
6	6.68 Mins

*** Around 4 processors, it reaches state of equilibrium**

*** This test done with GNU compiler**

Compile WRFV3

➔ **In the WRFV3/ directory, type:**

./compile em_case >& log.compile

Important in case there are compile problems

Where **em_case** is one of the following

(type **./compile** to see all options)

3d Ideal

- em_real (3d real case)
- em_quarter_ss
- em_b_wave
- em_les
- em_heldsuarez
- em_tropical_cyclone
- em_convrad

2d Ideal

- em_hill2d_x
- em_squall2d_x
- em_squall2d_y
- em_grav2d_x
- em_seabreeze2d_x

em_scm_xy (1d ideal)

** Compilation should take at most ~30 mins **

Successful Compilation

➔ **If the compilation is successful,**

you should find these executables in WRFV3/main:

Real data case:

wrf.exe – model executable

real.exe – real data initialization

ndown.exe – one-way nesting

tc.exe – for TC bogusing (serial only)

Unsuccessful Compilation

➔ **Use your 'log.compile' file to search for errors!**

– Search for 'Error' with a capital 'E'

➔ **Before recompiling:**

– issue a 'clean –a'

➔ **Contact wrfhelp@ucar.edu**

Configure for WPS

➔ Inside the WPS/ directory, type: **./configure**

```
Will use NETCDF in dir: /glade/apps/opt/netcdf/4.3.0/intel/12.1.5
$JASPERLIB or $JASPERINC not found in environment. Using default values for library paths...
------------------------------------------------------------------------
Please select from among the following supported platforms.

   1.  Linux x86_64, gfortran    (serial)
   2.  Linux x86_64, gfortran    (serial_NO_GRIB2)
   3.  Linux x86_64, gfortran    (dmpar)
   4.  Linux x86_64, gfortran    (dmpar_NO_GRIB2)
   5.  Linux x86_64, PGI compiler   (serial)
   6.  Linux x86_64, PGI compiler   (serial_NO_GRIB2)
   7.  Linux x86_64, PGI compiler   (dmpar)
   8.  Linux x86_64, PGI compiler   (dmpar_NO_GRIB2)
```

- Choose to compile WPS **serially**, even if you compile WRFV3 in parallel (unless you have a very large domain)
  **NOTE: if you do compile WPS in parallel, ungrib.exe must run serially
- Output from configuration: a file called 'configure.wps'

Compile WPS

➔ In the WPS/ directory, type: **./compile >& log.compile**

➔ Compilation should only take a few minutes

➔ If successful, these executables should be in your WPS/ directory (and they are linked, respectively, from their source code directories):

geogrid.exe -> geogrid/src/geogrid.exe
ungrib.exe -> ungrib/src/ungrib.exe
metgrid.exe -> metgrid/src/metgrid.exe

Unsuccessful WPS Compilation (1/2)

➔ If no geogrid.exe or metgrid.exe?

– The external I/O libraries in the WRFV3/external/directory

– Compiler (and version) as you used to compile WRFV3

– The netCDF library that you used to build WRFV3

– Name or path of the WRFV3/ directory

Unsuccessful WPS Compilation (2/2)

➔ No ungrib.exe

– Make sure you have installed your jasper, zlib, and libpng libraries correctly.

– Make sure that you are using the correct path and format for the following lines in the configure.wps file:

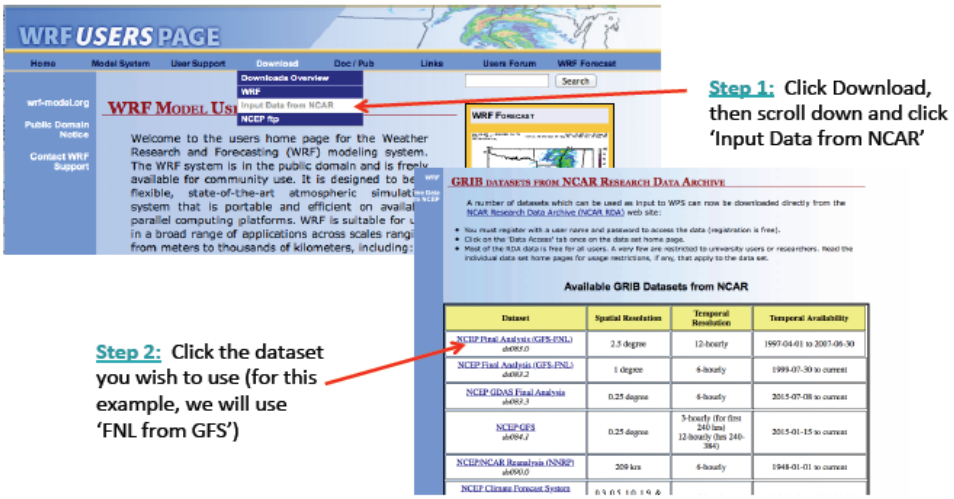
```
COMPRESSION_LIBS = -L/${DIR}/UNGRIB_LIBRARIES/lib –ljasper –lpng –lz
COMPRESSION_INC = -I/${DIR}/UNGRIB_LIBRARIES/include
```

→ Save configure.wps and recompile.

Download Datasets (1/4)

➔ From the WRF Users' page:

http://www2.mmm.ucar.edu/wrf/users/



Step 1: Click Download, then scroll down and click 'Input Data from NCAR'

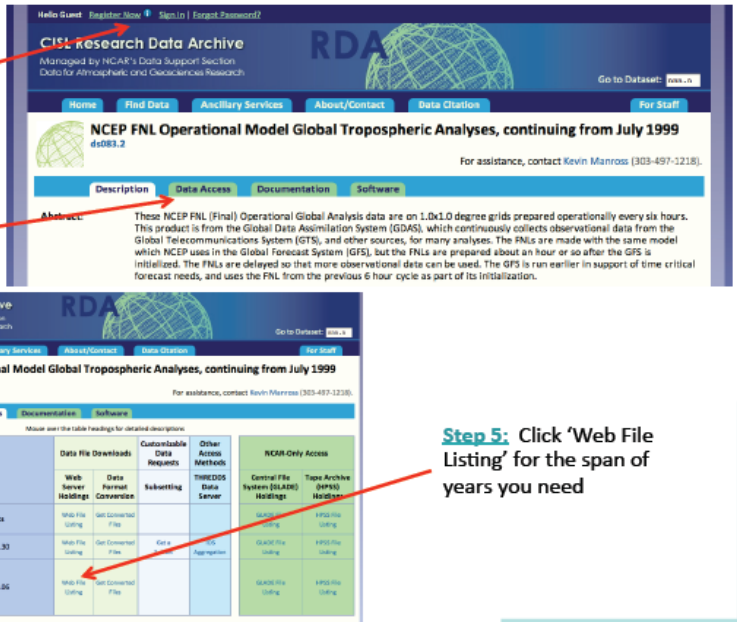
Step 2: Click the dataset you wish to use (for this example, we will use 'FNL from GFS')

*Note: NOMADS site has several types of useful data: http://nomads.ncdc.noaa.gov

Download Datasets (2/4)

Step 3: Register, or sign in, if you already have an account

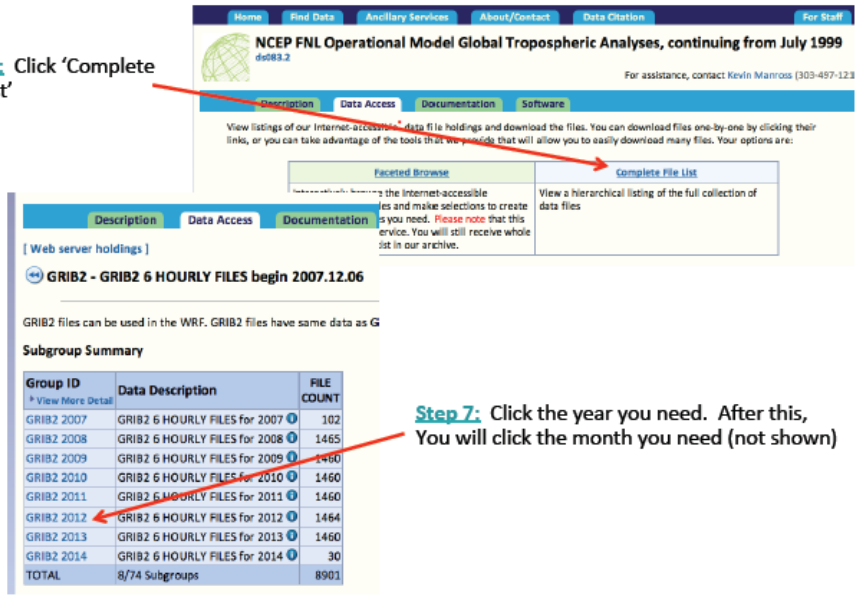
Step 4: Click 'Data Access'



Step 5: Click 'Web File Listing' for the span of years you need

Download Datasets (3/4)

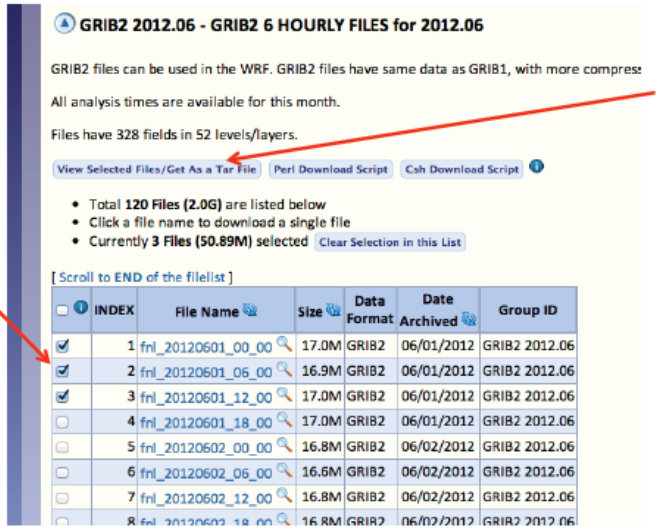
Step 6: Click 'Complete File List'



Step 7: Click the year you need. After this, You will click the month you need (not shown)

Download Datasets (4/4)

Step 8: Click a box for each time span that you need



Step 9: Once you have chosen All your times, click on the 'View Selected Files/Get As a Tar File' button To download one tar file with all your Dates/times

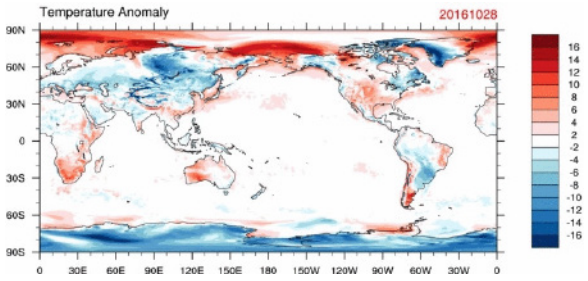
Initial and Boundary Data

➔ GME 10day forecast data

– Initial date: 2016.10.28 00 UTC

– Time interval: 1 hour

– Spatial resolution: 20km



➔ Introduction of GME

– Operational global numerical weather prediction model of the German Weather Service (DWD, Deutscher Wetterdienst)

– The model is on almost uniform icosahedral-hexagonal grid

– The GME gridpoint approach avoids the disadvantages of spectral techniques as well as the pole problem in latitude–longitude grids (singularity at the poles) and provides a data structure well suited to high efficiency on distributed memory parallel computers.

How to Make Intermediate File

➔ Vtable are not provided for GME

– We should make intermediate file by self instead of run ungrib.exe with linking Vtable.

1) Set up environments for processing (date, path, etc.)

2) Extract 2D and 3D variables from GME output using wgrib and NCL

3) Make intermediate files

Setting EXP_SET.sh (1/2)

➔ Edit EXP_SET.sh

```
#=========================================================
#                1. Set the GME date
#=========================================================
cnt_yy=`date '+%Y-%m-%d' |cut -c1-4` ;export cnt_yy
cnt_mm=`date '+%Y-%m-%d' |cut -c6-7` ;export cnt_mm
cnt_dd=`date '+%Y-%m-%d' |cut -c9-10` ;export cnt_dd

day3_pre_yy=2016
day3_pre_mm=10
day3_pre_dd=28

export day3_pre_yy
export day3_pre_mm
export day3_pre_dd

day3_pre_date=${day3_pre_yy}${day3_pre_mm}${day3_pre_dd}
export day3_pre_date

#=========================================================
# day3_pre_num_mm = day3 previous month for wgrib (number)
# day3_pre_num_dd = day3 previous day for wgrib (number)
# check_num_mm = if ten digits of month is 0
# check_num_dd = if ten digits of day is 0

check_num_mm=`echo $day3_pre_mm |cut -c1-1`
check_num_dd=`echo $day3_pre_dd |cut -c1-1`

export check_num_mm
export check_num_dd

start_ihh=00
export start_ihh

num_of_gme_output=24
export num_of_gme_output
#=========================================================
```

Initial date
- day3_pre_yy: year (2016)
- day3_pre_mm: month (10)
- day3_pre_dd: day (28)

Initial time(UTC) and number of 10day outputs
- start_ihh: initial time (00)
- num_of_gme_output: from 00 to 23 UTC (24)

Setting EXP_SET.sh (2/2)

➔ Edit EXP_SET.sh

```
#=================================================================
#
#       gme_rsl = Horizontal Resolution of GME (e.g., i384 or i192)
#       gme_out_dir = Directory Path of GME Output
#       gme_tint = Time Interval of GME
#       gme_nt = The Total Number of GME Output Files
#       gme_end_d = The Number of End Day of GME (e.g., If 10days, then 10)

gme_out_dir=/dadama01/jwoh/CAgM_tutorial/10DAY_OUT/${day3_pre_date}00_ni384_r
export gme_out_dir
main_dir=/home01/jwoh/WRFV3.7/WPS/conv_gme_for_wrf
export main_dir

gme_tint=1
gme_tint_real=1.0
gme_end_d=1
export gme_tint
export gme_tint_real
export gme_end_d
```

Time interval of GME
- gme_tint: 1hour interval (1)
- gme_end_d: 1day (1)

➔ source EXP_SET.sh

➔ If you don't have error message move on next step.

Extract the Variables from GME output (1/4)

➔ 1) Extract the variables from GME output using WGRIB in GME-WRF_WGRIB_v1.sh

```
# 2D variables
wgrib -s $gme_out_dir/gfrf${dd}${tt}0000 | egrep "(PS|:T:s|T:2|:U:10|:V:10|:QV:2|:FI:|:FR_LAND:|:FR_ICE:|:H_SNOW:|:W_SNOW:)" | wgrib -s $gme_out_dir/gfrf${dd}${tt}0000 -i -ieee -o ${main_dir}/intdata/${day3_pre_date}00/gme_sfc.${yy}${mm}${dday}${ttime}

# 3D variables
wgrib -s $gme_out_dir/gfrf${dd}${tt}0000p | egrep "(FI:|T:|:V:|U:|PMSL:|RELHUM:)" | wgrib -s $gme_out_dir/gfrf${dd}${tt}0000p -i -ieee -o ${main_dir}/intdata/${day3_pre_date}00/gme_plv.${yy}${mm}${dday}${ttime}

# 3D soil temperature (9 levels)
wgrib -s $gme_out_dir/gfrf${dd}${tt}0000 | egrep "(:T_SO:)" | wgrib -s $gme_out_dir/gfrf${dd}${tt}0000 -i -grib -o ${main_dir}/intdata/${day3_pre_date}00/TSO_${yy}${mm}${dday}${ttime}.grb


-grib -o ${main_dir}/intdata/${day3_pre_date}00/WSO_${yy}${mm}${dday}${ttime}.grb
```

Extract the Variables from GME output (1/4)

➔ 2D&3D variables are extracted to make intermediate files:

2D Variables (11)	3D Variables (6)
surface pressure	geopotential
surface temperature	temperature
temperature above 2m ground	zonal wind
zonal wind above 10m ground	meridional wind
meridional wind above 10m ground	mean sea level pressure
specific humidity above 2m ground	Relative humidity
geopotential at half levels	
land fraction of surface	
ice fraction for ocean or lake surface	
snow depth	
water content of snow	

Extract the Variables from GME output (2/4)

➔ 2) Vertical interpolation from GME soil levels to four Soil levels (0-10,10-40,40-100,100-200 cm) with NCL program in GME-WRF_WGRIB_v1.sh

```
begin
; ---------------------------------------------------------------------
 apath = systemfunc ("ls ${main_dir}/intdata/${day3_pre_date}00/TSO_${yy}${mm}${dday}${ttime}.grb")
 a = addfiles(apath,"r")
 tso = a[:]->SO_TEMP_GDS0_DBLL(:,:,:)
 printVarSummary(tso)
 lev = a[:]->lv_DBLL0

 lev1 = (/5.,25.,70.,150./) ; same compared to GFS Vtable (unit : cm, in case of t_so, don't need for unit conversion
 tso1 = int2p_n_Wrap (lev,tso,lev1,1,0)

 system("rm -rf ${main_dir}/intdata/${day3_pre_date}00/TSO_${yy}${mm}${dday}${ttime}.ieee")
 fbinrecwrite ("${main_dir}/intdata/${day3_pre_date}00/TSO_${yy}${mm}${dday}${ttime}.ieee", -1, tso1(0,:,:) )
 fbinrecwrite ("${main_dir}/intdata/${day3_pre_date}00/TSO_${yy}${mm}${dday}${ttime}.ieee", -1, tso1(1,:,:) )
 fbinrecwrite ("${main_dir}/intdata/${day3_pre_date}00/TSO_${yy}${mm}${dday}${ttime}.ieee", -1, tso1(2,:,:) )
 fbinrecwrite ("${main_dir}/intdata/${day3_pre_date}00/TSO_${yy}${mm}${dday}${ttime}.ieee", -1, tso1(3,:,:) )
 … … …
```

Extract the Variables from GME output (3/4)

➔ Run the shell script :

./ GME-WRF_WGRIB_V1.20160217.sh

```
1:0:d=16102800:FI:sfc:23hr fcst:NAve=0
3:6488076:d=16102800:FR_LAND:sfc:23hr fcst:NAve=0
4:9732114:d=16102800:PS:sfc:23hr fcst:NAve=0
5:12976152:d=16102800:T:sfc:23hr fcst:NAve=0
7:19464228:d=16102800:W_SNOW:sfc:23hr fcst:NAve=0
10:29196342:d=16102800:T:2 m above gnd:23hr fcst:NAve=0
12:35684418:d=16102800:QV:2 m above gnd:23hr fcst:NAve=0
15:45416532:d=16102800:U:10 m above gnd:23hr fcst:NAve=0
16:48660570:d=16102800:V:10 m above gnd:23hr fcst:NAve=0
43:136249596:d=16102800:FR_ICE:sfc:23hr fcst:NAve=0
61:194642280:d=16102800:H_SNOW:sfc:23hr fcst:NAve=0
1:0:d=16102800:PMSL:MSL:23hr fcst:NAve=0
2:3243710:d=16102800:U:1000 mb:23hr fcst:NAve=0
3:6487420:d=16102800:V:1000 mb:23hr fcst:NAve=0
4:9731130:d=16102800:U:925 mb:23hr fcst:NAve=0
5:12974840:d=16102800:V:925 mb:23hr fcst:NAve=0
6:16218550:d=16102800:U:850 mb:23hr fcst:NAve=0
7:19462260:d=16102800:V:850 mb:23hr fcst:NAve=0
8:22705970:d=16102800:U:700 mb:23hr fcst:NAve=0
9:25949680:d=16102800:V:700 mb:23hr fcst:NAve=0
10:29193390:d=16102800:U:600 mb:23hr fcst:NAve=0
11:32437100:d=16102800:V:600 mb:23hr fcst:NAve=0
12:35680810:d=16102800:U:500 mb:23hr fcst:NAve=0
13:38924520:d=16102800:V:500 mb:23hr fcst:NAve=0
14:42168230:d=16102800:U:400 mb:23hr fcst:NAve=0
15:45411940:d=16102800:V:400 mb:23hr fcst:NAve=0
16:48655650:d=16102800:U:300 mb:23hr fcst:NAve=0
17:51899360:d=16102800:V:300 mb:23hr fcst:NAve=0
18:55143070:d=16102800:U:250 mb:23hr fcst:NAve=0
19:58386780:d=16102800:V:250 mb:23hr fcst:NAve=0
```

- Wgrib processes each gme output.
- Forecast time :0-23 hr.

- Vertical interpolation is processing.
- tso and wso

```
Variable: tso
Type: float
Total Size: 58384800 bytes
            14596200 values
Number of Dimensions: 3
Dimensions and sizes:   [lv_DBLL0 | 9] x [g0_lat_1 | 901] x [g0_lon_2 | 1800]
Coordinates:
            lv_DBLL0: [0..1458]
            g0_lat_1: [-90..90]
            g0_lon_2: [ 0..359.8]
Number Of Attributes: 11
  center :      Offenbach (RSMC)
  long_name :   temperature of soil layers
  units :       K
  _FillValue :  1e+20
  level_indicator :     111
  gds_grid_type :       0
  parameter_table_version :     201
  parameter_number :    197
  forecast_time :       23
  forecast_time_units : hours
  initial_time :        10/28/2016 (00:00)
```

Extract the Variables from GME output (4/4)

- ➔ If successful, these files should be in your WPS/conv_gme_for_wrf/intdata directory.
- ➔ Move on next step.

```
gme_plv.2016102800  gme_sfc.2016102812  TSO_2016102812.asc  WSO_2016102806.asc
gme_plv.2016102801  gme_sfc.2016102813  TSO_2016102812.grb  WSO_2016102806.grb
gme_plv.2016102802  gme_sfc.2016102814  TSO_2016102813.asc  WSO_2016102807.asc
gme_plv.2016102803  gme_sfc.2016102815  TSO_2016102813.grb  WSO_2016102807.grb
gme_plv.2016102804  gme_sfc.2016102816  TSO_2016102814.asc  WSO_2016102808.asc
gme_plv.2016102805  gme_sfc.2016102817  TSO_2016102814.grb  WSO_2016102808.grb
gme_plv.2016102806  gme_sfc.2016102818  TSO_2016102815.asc  WSO_2016102809.asc
gme_plv.2016102807  gme_sfc.2016102819  TSO_2016102815.grb  WSO_2016102809.grb
gme_plv.2016102808  gme_sfc.2016102820  TSO_2016102816.asc  WSO_2016102810.asc
gme_plv.2016102809  gme_sfc.2016102821  TSO_2016102816.grb  WSO_2016102810.grb
gme_plv.2016102810  gme_sfc.2016102822  TSO_2016102817.asc  WSO_2016102811.asc
gme_plv.2016102811  gme_sfc.2016102823  TSO_2016102817.grb  WSO_2016102811.grb
gme_plv.2016102812  TSO_2016102800.asc  TSO_2016102818.asc  WSO_2016102812.asc
gme_plv.2016102813  TSO_2016102800.grb  TSO_2016102818.grb  WSO_2016102812.grb
gme_plv.2016102814  TSO_2016102801.asc  TSO_2016102819.asc  WSO_2016102813.asc
gme_plv.2016102815  TSO_2016102801.grb  TSO_2016102819.grb  WSO_2016102813.grb
gme_plv.2016102816  TSO_2016102802.asc  TSO_2016102820.asc  WSO_2016102814.asc
gme_plv.2016102817  TSO_2016102802.grb  TSO_2016102820.grb  WSO_2016102814.grb
gme_plv.2016102818  TSO_2016102803.asc  TSO_2016102821.asc  WSO_2016102815.asc
gme_plv.2016102819  TSO_2016102803.grb  TSO_2016102821.grb  WSO_2016102815.grb
```

Making Intermediate Files

- ➔ Check the pgi compiler has been loaded. If not:
  module swap gnu/4.4.7 pgi/7.1.6
- ➔ Compile the fortran code:
  pgf90 –Mfree –byteswapio GME-WRF_intermediate_V1.f

```
-------------------------------------------------------------
  THE WPS INTERMEDIATE MAKING PROGRAM FOR PKNU/GME NI384 20KM
            DEVELOPED BY T. KIM (JAN. 15, 2009)
     INTEGRATED CLIMATE SYSTEM MODELING LABORATORY (ICSyM)
   PUKYONG NATIONAL UNIVERSITY (PKNU), BUSAN, 608-737, KOREA
-------------------------------------------------------------

THE STARTING AND ENDING DATES TO PROCESS!
  => STTING DATE :  2016  10  28  00
  => ENDING DATE :  2016  10  28  23
  => T. INTERVAL :     1  HOURS
  => TOTAL LOOP  :    24  LOOPS

READ GME 20km PRESSURE & SFC LEVEL DATA!
  => /home01/jwoh/WRFV3.7.1/WPS/conv_gme_for_wrf/intdata/2016102800/gme_plv.2016102800
  => /home01/jwoh/WRFV3.7.1/WPS/conv_gme_for_wrf/intdata/2016102800/gme_sfc.2016102800
   27224.13        1.000000        67919.60        228.3218
   10000.00        230.0924       8.1794409E-05   -3.720535
  -3.487091        0.000000        40.00000
  => /home01/jwoh/WRFV3.7.1/WPS/conv_gme_for_wrf/intdata/2016102800/TSO_2016102800.asc
   223.7076        223.4459        222.9429        222.4249
  => /home01/jwoh/WRFV3.7.1/WPS/conv_gme_for_wrf/intdata/2016102800/WSO_2016102800.asc
   0.000000        0.000000        0.000000        0.000000

WRITE WPS INTERMEDIATE FILES FOR METGRID!
  => /home01/jwoh/WRFV3.7.1/WPS/conv_gme_for_wrf/20km/2016102800/GME:2016-10-28_00
   230.0924       -3.720535       -3.487091        97.03172
   2777.972        98987.00        67919.60        228.3218
   223.7076        223.4459        222.9429        222.4249
   0.000000        0.000000        0.000000        0.000000
   1.000000        0.000000        10000.00        40.00000
```

Date information

Read GME 20km pressure & surface level data each time interval.

Write intermediate files for metgrid process.

Making Intermediate Files

➔ If successful, the intermediate files should be in your WPS/conv_gme_for_wrf/20km directory.

```
GME:2016-10-28_00  GME:2016-10-28_06  GME:2016-10-28_12  GME:2016-10-28_18
GME:2016-10-28_01  GME:2016-10-28_07  GME:2016-10-28_13  GME:2016-10-28_19
GME:2016-10-28_02  GME:2016-10-28_08  GME:2016-10-28_14  GME:2016-10-28_20
GME:2016-10-28_03  GME:2016-10-28_09  GME:2016-10-28_15  GME:2016-10-28_21
GME:2016-10-28_04  GME:2016-10-28_10  GME:2016-10-28_16  GME:2016-10-28_22
GME:2016-10-28_05  GME:2016-10-28_11  GME:2016-10-28_17  GME:2016-10-28_23
```

➔ These files will be used for running metgrid.exe.
(We will skip running ungrib process in afternoon class 'Hands on training 2')

Lunch

AM 12:00 ~ 14:00

Hands on training (2)

Hands on training (2)

Running WPS & WRF with agricultural meteorological examples

December 8, 2016

Sin-Il Yang, Ji-Won Oh, Kyung-Min Choi, and Ga-Eun Kim
Pukyong National University, Busan, Korea

Running a Model Data Case

Step to Run WPS

- Go to *WPS/*
- Edit *namelist.wps* for your case
- Run *geogrid.exe* to set up domain
  - Run *plotgrids.exe* to configure your domain

    (or use *plotgrid.ncl*)
- Run *ungird. exe* to degrib met data
- Run *metgrid.exe* to interpolate met data to model grid

A note on namelist

- A Fortran namelist contains a list of *runtime options* for the program to read in during its execution.
- Use of a **namelist** allows one to change runtime configuration without the need to recompile the source code.
- Fortran 90 namelist has very specific format, so edit with care:

```
&namelist-record
/
```

- start
- end

- **As a general rule:**
  - Multiple columns: domain dependent
  - Single column: value valid for all domains

example of a partial WPS namelist (1)

```
&share
 wrf_core = 'ARW',
 max_dom = 3,
 start_date = '2016-10-28_00:00:00','2016-10-28_00:00:00`,’2016-10-28_00:00:00',
 end_date   = '2016-10-28_23:00:00','2016-10-28_23:00:00','2016-10-28_23:00:00',
 interval_seconds = 3600
 io_form_geogrid = 2,
/
&geogrid
 parent_id         =   1,   1,  2,
 parent_grid_ratio =   1,   3,  3,
 i_parent_start    =   1,   69,  58,
 j_parent_start    =   1,   69,  95,
 e_we              =   201, 156, 100,
 e_sn              =   181, 156, 100,
 geog_data_res     =   '10m','2m','2m'
 dx = 9000,
 dy = 9000,
 map_proj  = 'lambert',
 ref_lat   =  36.60,
 ref_lon   = 127.70,
 truelat1  =  30.0,
 truelat2  =  60.0,
 stand_lon = 127.70,
 geog_data_path = '/share/data/geog'
/
```

example of a partial WPS namelist (2)

```
&ungrib
 out_format = 'WPS',
 prefix = 'GME',
/
&metgrid
 fg_name = 'GME'
 io_form_metgrid = 2,
/
```

Running geogrid

➔ Edit namelist records `&share` and `&geogrid`

➔ make sure GEOGRID.TBL is linked to GEOGRID.TBL.ARW (by default, it is)

➔ Type the following to run:

```
./geogrid.exe
```

➔ If successful, you should see

```
Successful completion of geogrid
```

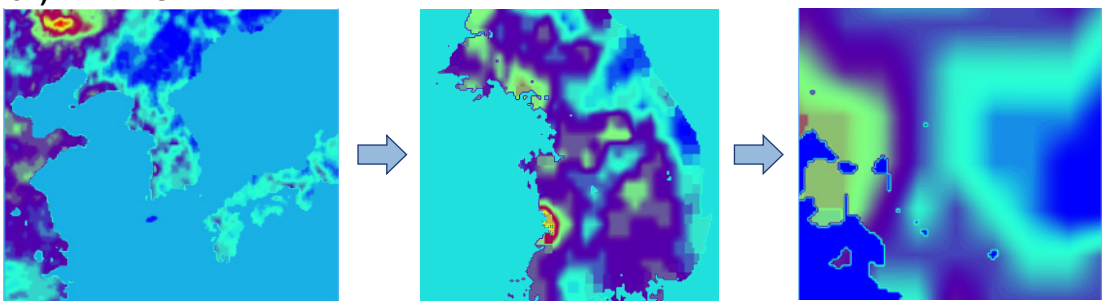
Running geogrid

➔ Output from geogrid:

`geo_em.d01.nc`
`geo_em.d02.nc` (for a nest)
`geo_em.d03.nc` (for a nest)

➔ Use tools like `ncview` to quickly check the output

ex) ALBEDO12M



`geo_em.d01.nc` `geo_em.d02.nc` `geo_em.d03.nc`

Running ungrib (general process)

- Edit namelist record `&share` (for dates) and `&ungrib` in `namelist.wps`
- Link the correct Vtable from `WPS/ungrib/Variable_Tables/` directory to the file name "Vtable" in the run directory. e.g.

```
ln -s ungrib/Variable_Tables/Vtable.GFS Vtable
```

- Link GRIB files using provided script `link_grib.csh`:

```
link_grib.csh /data/GRIB/gfs/gfs*
```

Running ungrib (general process)

- Type the following to run ungrib:

```
./ungrib.exe >& log.ungrib
```

If successful, you should see

```
Successful completion of ungrib
```

- output files from ungrib, one per time period:

```
GME:2016-10-28_00
GME:2016-10-28_01 …
```

We have already done the process of ungrib in 'training (1)'

We just type `ln -s /conv_gme_for_wrf/20km/* .`

Running metgrid

- Edit namelist record `&share` and `&ungrib`
- Type the following to run metgrid:

```
./metgrid.exe >& log.metgrid
```

- If successful, you should see

```
Successful completion of metgrid
```

- Output from metgrid program:

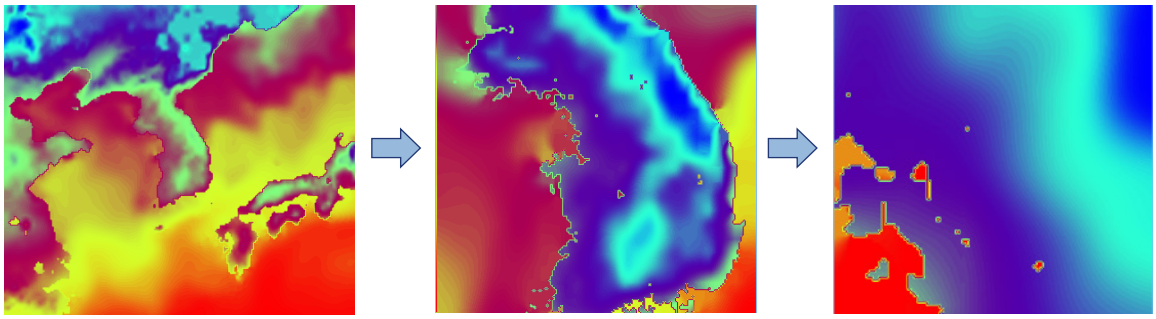
```
met_em.d01.2016-10-28_00:00:00
met_em.d01.2016-10-28_01:00:00 ...
met_em.d02.2016-10-28_00:00:00 ...
```

(for a nest, usually only one time period is needed)

Running metgrid

- Use tools like `ncview` to quickly check the output

ex) SKINTEMP



`met_em.d01` `met_em.d02` `met_em.d03`

Typical Errors Running WPS

- Using wrong Vtable
- Missing some surface data, which may result and error message like:

```
WRF_DEBUG: Warning DIM  4 , NAME num_metgrid_levels
REDIFINED  by var TT  27 26  in wrf_io.F90 line
2420 ERROR: Error in ext_pkg_write_field
```

- Missing soil temperature or moisture
- Check the log file from running ungrib to know what met fields you have got

Steps to Run real and wrf

- cd to *run/* or one of the test case directories
- Link or copy WPS output files to the directory for real-data cases
- Edit *namelist.input* file for the appropriate grid and times of the case
- Run initialization program (*ideal.exe*, *real.exe*)
- Run model executable, *wrf.exe*

Running Real-Data Case

➔ If you have compiled the *em_real* case, you should have:

real.exe - real data initialization program

wrf.exe - model executable

ndown.exe - program for doing one-way nesting

➔ These executables are linked to:

WRFV3/run

and

WRFV3/test/*em_real*

➔ One can go to either directory to run.

Running WRF Real-Data Case

➔ One must successfully run WPS, and create **met_em.*** file for more than one time period

➔ Link or copy WPS output files to the run directory:

```
cd test/em_real
ln -s ../../../WPS/met_em.* .
```

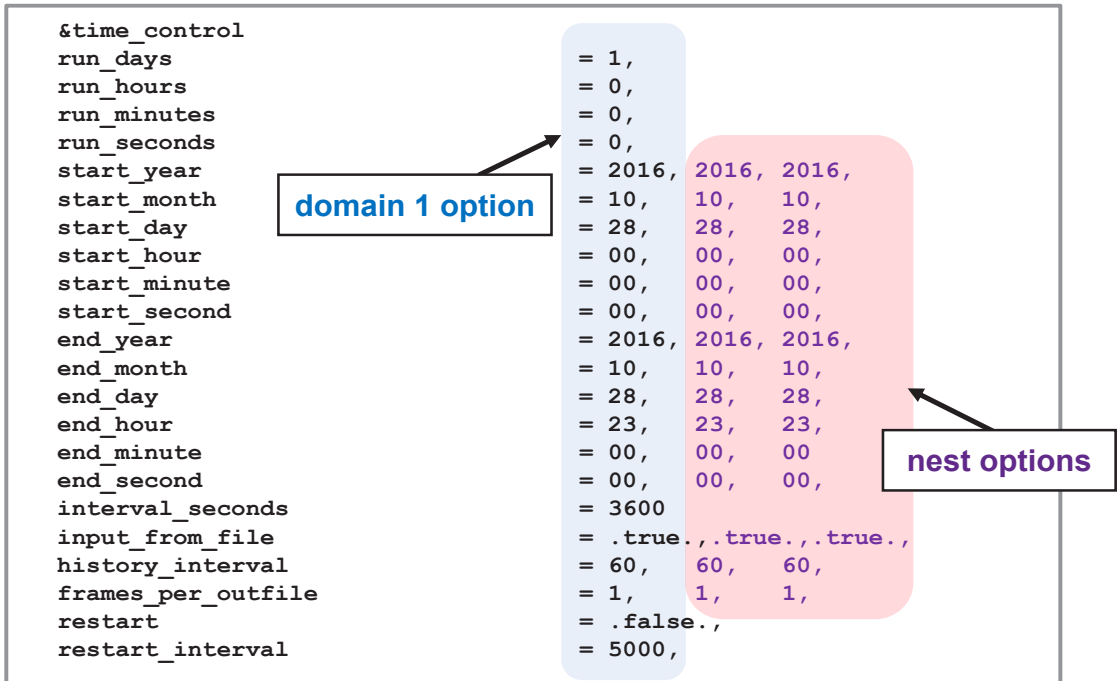
Running WRF Real-Data Case

➔ Edit **`namelist.input`** file for runtime options

(at minimum, one must edit **`&time_control`** for start, end and integration times, and **`&domains`** for grid dimensions)

Pay attention to first column in the **`namelist.input`** file if you are using a single domain. For nested runs, multiple columns of namelists need to be edited.

example of `namelist.input` file: `&time_control`



example of `namelist.input` file: `&domains`

```
&domains
time_step                           = 81,
time_step_fract_num                 = 0,
time_step_fract_den                 = 1,
max_dom                             = 3,
e_we                                = 201,  157, 100,
e_sn                                = 181,  157, 100,
e_vert                              = 36,   36,   36,
p_top_requested                     = 5000,
num_metgrid_levels                  = 18,
num_metgrid_soil_levels             = 4,
dx                                  = 9000, 3000, 1000,
dy                                  = 9000, 3000, 1000,
grid_id                             = 1,     2,      3,
parent_id                           = 0,     1,      2,
i_parent_start                      = 1,     69,    58,
j_parent_start                      = 1,     69,    95,
parent_grid_ratio                   = 1,     3,      3,
parent_time_step_ratio              = 1,     3,      3,
feedback                            = 1,
smooth_option                       = 0
/
```

Match the dimensions defined in WPS

nest options

Running WRF Real-Data Case

➔ Run the real-data initialization program:

`./real.exe,` if compiled serially / SMP, or
`mpirun -np N ./real.exe,` for a MPI job where is ***N*** the number of processors requested.

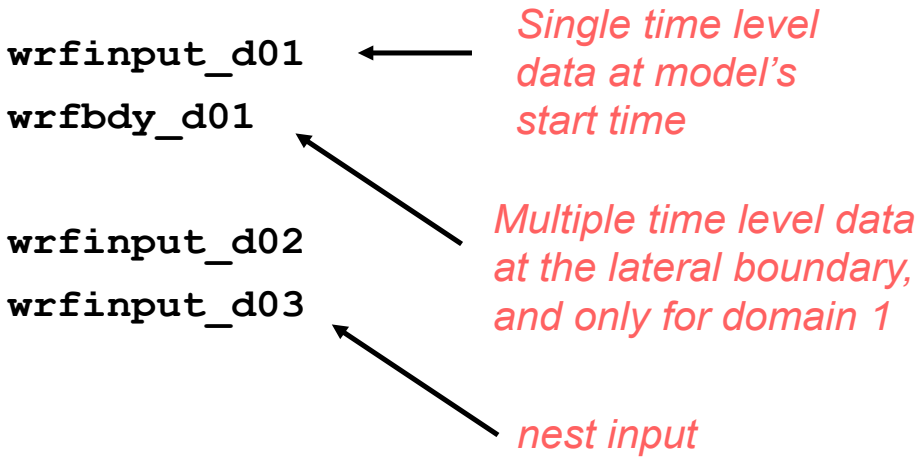
➔ Depending on how the computer is configured, one may need to specify *machinefile* option:

`mpirun -machinefile mach -np N ./real.exe`

- here `'mach'` is a file containing a list of machine/processor names

Running WRF Real-Data Case

➔ Successfully running this program will create model initial and boundary files:



Running WRF Real-Data Case

➔ Run the model executable by typing:

```
./wrf.exe >& wrf.out &
```

or

```
mpirun -np N ./wrf.exe &
```

➔ Successfully running the model will a create model *history* file:

```
wrfout_d01_2016-10-28_00:00:00 …
wrfout_d02_2016-10-28_00:00:00 …
wrfout_d03_2016-10-28_00:00:00 …
```

Where do I start?

➔ **Always start with a namelist template provided in a test case directory, whether it is a ideal or real data case.**

- A number of namelist templates are provided in *test/test-case/* directories

➔ **Use document to guide the modification of the namelist values:**

- run/README.namelist
- Full list of namelists and their default values can be found in Registry files: Registry.EM, and registry.io_boilerplate (IO options)

To run a job in a different directory

➔ Directories *run/* and test_<*case*>/ are convenient places to run, but it does not have to be.

➔ Copy or link the content of these directories to another directory, including physics data files, wrf input and boundary files and wrf namelist and executables, and you should be able to run a job anywhere on your system.

Check Output

Output After a Model Run

- **Standard out/error files:**

  `wrf.out`, or `rsl`.* files

- **Model history file(s):**

  `wrfout_d01_<date>`

- **Model restart file(s), optional**

  `wrfrst_d01_<date>`

Output from a multi-processor run

➔ The standard out and error will go to the following files for a MPI run:

```
mpirun -np 4 .wrf.exe ➔
```

```
rsl.out.0000        rsl.error.0000
rsl.out.0001        rsl.error.0001
rsl.out.0002        rsl.error.0002
rsl.out.0003        rsl.error.0003
```

➔ There is one pair of files for each processor requested

What to Look for in a standard out File?

➔ Check run log file by typing

```
tail wrf.out,
```
or

```
tail rsl.out.0000
```

➔ You should see the following if the job is successfully completed:

```
wrf: SUCCESS COMPLETE WRF
```

How to Check Model History File?

➔ Use **ncdump**:

ncdump -v Times wrfout_d01_<date>

to check output times. Or

ncdump -v U wrfout_d01_<date>

to check a particular variable (U)

➔ Use **ncview** or **ncBrowse** (great tools!)

➔ Use post-processing tools (see talks later)

What is in a *wrf.out* or *rsl* file?

➔ A print of namelist options

➔ Time taken to compute one model step:

```
Timing for main: time 2016-10-28_00:01:39 on domain   1:    7.39984 elapsed seconds
Timing for main: time 2016-10-28_00:03:00 on domain   1:    3.51456 elapsed seconds
Timing for main: time 2016-10-28_00:04:21 on domain   1:    3.43645 elapsed seconds
Timing for main: time 2016-10-28_00:05:42 on domain   1:    3.45390 elapsed seconds
```

➔ Time taken to write history and restart file:

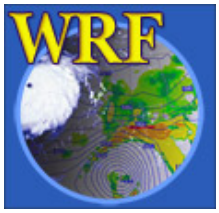
```
Timing for Writing wrfout_d01_2016-10-28_00:00:00 for domain   1:   3.57025 elapsed seconds
```

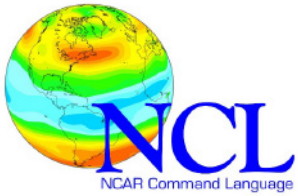
➔ Any model error prints:

```
5 points exceeded cfl=2 in domain 1 at time 4.200000 MAX AT i,j,k: 123 48 3 cfl,w,d(eta)=
4.165821
```

→ An indication the model has become numerically unstable

Hands on training (3)





Hands on training (3)

Practice post-processing WRF-ARW data with the NCAR Command Language (NCL)

December 8, 2016

Sin-Il Yang, Ji-Won Oh, Kyung-Min Choi, and Ga-Eun Kim
Pukyong National University, Busan, Korea

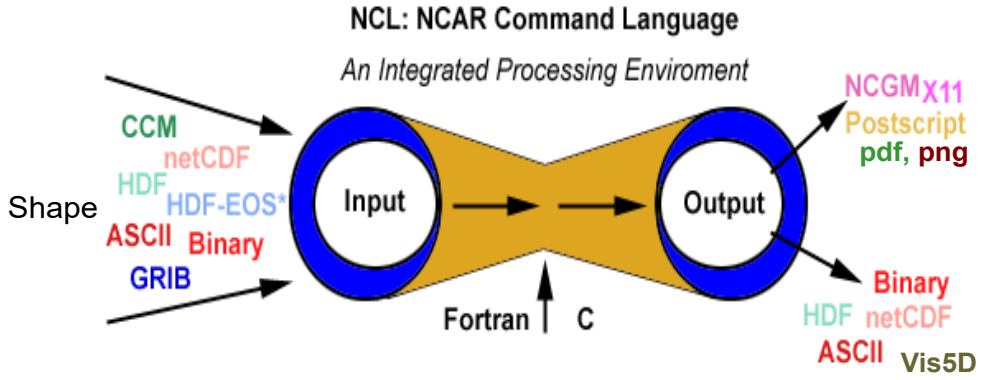
Main goals

1. **Introduce you to NCL and WRF-NCL**

2. **Get you familiar with WRF-NCL scripts**
   - Opening and examining a WRF output data file
   - Reading and querying variables
   - Plotting variables

NCL Overview

- **Integrated** data processing environment



- **freeware:** supported, public domain
- **portable:** linux/unix, windows (cygwin), MacOS
- **general purpose:** unique capabilities
- **excellent 2D graphics** (limited 3D)

Introduction: Key Points

NCL execution: scripts via unix command line
- unix prompt> ncl test ncl

NCL variable model ←→ netCDF variable model

basic syntax to access/create the variable model
- = , := , @ , ! , &, $

Data Printing
- Still the best for debugging; (NCL no built-in debugger)

netCDF / NCL Relationship

- netCDF is a **file format** ; NCL is a **language.** What is the connection?

- NCL **variable** model is **based** on the netCDF variable model

- NCL makes GRIB, HDF, HDF-EOS **look like** netCDF files

- This **consistent view** of variables and file contents from disparate file formats is a very powerful feature of NCL.

Parts of netCDF file

Example) ncdump –h foo.nc (or **ncl_filedump** foo.nc)

DIMENSION SIZES & NAMES
dimensions:
lat = 64
lon = 128
time = 12

time=UNLIMITED (12 currently)

FILE ATTRIBUTES
global attributes:
title = "Temp: 1999"
source = "NCAR"
Conventions = "CF-1.0"

Exercise:
ncl_filedump foo.nc | less
ncl_filedump foo.grb | less

VARIABLES:
Names , Types, Attributes, Coordinate Variables
variables:
float lat(lat)
lat:long_name = "latitude"
lat:units = "degrees_north"
float lon(lon)
lon:long_name = "longitude"
lon:units = "degrees_east"
double time(time)
time:long_name = "time"
time:units = "hours_since …"
float T(time, lat, lon)
T:long_name = "Temperature"
T:units = "degC"
T:**missing_value** = 1.e+20f
T:**_FillValue** = 1.e+20f

NCL/netCDF Variable Semantics

Example) variable #1

```
double T(time, lat, lon)
  T: long_name = “Temperature”
  T: units = “degC"
  T: _FillValue = 1.e+20f
```

variable type – double (float, int, short,..)
variable name – T
named dimensions – time, lat, lon
attributes – long_name, units, _FillValue

Example) variable #2

```
float prr(time, y, x)
        prr:_FillValue = -9999.f ;                          CF
        prr:missing_value = -9999.f ;                       COARDS
        prr:long_name = "Liquid Precipitation" ;            CF, COARDS
        prr:grid_mapping = "Lambert_Conformal" ;
        prr:units = "kg m-2 s-1" ;                          CF, COARDS
        prr:height = "surface" ;
        prr:coordinates = "lon lat" ;                       CF
```

netCDF/NCL variable

- **array** [could be of length 1 (**scalar**)]
- (may have) additional information: **not required**

x

```
  4.35    4.39    0.27   -3.35   -6.90
  4.36    4.66    3.77   -1.66    4.06
  9.73   -5.84    0.89    8.46   10.39
    17    3.68    5.08    0.14   -5.63
 -0.63   -4.12   -2.51    1.76   -1.43
 -4.29    0.07    5.85    0.87    8.65
```

name: x
type: float [real]
shape: 2-dimensions
size: 6 (**rows**) x 5 (**columns**)
values: x(2,3) = 8.46 [row major, 0-based indexing]

long_name: “Temperature”
units: “degC”
named dimensions: x(time,lat)
lat: (/ -60, -30 ,0, 30, 60 /)
time: (/2000, 2001, 2002, 2003, 2004, 2005, 2006 /)

Meta data

netCDF [NCL] Variable model

X

Scalar
or
Array

attributes

long_name
_FillValue
units
add_offset
scale_factor
etc.

coordinates

time
lev
lat
lon
etc.

```
f = addfile("uv300.nc","r")    ; grb/hdf
x = f->X
```

NCL reads the scalar/array, attributes, and coordinate variables as one object (structure)

X

accessed via @

accessed via &

values

Scalar
or
Array

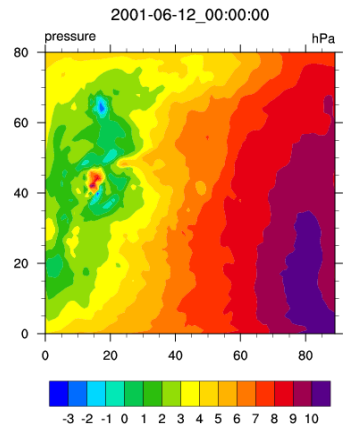
attributes

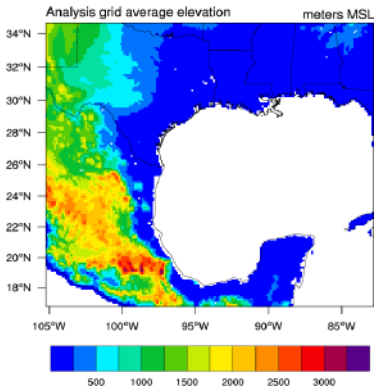
long_name
_FillValue
units
add_offset
scale_factor
etc.

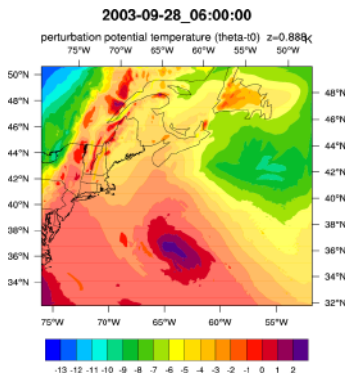
coord var

time
lev
lat
lon
etc.

Example plots of WRF-NCL

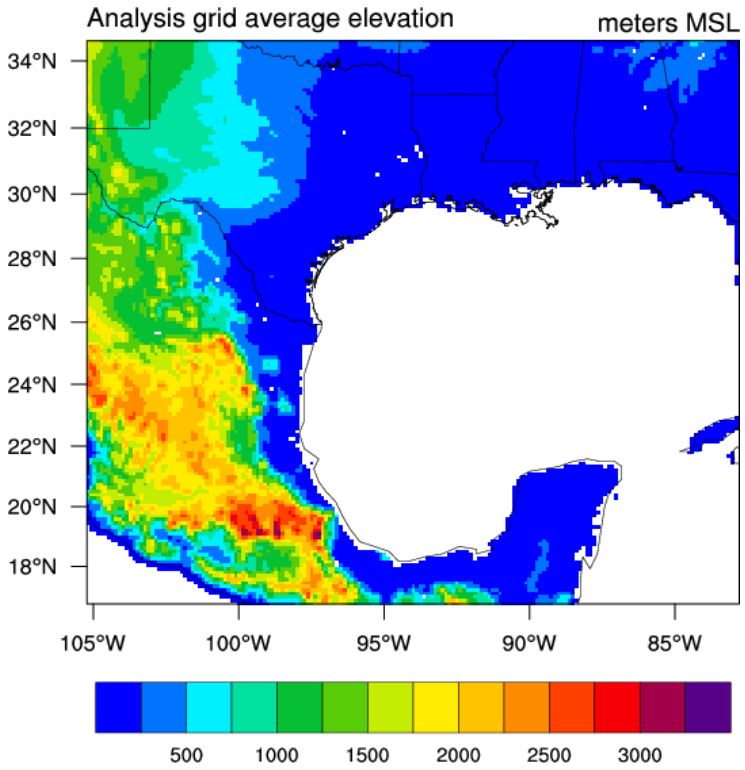






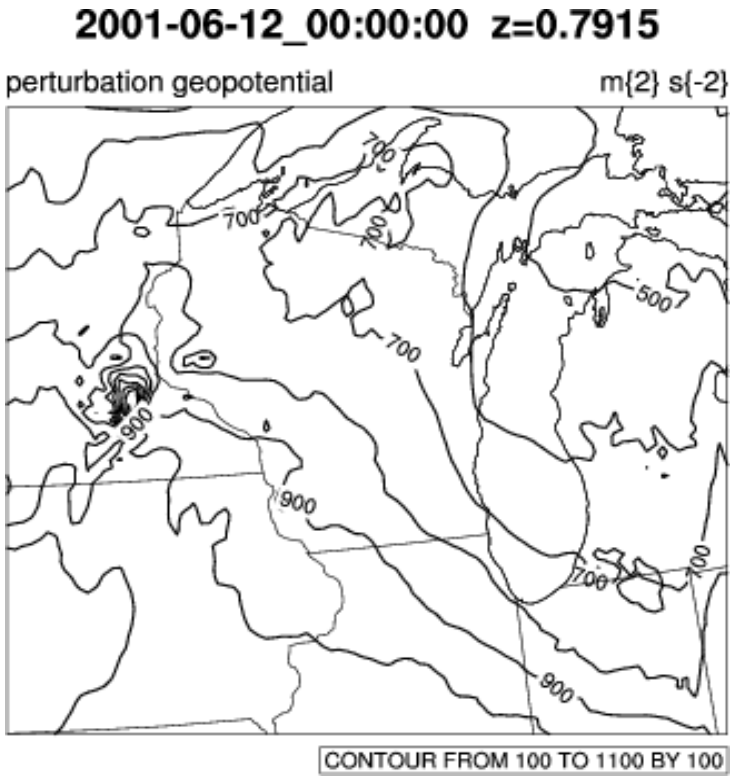
Sample read of the average height.

WRF_cn_2.ncl



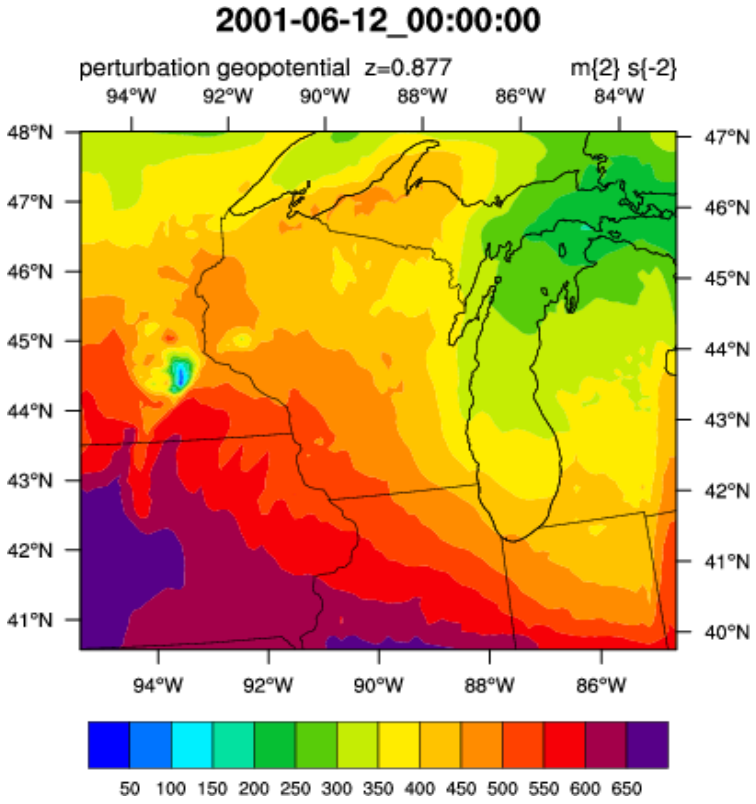
Basic black and white contour on **Lambert Conformal Map**

WRF_lc_1.ncl



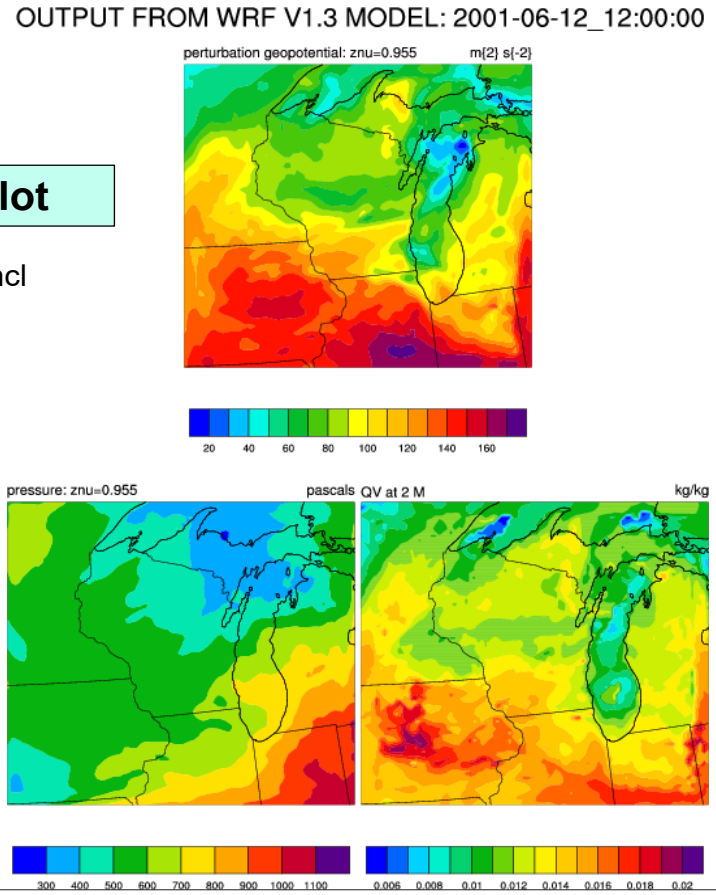
A color version

WRF_lc_2.ncl



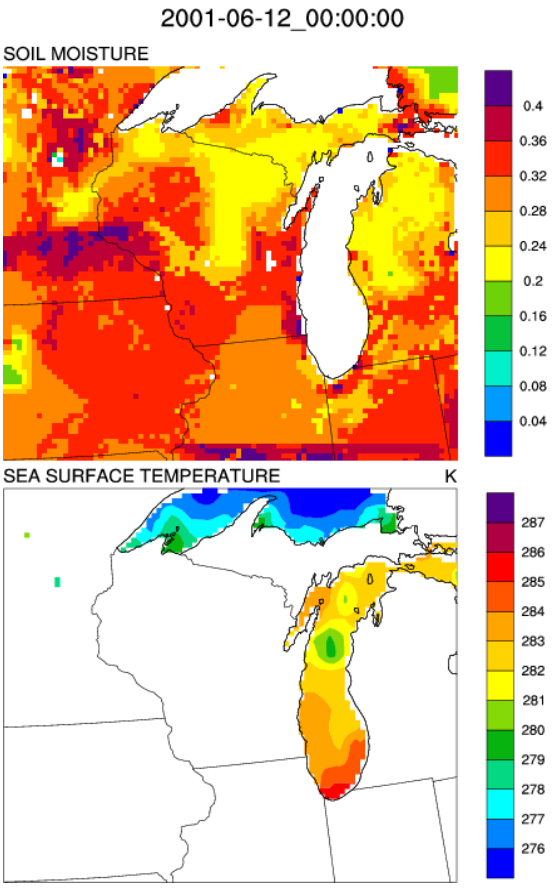
A panel plot

WRF_lc_3.ncl



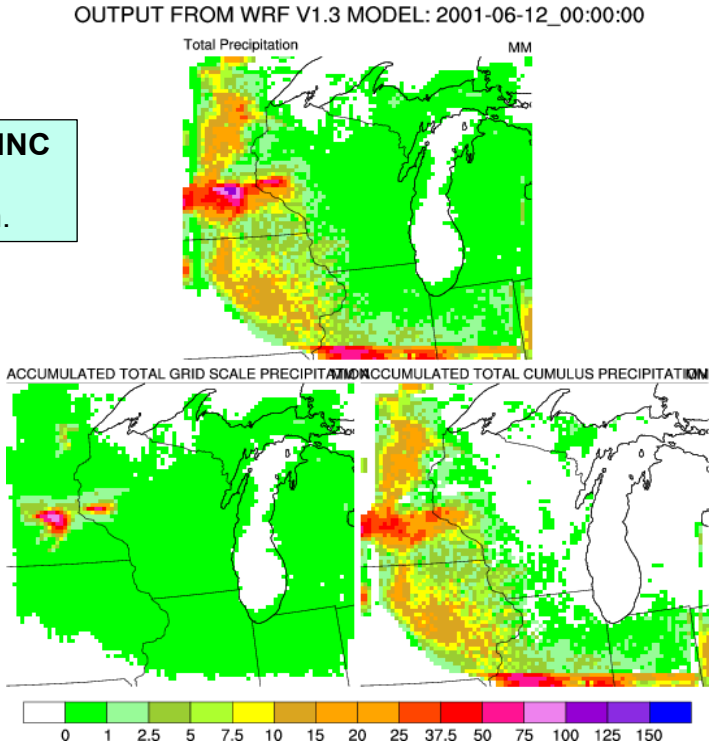
The **SMOIS** (soil moisture) and **SST** (sea surface temperature)

WRF_lc_5.ncl

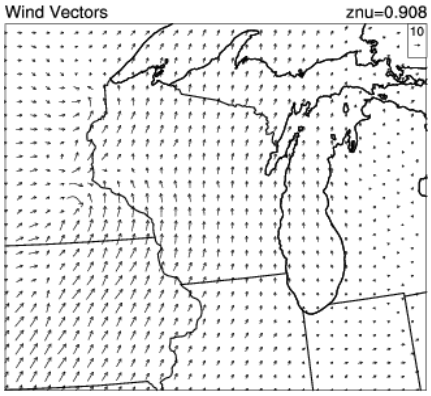


Use the **RAINC** and **RAINNC** variables **to calculate the total precipitation**.

WRF_lc_6.ncl



Basic vectors 2001-06-12_00:00:00

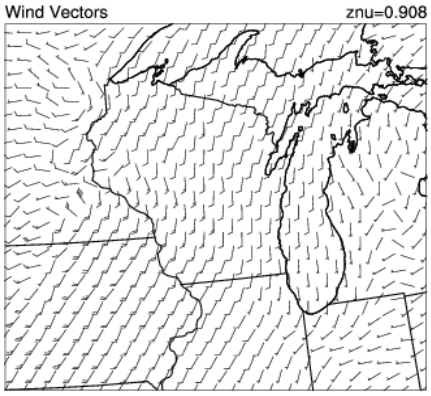


The **U and V components** are on a **staggered grid**.

Array syntax is used to **create u and v on the mass grid** whose grid locations are specified **by XLAT and XLONG**.

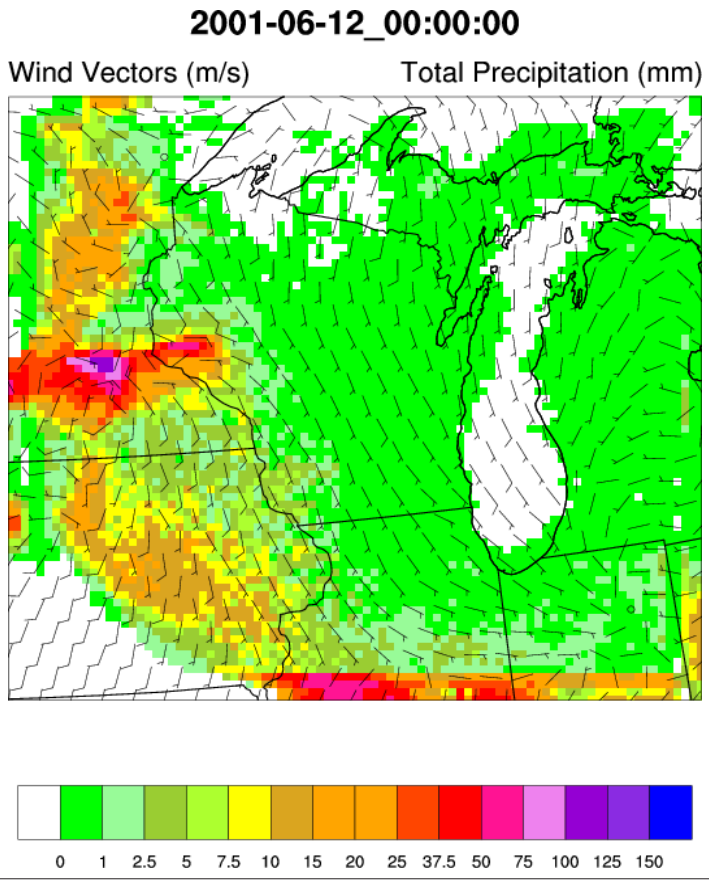
WRF_lc_7.ncl

wind barbs 2001-06-12_00:00:00



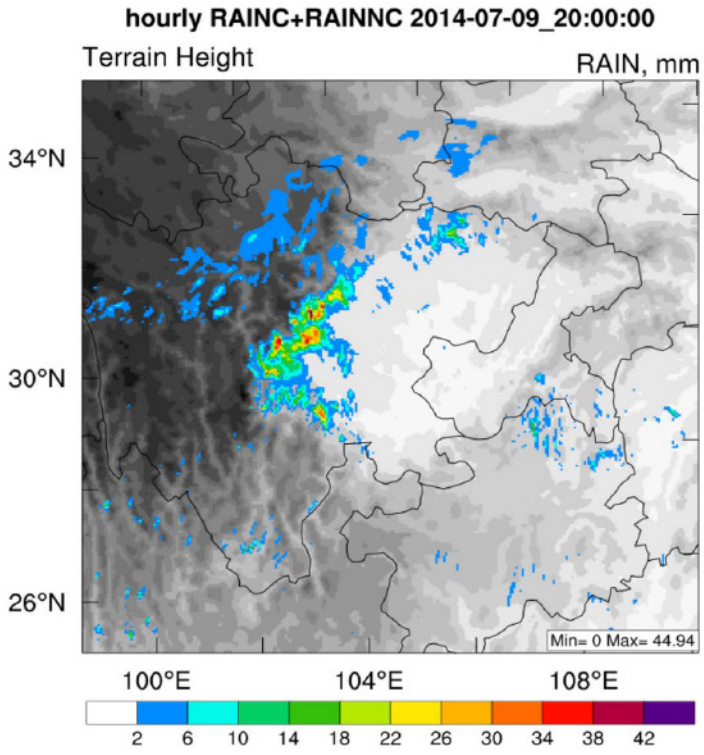
Overlay winds at 10 meters over the total precipitation.

WRF_lc_8.ncl



Overlay precipitation on a terrain map

WRF_pcp_2.ncl



Q&A