WRF Overview

WRF Modeling System Overview

Jimy Dudhia

What is WRF?

- WRF: Weather Research and Forecasting Model
  - Used for both research and operational forecasting
- It is a supported “community model”, i.e. a free and shared resource with distributed development and centralized support
- Its development is led by NCAR, NOAA/GSD and NOAA/NCEP/EMC with partnerships at AFWA, FAA, NRL, and collaborations with universities and other government agencies in the US and overseas

What are ARW and NMM?

- The Advanced Research WRF (ARW) and Nonhydrostatic Mesoscale Model (NMM) are dynamical cores
  - Dynamical core includes mostly advection, pressure-gradients, Coriolis, buoyancy, filters, diffusion, and time-stepping
- Both are Eulerian mass dynamical cores with terrain-following vertical coordinates
- ARW support and development are centered at NCAR/MMM
- NMM development is centered at NCEP/EMC and support is provided by NCAR/DTC
- This tutorial is for both dynamical cores
- Both are downloadable in the same WRF tar file
- Physics, the software framework, and parts of data pre- and post-processing are shared between the dynamical cores

WRF as a Community Model

- Version 1.0 WRF was released December 2000
- Version 2.0 May 2004 (NMM added, EM nesting released)
- Version 2.1 August 2005 (EM becomes ARW)
  - Version 2.1.1 Nov 2005 (NMM released)
  - Version 2.1.2 Jan 2006
- Version 2.2 December 2006 (WPS released)
  - NMM nesting released in 2007
  - 2.2.1 released in Nov 2007
- Version 3.0 released in April 2008
- Version 3.1 to be released March 2009

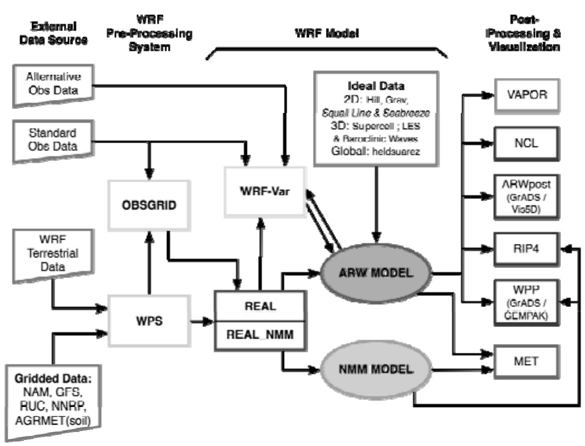
What can WRF be used for?

- ARW and NMM
  - Atmospheric physics/parameterization research
  - Case-study research
  - Real-time NWP and forecast system research
  - Teaching dynamics and NWP
- ARW only
  - Regional climate and seasonal time-scale research
  - Coupled-chemistry applications
  - Global simulations
  - Idealized simulations at many scales (e.g. convection, baroclinic waves, large eddy simulations)
  - Data assimilation research

Who uses WRF?

- Academic atmospheric scientists (dynamics, physics, weather, climate research)
- Forecast teams at operational centers
- Applications scientists (e.g. Air Quality, Hydrology, Utilities)

WRF Modeling System Flow Chart



Modeling System Components

- WRF Pre-processing System (WPS)
  - Real-data interpolation for NWP runs
  - New obsgrid program for adding more obs to analysis
- WRF Model (ARW and NMM dynamical cores)
  - Initialization programs for real and (for ARW) idealized data (real.exe/ideal.exe)
  - Numerical integration program (wrf.exe)
- Graphics and verification tools including MET
- WRF-Var (separate tutorial)
- WRF-Chem (separate tutorial)

WPS Functions

- Define simulation domain area (and nests)
- Produce terrain, landuse, 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, sea-surface temperature, etc.)
- Interpolate meteorological data to WRF model grid (horizontally)
- Optionally add more observations to analysis (separate obsgrid program)

WPS and WRF

Running WPS

- Several executable stages with namelist input
  - geogrid.exe (interpolate maps and time-independent fields)
  - ungrib.exe (convert time-dependent Grib-formatted data to simple binary format)
  - metgrid.exe (interpolate time-dependent initial and boundary data)
  - obsgrid.exe (optional stage to add more observations)

Running WRF

- Two executable stages with namelist input
  - real.exe or real_nmm.exe (set up vertical model levels for model input and boundary files)
  - wrf.exe (run model)

ARW only

WRF-Var Functions

- Variational data assimilation
- Ingest observations into WRF input analysis from WPS
- May be used in cycling mode for updating WRF initial conditions after WRF run
- Also used for observation impact data studies

WRF 3DVAR

- Supported data types
  - Conventional surface and upper air, wind profiler
  - Remote sensing data: Cloud-track winds, ATOVS thickness, ground-based GPS TPW, SSM/I, SSM/T1, SSM/T2, SSM/I brightness temp, Quikscat ocean surface winds, radar radial velocity and reflectivity
- Background error covariance for ARW from
  - NMC method
  - Ensemble method

WRF-Chem

- Supported by NOAA/ESRL
- Includes chemistry species and processes
- Also needs emissions data
- Included in WRF tar file, but requires special compilation option

WRF real and ideal functions

- REAL
  - Creates initial and boundary condition files for real-data cases
  - Does vertical interpolation to model levels (when using WPS)
  - Does vertical dynamic (hydrostatic) balance
  - Does soil vertical interpolations and land-use mask checks
- IDEAL (ARW only)
  - Programs for setting up idealized case
  - Simple physics and usually single sounding
  - Initial conditions and dynamic balance

WRF Model

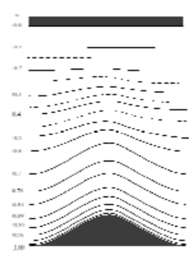
- WRF
  - Dynamical core (ARW or NMM) is compile-time selectable
  - Uses initial conditions from REAL or IDEAL
  - Real-data cases use boundary conditions from REAL
  - Runs the model simulation with run-time selected namelist switches (such as physics choices, timestep, length of simulation, etc.)
  - Outputs history and restart files

ARW Dynamics

Key features:

- Fully compressible, non-hydrostatic (with hydrostatic option)
- Mass-based terrain following coordinate, η

$$\eta = \frac{(\pi - \pi_t)}{\mu}, \qquad \mu = \pi_s - \pi_t$$



where π is hydrostatic pressure, μ is column mass

- Arakawa C-grid staggering

```
   v
u  T  u
   v
```

ARW Model

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

ARW Model

Key features:

- Choices of lateral boundary conditions suitable for real-data and idealized simulations
  - Specified, Periodic, Open, Symmetric, Nested
- Full physics options to represent atmospheric radiation, surface and boundary layer, and cloud and precipitation processes
- Grid-nudging and obs-nudging (FDDA)
- New Digital Filter Initialization option

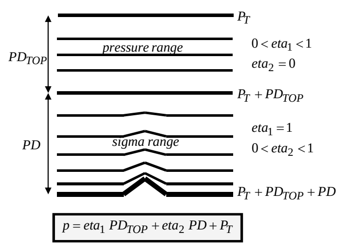
NMM Dynamics

Key features:

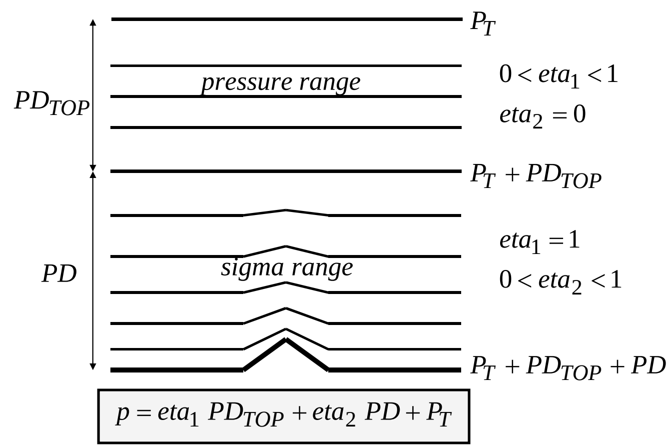
- Fully compressible, non-hydrostatic or hydrostatic
- Mass-based sigma-pressure hybrid terrain following coordinate similar to ARW but with constant pressure surfaces above 400 hPa
- Arakawa E-grid staggering

T **V** T
V T **V**
T **V** T

where **V** is u and v



NMM Dynamics



NMM Model

Key features:

- Adams-Bashforth and Crank-Nicholson time integration schemes
- High-order advection scheme
- Scalar and energy conserving
- Coriolis, curvature and mapping terms
- One-way and two-way nesting

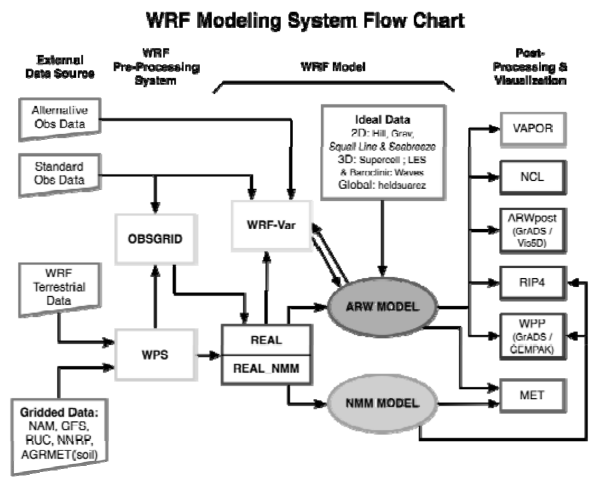
NMM Model

Key features:

- Lateral boundary conditions suitable for real-data and nesting
- Full physics options to represent atmospheric radiation, surface and boundary layer, and cloud and precipitation processes

Graphics and Verification Tools

- ARW and NMM
  - RIP4 (Read, Interpolate and Plot)
  - WRF Post-Processor (WPP)
    - Conversion to GriB (for GrADS and GEMPAK)
  - MET (Model Evaluation Toolkit)
- ARW
  - NCAR Graphics Command Language (NCL)
  - ARWPost
    - Conversion program for GrADS and Vis5D



Basic Software Requirement

- Fortran 90/95 compiler
- C compiler
- Perl
- netCDF library
- Public domain mpich for MPI

Portability

- Runs on Unix single, OpenMP and MPI platforms:
  - IBM SP AIX (xlf)
  - Linux (PGI, Intel, g95, gfortran, Pathscale compilers)
  - SGI Altix (Intel)
  - Cray XT (PGI, Pathscale)
  - Mac Darwin (xlf, PGI, Intel, g95 compilers)
  - Others (HP, Sun, SGI Origin, Compaq)

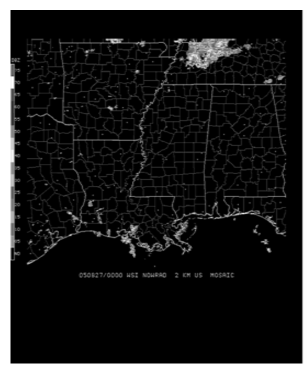
User Support

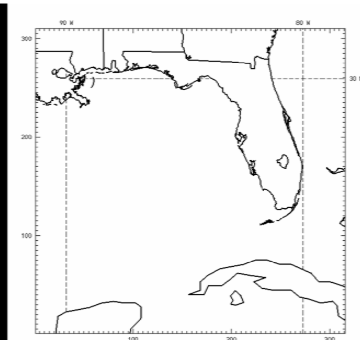
- Email: wrfhelp@ucar.edu
- User Web pages:

  ARW:http://www.mmm.ucar.edu/wrf/users/

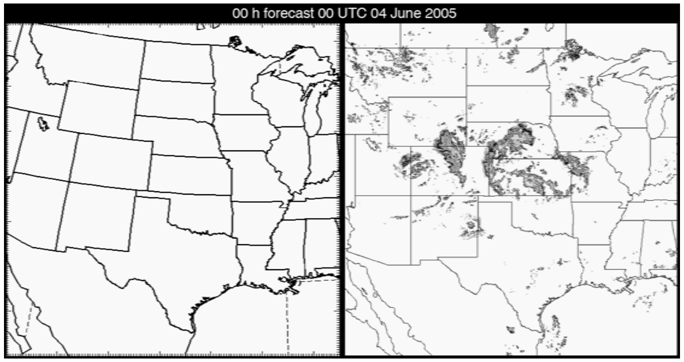
  NMM:http://www.dtcenter.org/wrf-nmm/users/
  - Latest update for the modeling system
  - WRF software download
  - Various documentation
    - Users' Guide
    - Technical Note (ARW Description)

ARW Hurricane Katrina Simulation (4km)





ARW Convective-scale Forecasting (4km)



Tutorial Schedule

- Lectures for WRF: Mon.-Fri.
- Practice for WRF: Mon.-Fri.
  - 2 Groups (a.m./p.m.)
- Tutorial Lunch: Wed.
- Ends Friday pm
- Next week
  - WRF-Var tutorial: Mon.-Wed.
  - MET tutorial: Wed.-Thu.

WPS
Description of General Functions

The WRF Preprocessing System: Description of General Functions

Michael Duda

Purpose of this Lecture

In this lecture, our goals are to:

1) Understand the purpose of the WPS
2) Learn what each component of the WPS does

– The details of *actually running* the WPS are covered in the second WPS lecture

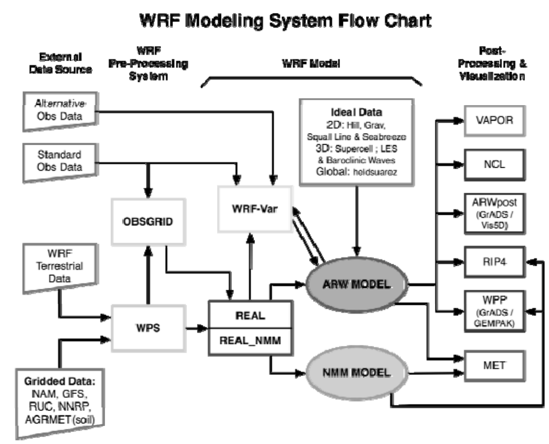
– *Advanced usage* of the WPS is covered in the third lecture

Purpose of the WPS

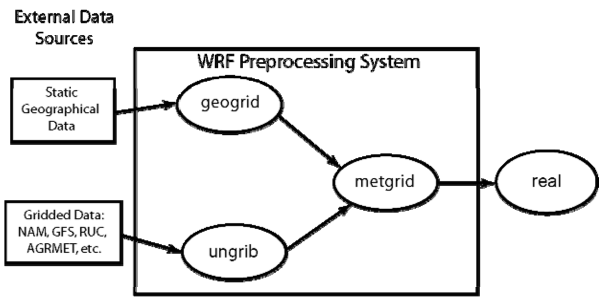
The purpose of the WPS is to prepare input to WRF for real-data simulations:

1. Defines simulation domain and ARW nested domains
2. Computes latitude, longitude, map scale factors, Coriolis parameters at every grid point
3. Interpolates time-invariant terrestrial data to simulation grids (e.g., terrain height and soil type)
4. Interpolates time-varying meteorological fields from another model onto simulation domains

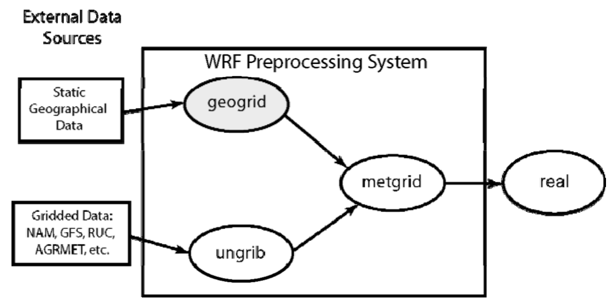
WRF Modeling System Flowchart



WPS Program Flowchart



The *geogrid* program



geogrid: think <u>geo</u>graphical

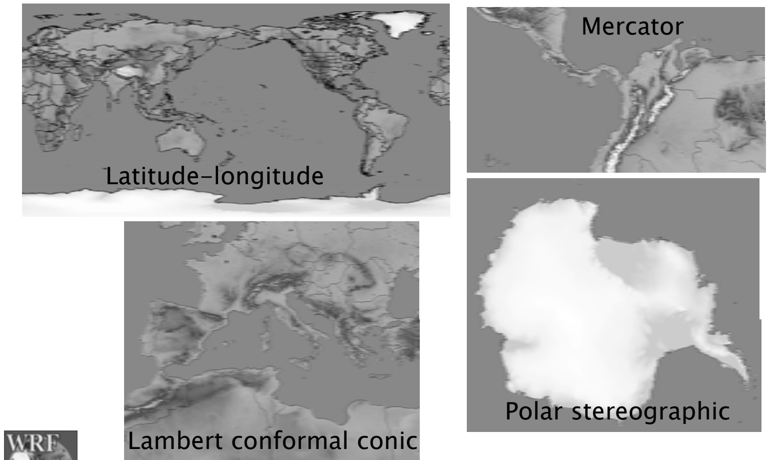
The *geogrid* program

- For WRF model domains, geogrid defines:
  - Map projection (all domains must use the same)
  - Location of domains
  - Dimensions of domains
- Geogrid provides values for static (time-invariant) fields at each model grid point
  - Compute latitude, longitude, map scale factor, and Coriolis parameters at each grid point
  - Horizontally interpolate static terrestrial data (e.g., topography height, land use category, soil type, vegetation fraction, monthly surface albedo)

Geogrid: Defining model domains

- First, we choose a map projection to use for the domains; why?
  - The real earth is (roughly) an ellipsoid
  - But WRF computational domains are defined by rectangles in the plane
- NMM uses a rotated latitude-longitude projection
- ARW can use any of the following projections:
  1. Lambert conformal
  2. Mercator
  3. Polar stereographic
  4. Latitude-longitude (for global domain, *must* choose this!)

Supported Projections in ARW



Geogrid: Defining Model Domains

- Define projection of domains using subset of the following parameters
  - **MAP_PROJ**: ‘lambert’, ‘mercator’, ‘polar’, ‘lat-lon’, or ‘rotated_ll’
  - **TRUELAT1**: First true latitude (*)
  - **TRUELAT2**: Second true latitude (*only for Lambert conformal*) (*)
  - **POLE_LAT, POLE_LON**: Location of North Pole in WRF computational grid (*only for ‘lat-lon’*) (*)
  - **STAND_LON**: The meridian parallel to *y*-axis
- All parameters reside in the file *namelist.wps*

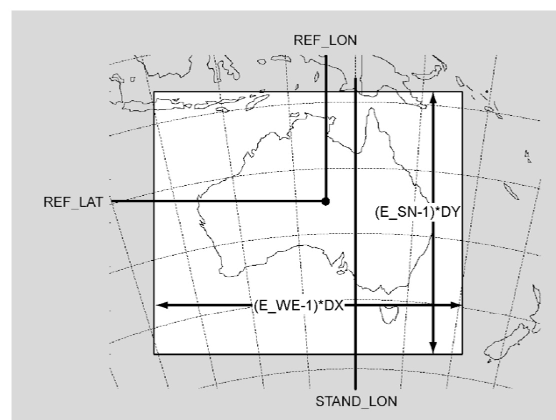
*ARW only

See p. 3-9 and 3-34

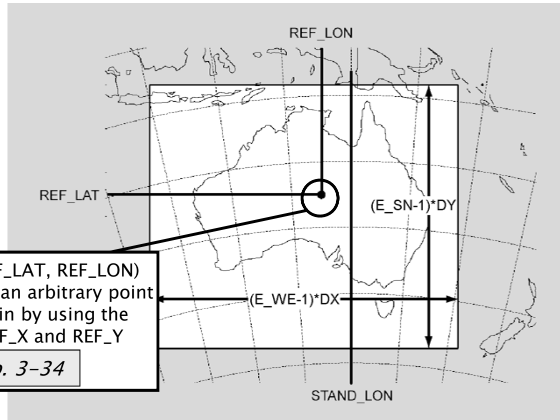
Geogrid: Defining Model Domains

- Define the area covered (dimensions and location) by coarse domain using the following:
  - **REF_LAT, REF_LON**: The (lat,lon) location of a known location in the domain (*by default, the center point of the domain*)
  - **DX, DY**: Grid distance where map factor = 1
    - For Lambert, Mercator, and polar stereographic: **meters**
    - For (rotated) latitude-longitude: **degrees**
  - **E_WE**: Number of velocity points in west-east direction for ARW; number of mass points in odd rows for NMM
  - **E_SN**: Number of velocity points in south-north direction for ARW; number of rows for NMM

Geogrid: Defining ARW Domains



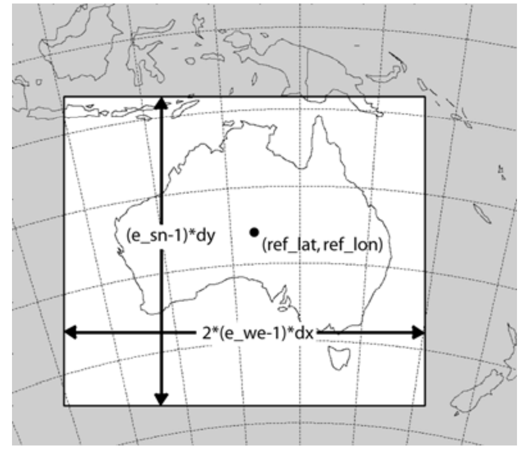
Geogrid: Defining ARW Domains



In ARW, (REF_LAT, REF_LON) can refer to an arbitrary point in the domain by using the variables REF_X and REF_Y

See p. 3-34

Geogrid: Defining NMM Domains

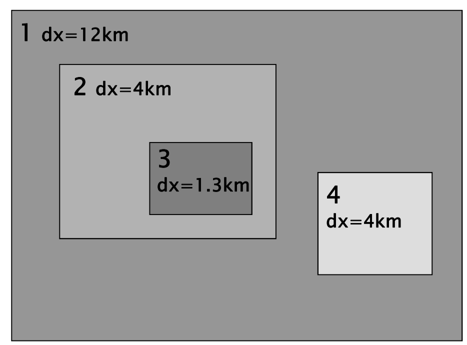


Geogrid: Nesting Basics

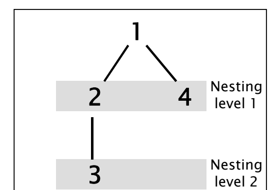
- A *nested domain* is a domain that is wholly contained within its *parent domain* and that receives information from its parent, and that may also feed information back to its parent
  - A nested domain has exactly one *parent*
  - A domain may have one or more *children*
- 2-*way nests* on the same *nesting level* must not overlap in coverage!

Geogrid: Nesting Example

Example configuration – 4 domains



Each domain is assigned a *domain ID #*



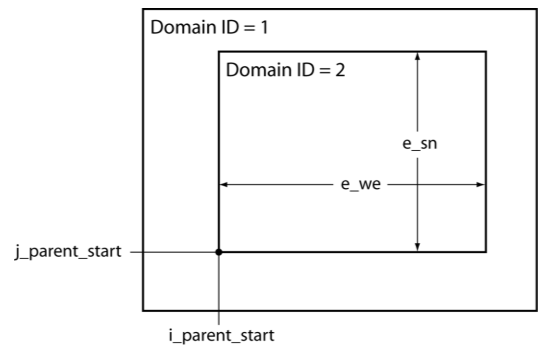
Nesting structure shown as a tree for the domains at left

Geogrid: Defining Nested ARW Domains

- Define the dimensions and location of nested domains using:
  - **PARENT_ID**: Which domain is the parent?
  - **PARENT_GRID_RATIO**: What is the ratio between grid spacing in parent to grid spacing in this nest?
  - **I_PARENT_START**: *i*-coordinate in parent of this nest's lower-left corner
  - **J_PARENT_START**: *j*-coordinate in parent of this nest's lower-left corner
  - **E_WE**: Number of velocity points in west-east direction
  - **E_SN**: Number of velocity points in south-north direction

See p. 3-15 and 3-33

Geogrid: Defining Nested Domains

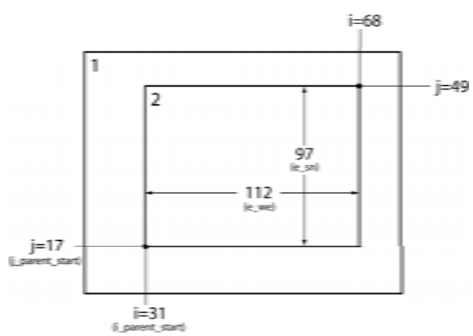


The grid spacing (*dx)* of domain 2 is determined by grid spacing of domain 1 and the *parent_grid_ratio*

NB: For NMM, the parent_grid_ratio is always 3!

Geogrid: Nesting example

Assuming *parent_grid_ratio* = 3



In ARW, nest dimensions must be

(*n* parent_grid_ratio* + 1)

for some integer *n*

112 = 3*n+1 for n=37

97 = 3*n+1 for n=32

Geogrid: Interpolating Static Fields

- Given definitions of all computational grids, geogrid interpolates terrestrial, time-invariant fields
  - Topography height
  - Land use categories
  - Soil type (top layer & bottom layer)
  - Annual mean soil temperature
  - Monthly vegetation fraction
  - Monthly surface albedo

Geogrid: Interpolating Static Fields

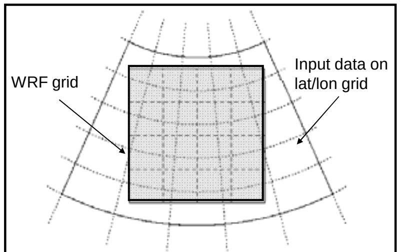


Figure 4.1

In general, source data are given on a different projection from the model grid

Geogrid: Interpolation Options

- 4-point bilinear
- 16-point overlapping parabolic
- 4-point average (simple or weighted)
- 16-point average (simple or weighted)
- Grid cell average
- Nearest neighbor
- Breadth-first search

See p. 3-45

Why have so many interpolation options?

- Different interpolators work best for different fields and different relative grid resolutions
  - Some interpolators preserve positive definiteness
  - Some interpolators produce "smoother" fields
  - Some interpolators are best suited for discrete or categorical fields
  - Some are good when going from a fine grid to a coarse grid
- Having a choice of how to interpolate fields is good!
  - We'll see in the third WPS lecture how several different options can be used for different regions of the same field

Geogrid: Program Flexibility

- The GEOGRID.TBL file determines
  1. Which fields will be produced by geogrid
  2. What sources of data will be used
  3. How the data will be interpolated/smoothed
  4. Any derived fields (e.g., dominant cat., df/dx)
- Acceptable defaults exist in GEOGRID.TBL, so user will not generally need to edit the file (*but more on this in the third WPS lecture*!)

Geogrid: Program Flexibility

- *geogrid* is flexible enough to ingest and interpolate new static fields
  - handles either continuous or categorical fields
- New data sets must be written to simple binary format
- User needs to add an entry to the file `GEOGRID.TBL`

Geogrid: Program Output

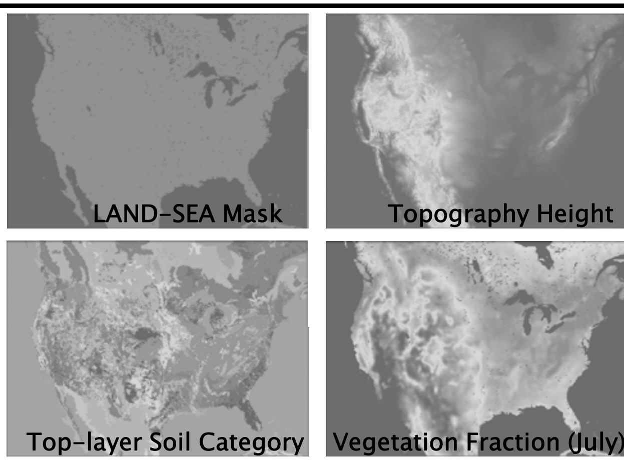
- The parameters defining each domain, plus interpolated static fields, are written using the WRF I/O API
  - One file per domain for ARW
  - One file per *nesting level* for NMM
- Filenames: geo_em.d0*n*.nc , or

  geo_nmm.d01.nc, geo_nmm_nest.l0*k*.nc

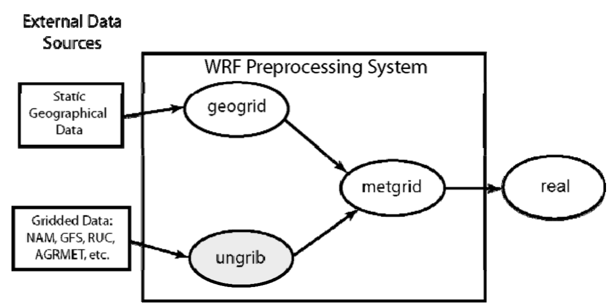
  (where *n* is the domain ID # and *k* is the nest level)
- Example:

geo_em.d01.nc	geo_nmm.d01.nc
geo_em.d02.nc (nest)	geo_nmm_nest.l01.nc (nest level)
geo_em.d03.nc (nest)	geo_nmm_nest.l02.nc (nest level)

Geogrid: Example Output Fields



The *ungrib* program



ungrib: think un+grib

The *ungrib* program

- Read GRIB Edition 1 and GRIB Edition 2 files
- Extract meteorological fields
- If necessary, derive required fields from related ones
  - E.g., Compute RH from T, P, and Q
- Write requested fields to an intermediate file format

Ungrib: Vtables

How does ungrib know which fields to extract?

Using Vtables (think: Variable tables)

- Vtables are files that give the GRIB codes for fields to be extracted from GRIB input files
- One Vtable for each source of data
- Vtables are provided for: NAM 104, NAM 212, GFS, AGRMET, and others

Ungrib: Example Vtable

GRIB1 Param	Level Type	From Level1	To Level2	UNGRIB Name	UNGRIB Units	UNGRIB Description
11	100	*		T	K	Temperature
33	100	*		U	m s-1	U
34	100	*		V	m s-1	V
52	100	*		RH	%	Relative Humidity
7	100	*		HGT	m	Height
11	105	2		T	K	Temperature at 2 m
52	105	2		RH	%	Relative Humidity at 2 m
33	105	10		U	m s-1	U at 10 m
34	105	10		V	m s-1	V at 10 m
1	1	0		PSFC	Pa	Surface Pressure
130	102	0		PMSL	Pa	Sea-level Pressure
144	112	0	10	SM000010	kg m-3	Soil Moist 0-10 cm below grn layer (Up)
144	112	10	40	SM010040	kg m-3	Soil Moist 10-40 cm below grn layer
144	112	40	100	SM040100	kg m-3	Soil Moist 40-100 cm below grn layer
144	112	100	200	SM100200	kg m-3	Soil Moist 100-200 cm below gr layer
85	112	0	10	ST000010	K	T 0-10 cm below ground layer (Upper)
85	112	10	40	ST010040	K	T 10-40 cm below ground layer (Upper)
85	112	40	100	ST040100	K	T 40-100 cm below ground layer (Upper)
85	112	100	200	ST100200	K	T 100-200 cm below ground layer (Bottom)
91	1	0		SEAICE	proprtn	Ice flag
81	1	0		LANDSEA	proprtn	Land/Sea flag (1=land,2=sea in GRIB2)
7	1	0		HGT	m	Terrain field of source analysis
11	1	0		SKINTEMP	K	Skin temperature (can use for SST also)
65	1	0		SNOW	kg m-2	Water equivalent snow depth
223	1	0		CANWAT	kg m-2	Plant Canopy Surface Water
224	1	0		SOILCAT	Tab4.213	Dominant soil type category
225	1	0		VEGCAT	Tab4.212	Dominant land use category

Ungrib: GRIB2 Vtable Entries

metgrid Description	GRIB2 Discp	GRIB2 Catgy	GRIB2 Param	GRIB2 Level
Temperature	0	0	0	100
U	0	2	2	100
V	0	2	3	100
Relative Humidity	0	1	1	100
Height	0	3	5	100
Temperature at 2 m	0	0	0	103
Relative Humidity at 2 m	0	1	1	103
U at 10 m	0	2	2	103
V at 10 m	0	2	3	103
Surface Pressure	0	3	0	1
Sea-level Pressure	0	3	1	101
Soil Moist 0-10 cm below grn layer (Up)	2	0	192	106
Soil Moist 10-40 cm below grn layer	2	0	192	106
Soil Moist 40-100 cm below grn layer	2	0	192	106
Soil Moist 100-200 cm below gr layer	2	0	192	106
Soil Moist 10-200 cm below gr layer	2	0	192	106
T 0-10 cm below ground layer (Upper)	0	0	0	106
T 10-40 cm below ground layer (Upper)	0	0	0	106
T 40-100 cm below ground layer (Upper)	0	0	0	106
T 100-200 cm below ground layer (Bottom)	0	0	0	106
T 10-200 cm below ground layer (Bottom)	0	0	0	106
Ice flag	0	2	0	1
Land/Sea flag (1=land, 0 or 2=sea)	2	0	0	1
Terrain field of source analysis	2	0	7	1
Skin temperature (can use for SST also)	0	0	0	1
Water equivalent snow depth	0	1	13	1
Dominant soil type cat.(not in GFS file)	2	3	0	1
Dominant land use cat. (not in GFS file)	2	0	198	1

Ungrib: Vtables

What if a data source has no existing Vtable?

Create a Vtable

- Get a listing of GRIB codes for fields in the source
  - Check documentation from originating center or use utility such as *wgrib, g1print, g2print*
- Use existing Vtable as a template
- Check documentation in Chapter 3 of the Users' Guide for more information about Vtables

See p. 3-27

Ungrib: Intermediate File Format

- After extracting fields listed in Vtable, ungrib writes those fields to intermediate format
- For meteorological data sets not in GRIB format, the user may write to intermediate format directly
  - *Allows WPS to ingest new data sources*; basic programming required of user
  - Simple intermediate file format is easily read/written using routines from WPS (read_met_module.F and write_met_module.F)

See p. 3-25

Ungrib: Program Output

- Output files named *FILE:YYYY-MM-DD_HH*
  - *YYYY* is year of data in the file; *MM* is month; *DD* is day; *HH* is hour
  - All times are UTC
- Example:

  FILE:2007-07-24_00

  FILE:2007-07-24_06

  FILE:2007-07-24_12

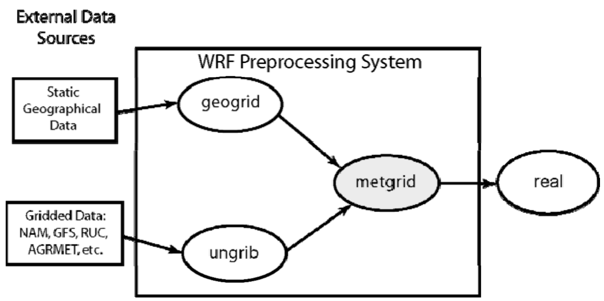
ungrib can also write intermediate files in the MM5 or WRF SI format! *(To allow for use of GRIB2 data with MM5, for example)*

Ungrib: Obtaining GRIB Data

- Where does one get GRIB data?
  - User's responsibility
  - Some free data are available from NCAR and NCEP. See
  - *http://www.mmm.ucar.edu/wrf/users/*

    > under the "Downloads" tab:
    - Some NCEP data in the past year
    - NCEP operational data available daily

The *metgrid* program



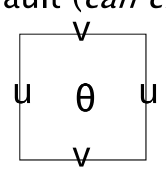
metgrid: think <u>met</u>eorological

The *metgrid* program

- Horizontally interpolate meteorological data (*extracted by ungrib*) to simulation domains (*defined by geogrid*)
  - Masked interpolation for masked fields
- Rotate winds to WRF grid
  - i.e., rotate so that U-component is parallel to *x*-axis, V-component is parallel to *y*-axis

Metgrid: ARW Grid Staggering

- For ARW, wind U-component interpolated to "u" staggering
- Wind V-component interpolated to "v" staggering
- Other meteorological fields interpolated to "θ" staggering by default (*can change this!*)



A single ARW grid cell, with "u", "v", and "θ" points labeled.

Metgrid: NMM Grid Staggering

- For NMM, wind U- and V-components interpolated to "V" staggering
- Other meteorological fields interpolated to "H" staggering by default (*can change this!*)

H	V	H	V	H
V	H	V	H	V
H	V	H	V	H

An NMM grid showing "V", and "H" points.

Metgrid: Interpolation Options*

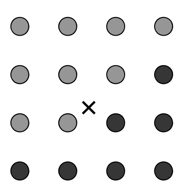
- 4-point bilinear
- 16-point overlapping parabolic
- 4-point average (simple or weighted)
- 16-point average (simple or weighted)
- Grid cell average
- Nearest neighbor
- Breadth-first search

* These are the same options available for geogrid!

Metgrid: Masked Interpolation

- *Masked fields* may only have valid data at a subset of grid points
  - E.g.: SST field only valid on water points
- When metgrid interpolates masked fields, it must know which points are invalid (masked)
  - Can use separate mask field (e.g., LANDSEA)
  - Can rely on special values (e.g., 1×10^{30}) in field itself to identify masked grid points

Metgrid: Masked Interpolation



● = valid source data

● = masked/invalid data

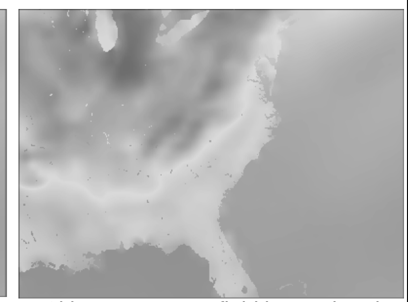
Suppose we need to interpolate to point X

- Using red points as valid data can give a bad interpolated value!
- Masked interpolation only uses valid blue points to interpolate to X

Not every interpolation option can handle masked points; we'll address this issue in the third lecture

Example: Masked Interpolation

Skin temperature field interpolated from GFS 0.5-deg field with no mask using a sixteen-point interpolator.



Skin temperature field interpolated using masks: GFS water points interpolated to model water points, GFS land points interpolated to model land points.

Metgrid: Wind Rotation

- Input wind fields (U-component + V-component) are either:
  - **Earth-relative**: U-component = westerly component; V-component = southerly component
  - **Relative to source grid**: U-component (V-component) parallel to source model x-axis (y-axis)
- WRF expects wind components to be relative to the simulation grid

Metgrid: Wind Rotation Example

A wind vector, shown in terms of its U and V components with respect to the source grid.

The same vector, in terms of its U and V components with respect to the WRF simulation grid.

This process may require *two* rotations: one from source grid to earth grid and a second from earth grid to WRF grid

Metgrid: Constant Fields

- For short simulations, some fields may be constant
  - E.g., SST or sea-ice fraction
- Use namelist option `CONSTANTS_NAME` option to specify such fields:
  - `CONSTANTS_NAME = 'SST_FILE:2007-07-24_00'`

Metgrid: Program Flexibility

- *metgrid* is capable of interpolating both isobaric and native vertical coordinate data sets
- User may specify interpolation methods and related options in the `METGRID.TBL` file
  - `METGRID.TBL` file similar in format to the file `GEOGRID.TBL`

Metgrid: Program Output

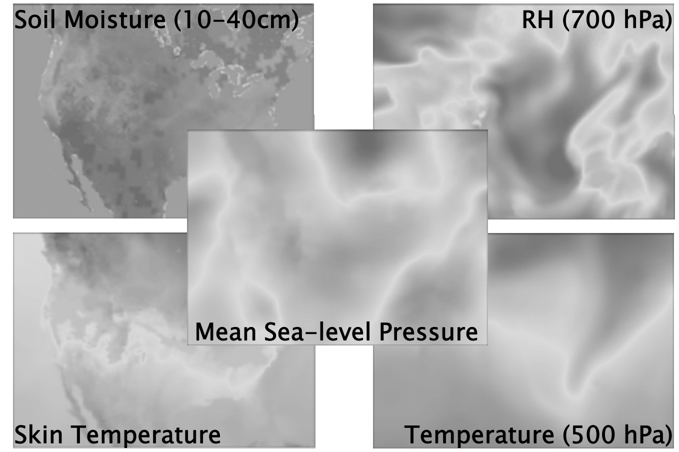
- For coarse domain, one file per time period
  - In ARW, we also get the first time period for all nested grids
- Files contain static fields from geogrid plus interpolated meteorological fields
- Filenames:

ARW: `met_em.d0`*`n`*`.`*`YYYY-MM-DD_HH:mm:ss`*`.nc`

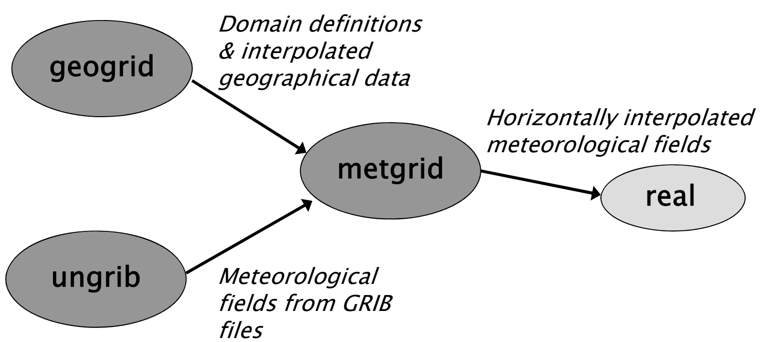
(where *n* is the domain ID #)

NMM: `met_nmm.d01.`*`YYYY-MM-DD_HH:mm:ss`*`.nc`

Metgrid: Example Output

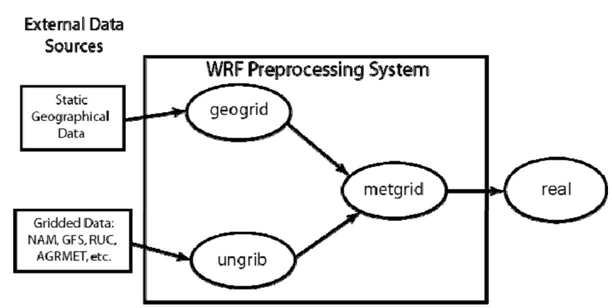


WPS Summary



And finally…

Vertical interpolation to WRF eta levels is performed in the *real* or *real_nmm* program



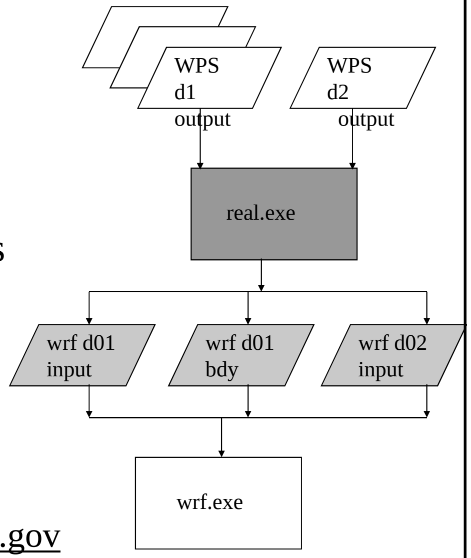
Real
Description of General Functions

Real

Description of General Functions

Dave Gill
gill@ucar.edu

Matt Pyle
matthew.pyle@noaa.gov

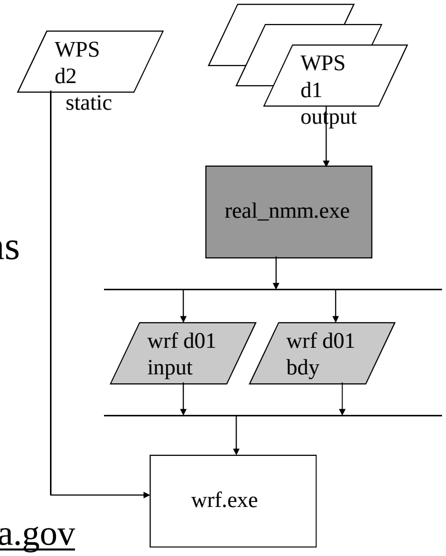


Real

Description of General Functions

Dave Gill
gill@ucar.edu

Matt Pyle
matthew.pyle@noaa.gov



Real-Data Initialization – ARW & NMM

- Definition of Terms
- Purpose and Tasks of Initialization Program
- Files before and after

Definition of Terms: real.exe & real_nmm.exe

- The ARW WRF model pre-processor is ***real.exe***
- The NMM WRF model pre-processor is ***real_nmm.exe***
- The real.exe and real_nmm.exe programs are available ***serial*** or ***DM parallel*** (primarily for aggregate memory purposes, as opposed to timing performance)
- This program is automatically generated when the model is built and the requested use is for a real data case
- The real.exe and real_nmm.exe programs take data ***from WPS*** and transform the data ***for WRF***
- Similar to the ARW idealized data pre-processor, both real.exe and real_nmm.exe are tightly coupled to the WRF model through the ***Registry***

Definition of Terms: Real Data Case

- ***3D forecast*** or simulation
- ***Meteorological input*** data that primarily originated from a previous forecast or analysis, probably via the WPS package
- Anticipated ***utilization of physics*** packages for microphysics, surface conditions, radiation, convection, and boundary layer (ARW: maybe usage of nudging capabilities)

Definition of Terms: Real Data Case

- A non-Cartesian ***projected domain***
  - ARW: Lambert conformal, Mercator, polar stereographic, rotated latitude/longitude (global or regional)
  - NMM: rotated latitude/longitude
- Selection of ***realistic static fields*** of topography, land use, vegetation, and soil category data
- Requirement of ***time dependent*** lateral boundary conditions for a regional forecast

Definition of Terms: Initialization

- Not referring to the ***Variational*** or the ***Digital Filtering*** usage of Initialization
- Generation of ***diagnostics*** necessary for assumed WRF model input
- Input field ***adjustment*** for consistency of static and time dependent fields (land mask with soil temperature, etc.)
- ARW: computation of ***reference*** and ***perturbation*** fields
- Generation of ***initial*** state
  - ARW: for each of the requested domains
  - NMM: for the coarse grid only
- Creation of a ***lateral boundary file*** for the most coarse domain
- ***Vertical interpolation*** for 3d meteorological fields and for sub-surface soil data

Real-Data Initialization – ARW & NMM

- Definition of Terms
- Purpose and Tasks of Initialization Program
- Files before and after

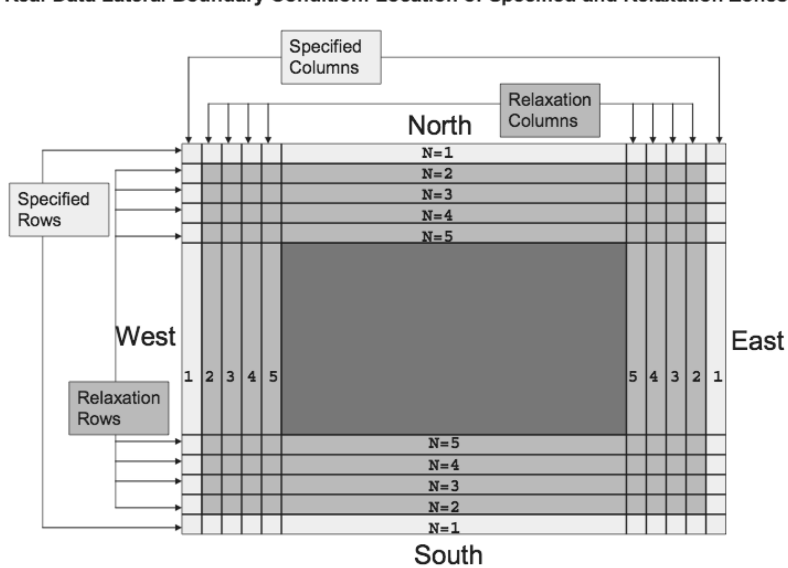
Tasks of the Initialization Program

- The primary purpose for the Real program (either ***real.exe*** and ***real_nmm.exe***) is to ***input*** data from WPS and create data for the WRF model, for a specific dynamical core. For the basic configuration, both an initial (***wrfinput_d01***) and a lateral boundary (***wrfbdy_d01***) file are generated.

Tasks of the Initialization Program ***Input Data*** for real.exe

- Ingest ***time dependent*** upper-air (horizontal winds, height, temperature, relative humidity), surface (SLP, surface pressure, elevation, sea ice, sea-surface temperature, skin temperature), and sub-surface (soil temperature, soil moisture)
- Ingest ***static fields*** of terrestrial (elevation, land use, vegetation category, soil texture category, monthly climo for greenness and albedo) and projection (map factors, latitude and longitude, projection rotation angles) information
- ***Multiple time periods*** of data are processed for the outer-most grid (for the lateral boundary conditions)
  - ARW: the ***initial time of the fine grid*** is processed
  - NMM: ***fine grid*** static information is provided by ***WPS*** directly to the model

Real-Data Lateral Boundary Condition: Location of Specified and Relaxation Zones



Tasks of the Initialization Program ***Consistency Checks***

- ARW: defining ***sea ice*** based on user criteria: a water point and the skin temperature or sea-surface temperature is cold enough (user defined setting, default about 271 K)
  - Switching to a sea ice point requires changing approximately a dozen associated fields: turn the location into a land point, fix the soil category and land use category
  - Compute a sub-surface temperature, linearly interpolated from the sea-surface temperature and the skin temperature (for example, 4 levels evenly spaced through a depth of 3 m for the Noah LSM scheme)

Tasks of the Initialization Program
Consistency Checks

- NMM: defining ***sea ice*** based on nearby fields:
  - If a land point or a water point is surrounded by sea ice, turn the middle value into a sea ice point, reset the land mask to a water point
  - After adjustment, make sure that the SST, skin temperature, land mask, and sea ice all agree

Tasks of the Initialization Program
Consistency Checks

- Figure out what ***optional data*** is available (soil data, sea-surface temperature, surface pressure, elevation of first guess data)
- ARW: consistency check for ***land mask*** and time dependent fields
  - Land grid points require fields such as soil category, skin temperature, soil temperature (optionally soil moisture, depending on the surface physics selection)
  - If not all of these fields are available, the grid point is turned into a water point

Tasks of the Initialization Program
Consistency Checks

- ARW: If the first-guess (GFS, NAM, etc.) elevation is available:
  - -6.5 K/km lapse rate is applied for the ***soil temperature*** and ***skin temperature*** fields
  - Large elevation adjustments (> 3 km) are bypassed as probably reflecting flag values in the first guess elevation
  - Water points for skin temperature are skipped for the elevation-based lapse rate adjustment.
- Assignment of ***sea-surface temperature*** to the skin temperature array when the location is a water point as defined by the land mask field

Tasks of the Initialization Program
Consistency Checks

- NMM: Modify the model topography when it differs significantly from the input hybrid surface height:
  - If the incoming topo is ***more than 150 m less*** than RUC, set the topo to RUC surface height minus 150
  - If the incoming topo is ***more than 150 m greater*** than RUC, set the topo to RUC surface height plus 150
- NMM: ***Smooth*** the lateral boundary ***topography*** (6 mass points in from the left and right, 12 rows in from the top and bottom) if not a water point

Tasks of the Initialization Program
Consistency Checks

- Assignment of reasonable fields to ***skin temperature*** if the field is undefined at the location due to internal consistency checks or if the WPS provided a flag value:
  – ARW: 0 – 10 cm soil temperature, sea-surface temperature, annual mean temperature, surface air temperature
  – NMM: surface air temperature

Tasks of the Initialization Program
Consistency Checks

- Verify that necessary fields for each grid point are available (***bounds check***)
- Stop code prior to model running is obvious errors occur in soil temperature, soil moisture, skin temperature, deep soil/annual mean temperature, surface pressure, sea-level pressure

Tasks of the Initialization Program
Consistency Checks

- The soil moisture field for the ***Noah LSM*** scheme assumes a total volumetric content.
- The soil moisture from the ***RUC LSM*** provides the amount of moisture in excess of a specified point for that soil category.
- ***Mixing*** Noah input and the RUC selection in the model (or vice versa), requires that adjustments are made to the soil moisture arrays to account for total and residual amounts.

Tasks of the Initialization Program
Consistency Checks

- Both the static and the first-guess fields can provide information for ***land use*** and for ***soil texture***.
- Static: 30 sec resolution, fractional values (24 USGS land use / vegetation type, 16 soil texture categories), not consistent with soil moisture field
- First-Guess: the resolution of the data file, dominant category, but consistent with the soil moisture field
- ***User selects*** which to provide to the WRF model at run-time

Tasks of the Initialization Program
Soil Fields

- Fields: soil temperature, soil moisture, soil liquid (ARW: for the Noah scheme, set to zero, then reinitialized in model based on soil moisture and soil temp)
- ***Vertically interpolated*** to the levels required by the specified surface physics option from the namelist file
- At least two vertical levels must be provided from the WPS that surround the output levels requested (for manufactured sea ice, a skin temperature and the SST threshold are linearly interpolated)
- ***Schemes***: simple diffusion (5 layers, temperature only), Noah (4 layers), RUC (6 levels), Pleim-Xiu (2 levels)
- The ***different number of levels*** is why the real program is re-run when the surface layer is changed in the model

Tasks of the ARW Initialization Program
3D Time Dependent Data ***from WPS***

- The 3d fields are vertically interpolated to the η surfaces
- SLP, topo, T, Qv, Z used to compute total surface p
- Remove moisture in column of input fields for dry pressure
- User specifies the selected η surfaces in the namelist
- Dry surface pressure to compute target WRF coordinates
- Vertically interpolate input fields in dry pressure

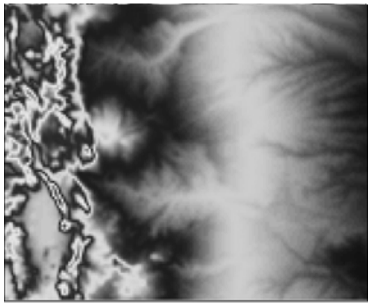
Tasks of the ARW Initialization Program
3D Time Dependent Data ***from WPS***

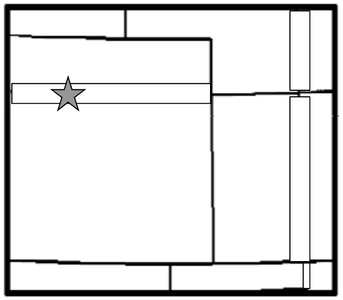
- A number of vertical interpolation options are available to users
- The options can have a significant impact on the initial conditions passed to the model
- More information is contained in the info file ***README.namelist*** in the ***run*** directory
- Options are located in the ***&domains*** namelist record of ***namelist.input***

Tasks of the ARW Initialization Program
3D Time Dependent Data ***from WPS***

- Impact: *Expected region of changes*
- *Non-standard setting*
- Which level is being viewed

- Topography and domain for difference plots, 160x140, 4 km, input = 40 km NAM



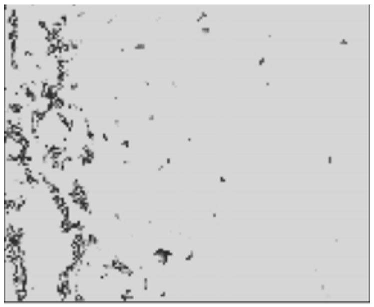


Tasks of the ARW Initialization Program 3D Time Dependent Data ***from WPS***

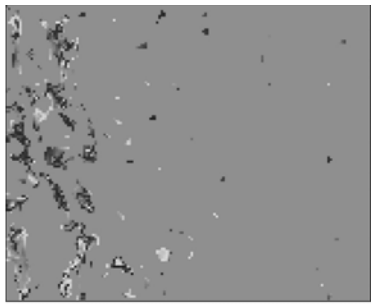
- Impact: few lowest levels only
- force_sfc_in_vinterp = 0
- η level 1

η level 1

- Theta (-8 K blue, 0 K yellow)



- U (-3 m/s blue, 2 m/s red)

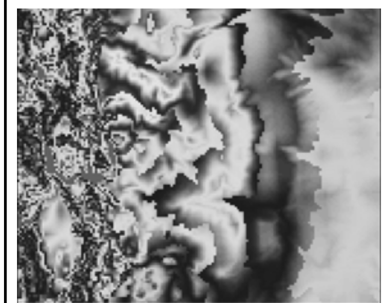


Tasks of the ARW Initialization Program 3D Time Dependent Data ***from WPS***

- Impact: few lowest levels only
- force_sfc_in_vinterp = 6
- η level 4

η level 1

- Theta (0 K blue, 10 K red)



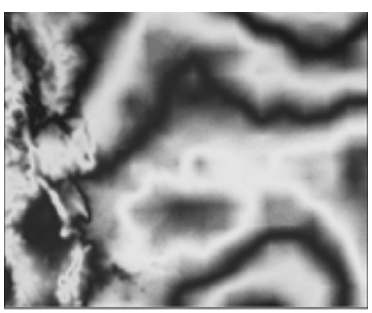
- U (-5 m/s blue, 6 m/s red)

Tasks of the ARW Initialization Program 3D Time Dependent Data ***from WPS***

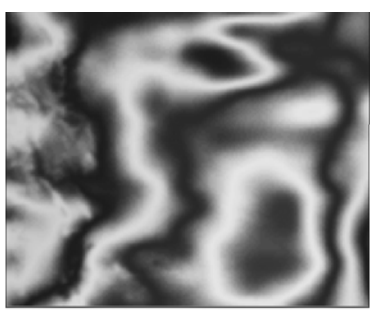
- Impact: above first 4 levels, most near tropopause
- lagrange_order = 2
- η level TOP

η level 1

- Theta (0.7 K blue, 1.6 K red)



- U (0.4 m/s blue, 1.4 m/s red)

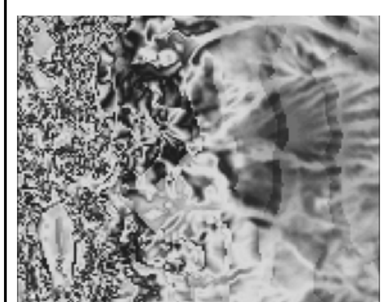


Tasks of the ARW Initialization Program 3D Time Dependent Data ***from WPS***

- Impact: lowest level only
- lowest_lev_from_sfc = T
- η level 1

η level 1

- Theta (-10 K blue, 8 K red)



- U (-3 m/s blue, 7 m/s red)

Tasks of the ARW Initialization Program 3D Time Dependent Data *from WPS*

- Impact: outer few rows and column, amplitude damps upward
- smooth_cg_topo = T
- η level 1

η level 1

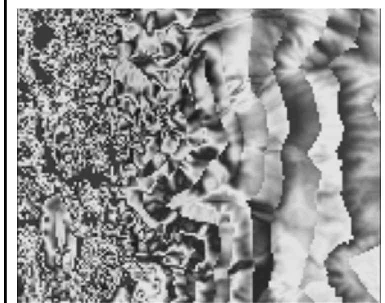
- Theta (-10 K blue, 9 K red)
- U (-6 m/s blue, 6 m/s red)

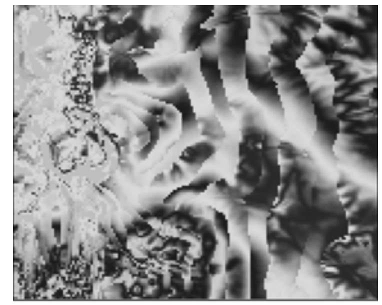
Tasks of the ARW Initialization Program 3D Time Dependent Data *from WPS*

- Impact: lowest few levels
- use_surface = F
- η level 1

η level 1

- Theta (-11 K blue, 0 K red)
- U (-3 m/s blue, 4 m/s red)





Tasks of the ARW Initialization Program 3D Time Dependent Data *from WPS*

- All variables are on the correct horizontal staggering: U, V, relative humidity, temperature, height
- U, V, Qv (diagnosed) pass through without any modification (other than vertical interpolation for WPS input)
- Other moisture species (cloud water, snow, rain, graupel, cloud ice) are available as input, but require the compile-time use of a ***-DRUC_CLOUD*** cpp directive; initialized, not used in lateral boundary file)
- Potential temperature has constant factor removed (300 K) for numerical round-off purposes (***looks like Celsius*** near surface, be careful)

Tasks of the NMM Initialization Program 3D Time Dependent Data *from WPS*

- Make sure input data is vertically ordered as expected, limit hybrid topography deviation
- Input 3-D pressure and T, topo, Z used to compute total surface pressure
- Compute target vertical coordinate, total surface pressure through dp/dz, 3d pressure
- User specifies the selected σ surfaces in the namelist (or can be computed)
- Vertically interpolate input fields in pressure to the σ surfaces: T linearly in pressure; mixing ratio linearly in log(pressure); u and v linear (then an adjustment if using a hybrid input source)

Tasks of the NMM Initialization Program 3D Time Dependent Data ***from WPS***

- All input variables are on the correct horizontal staggering: u, v, RH, temperature, *etc*.
- u, v pass through without any modification (other than vertical interpolation for WPS input)
- Specific humidity diagnosed from relative humidity
- Monthly values greenness fraction and albedo interpolated to a specific date
- Adjust albedo for sea-ice and soil moisture fields over water, and snow cover and snow depth over land

Tasks of the ARW Initialization Program Base State

- Mass coordinate (ARW WRF model's computational surface) is reference pressure based, surfaces move up and down in pressure space
- Base state surface pressure is a function of terrain elevation plus several user supplied constants
  - Base surface pressure => base 3D pressure
  - Base 3D pressure => base 3D potential temperature
  - Base 3D pressure and potential temperature => base inverse density
  - Base inverse density integrated up => geopotential
- Base state computations follow the model's definition of the equation of state and the hydrostatic relation

Tasks of the NMM Initialization Program Surface Level, Projection, Boundaries

- Compute and analytically define ground temperature
- Sort SST to be only over water and skin temperature to be only over land points
- Set "soil" temperature for sea ice and water points to fixed constants
- Compute roughness height based on land mask and elevation
- Compute projection constants: Coriolis, grid distance
- Increase diffusion along lateral boundaries

Real-Data Initialization – ARW & NMM

- Definition of Terms
- Purpose and Tasks of Initialization Program
- Files before and after

Purpose of the Initialization Program
Input Files for the WRF Model

- Provide ***initial condition*** data from the WPS to the WRF model (if ARW, then possibly for multiple domains)
- Compute ***lateral boundary*** conditions for outer-most grid

- ARW optional file: ***lower boundary file*** with time dependent sea-surface temperature and sea ice
- ARW optional file: ***grid nudging*** requires multiple time periods of data in the initial condition format

- ARW: output from the real.exe program is suitable to be used as input to the WRF Var package for a "cold start"

Tasks of the Initialization Program
Boundary Output Fields to WRF

- ARW: ***couple*** momentum with total dry column pressure and map factors for use in ***lateral boundary*** values and tendencies
- ARW: geopotential, potential temperature, and moisture (Qv only) are coupled with total dry column pressure for boundary conditions

- NMM: pressure, u, v, T, specific humidity, cloud water, TKE are the boundary output fields

Tasks of the Initialization Program
Output Fields to WRF

- Boundary tendencies are linear ***differences*** valid between the bounding times provided from the WPS data's temporal availability
- The lateral boundaries are arrays for each of the four domain sides; defined for the entire length of the side, the entire height (for 3D arrays)

- ARW: several rows/columns (user defined)
- NMM: one row and column

- ***One less boundary time*** period is created than time periods of WPS data processed

Tasks of the Initialization Program
ARW Nest Domains

Loop over model ***domains***
    Loop over time ***periods***
        Input Data from WPS
        Process Data (consistency, base state, perturbation calculations)
        If time loop = 1 => output IC
        If time loop = 1 & domain loop > 1 => exit time loop
        If time loop >1 => couple data, output BC
    End time period loop
End model domain loop

Tasks of the ARW Initialization Program
ARW Nest Domains

- Must have ***WPS input*** data for each nested domain to be initialized by real.exe (the model can horizontally interpolate domains)
- No inter-domain consistency checks, handled by the model during feedback steps
- No horizontal interpolation from the parent domain to the child domain
- Fine domains are only processed at the ***first time*** provided from the WPS by default (except during grid nudging, or run-time namelist request)
- ***User specifies*** 1) which domains to process and 2) that an additional input file is being supplied

Required Input Files

- Simple data checks: times, dims, grid distance, model top
- Physics options are infrequently impacted by WPS output EXCEPT – the real program must be re-run when ***changing the surface physics option*** in the WRF model

Generated Output Files
ARW optional: *Lower Boundary File*

- An optional file that is available for output is the lower boundary condition file
- Contains time dependent ***sea-surface temperature*** and ***sea ice***
- Values are provided, ***no tendencies***
- The ***temporal resolution*** is the same as for the lateral boundary file
- Useful typically for ***long model runs***, such as where a static sea-surface temperature is an invalid assumption

WRF & WPS:
Compile

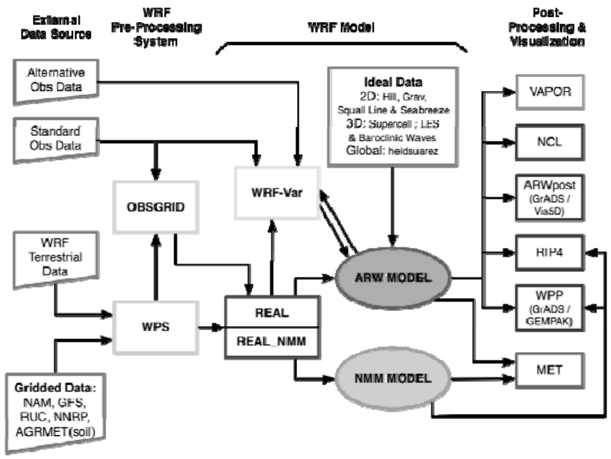
WRF and WPS: Compile

Laurie Carson
National Center for Atmospheric Research (NCAR)
The Developmental Testbed Center (DTC)

26th January, 2009

1

Program Flow



*First need to compile WRF and then WPS

2

System Requirements

Required libraries (WRF and WPS):

- FORTRAN 90/95 compiler
- C compiler
- Perl
- netCDF
- NCAR Graphics (optional, but recommended – used by graphical utility programs)

Optional libraries* for GRIB2 support (WPS):

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

*Installation of these libraries is not part of the WPS installation script

3

Installing WRF

- Download source code
- Set environment
- Configure and Compile WRF

4

Download WRF Source Code

- The WRF source code can be obtained from:
  http://www.mmm.ucar.edu/wrf/users/download/get_source.html
  - Click 'New Users', register and download, or
  - Click 'Returning User', enter email and download
- Both the ARW and NMM cores are included in:

  WRFV3.TAR.gz *(or the latest release available)*
- After ***gunzip*** and ***untar***, should see a directory **WRFV3/**

  tar –zxvf WRFV3.0.TAR.gz
- ***cd*** to **WRFV3/** directory

5

WRFV3 Directory

	Makefile	**Top-level makefile**
	README	**General information about WRF code**
	README.NMM	**NMM specific information**
	README_test_cases	**Explanation of the test cases for ARW**
Data dictionary →	**Registry/**	**Directory for WRF Registry file**
Compile rules →	**arch/**	**Directory where compile options are gathered**
Compile scripts →	**clean**	**script to clean created files and executables**
	compile	**script for compiling WRF code**
	configure	**script to configure the configure.wrf file for compile**
Source code directories	**dyn_em**	**Directory for ARW dynamic modules**
	dyn_exp/	**Directory for a 'toy' dynamic core**
	dyn_nmm/	**Directory for NMM dynamic modules**
	external/	**Directory that contains external packages, such as those for IO, time keeping and MPI**
	frame/	**Directory that contains modules for WRF framework**
	inc/	**Directory that contains include files**
	main/	**Directory for main routines, such as wrf.F, and all executables**
	phys/	**Directory for all physics modules**
	share/	**Directory that contains mostly modules for WRF mediation layer and WRF I/O**
	tools/	**Directory that contains tools**
Run directories →	**run/**	**Directory where one may run WRF**
	test/	**Directory containing sub-directories where one may run specific configurations of WRF.**

6

Set environment

- If the ***netCDF*** is not in the standard ***/usr/local*** then set the **NETCDF** environment variable before typing '***./configure***':

  Example: ***setenv NETCDF /usr/local/netcdf-pgi***
- WRF needs both the ***lib*** and ***include*** directories
- As a general rule for LINUX systems, make sure the ***netCDF*** and ***MPI*** libraries are installed using the same compiler (PGI, Intel, g95) that will be used to compile WRF.

7

Set environment, cont.

- Most of these settings are not required, but if difficulties are encountered you may want to try:
  - ***unset limits***
    - Especially if you are on a small system
  - ***setenv MP_STACK_SIZE 64000000***
    - OpenMP blows through the stack size, set it large
  - ***setenv OMP_NUM_THREADS n*** (where ***n*** is the number of processors to use)
    - For systems with OpenMP installed, this is how the number of threads is specified
  - ***setenv MPICH_F90 f90*** (Or whatever FORTRAN compiler may be called)
    - WRF needs the ***bin***, ***lib*** and ***include*** directories

8

Configuring WRF

- To create a WRF configuration file for your computer, type:

  ./configure

- This script checks the system hardware and software (mostly *netCDF*), and then offers the user choices for configuring WRF:
  - Type of compiler
  - Serial, OpenMP, or MPI
  - Type of nesting (basic, preset moves, vortex following)

9

List of Configure Options - I

<u>Choices for 32-bit LINUX operated machines are:</u>

1. Linux i486 i586 i686, gfortran compiler with gcc (serial)
2. Linux i486 i586 i686, gfortran compiler with gcc (smpar)
3. Linux i486 i586 i686, gfortran compiler with gcc (dmpar)
4. Linux i486 i586 i686, gfortran compiler with gcc (dm+sm)
5. Linux i486 i586 i686, g95 compiler with gcc (serial)
6. Linux i486 i586 i686, g95 compiler with gcc (dmpar)
7. Linux i486 i586 i686, PGI compiler with gcc (serial)
8. Linux i486 i586 i686, PGI compiler with gcc (smpar)
9. Linux i486 i586 i686, PGI compiler with gcc (dmpar)
10. Linux i486 i586 i686, PGI compiler with gcc (dm+sm)
11. Linux x86_64 i486 i586 i686, ifort compiler with icc (serial)
12. Linux x86_64 i486 i586 i686, ifort compiler with icc (smpar)
13. Linux x86_64 i486 i586 i686, ifort compiler with icc (dmpar)
14. Linux x86_64 i486 i586 i686, ifort compiler with icc (dm+sm)
15. Linux i486 i586 i686 x86_64, PathScale compiler with pathcc (serial)
16. Linux i486 i586 i686 x86_64, PathScale compiler with pathcc (dmpar)

10

List of Configure Options - II

<u>Choices for IBM machines are:</u>

1. AIX xlf compiler with xlc (serial)
2. AIX xlf compiler with xlc (smpar)
3. AIX xlf compiler with xlc (dmpar)
4. AIX xlf compiler with xlc (dm+sm)

11

List of Configure Options - III

<u>Choices for Nesting are:</u>

0. no nesting (only available for serial and smpar)
1. basic
2. preset moves
3. vortex following

- default is option 0 for serial/smpar, 1 for dmpar
- in addition, if running NMM with nesting:

  setenv WRF_NMM_NEST 1

12

Configuring WRF, cont.

- The ***./configure*** command will create a file called ***configure.wrf***
  - This file contains compilation options, rules, etc. specific to your computer and can be edited to change compile options, if desired.
- WRFV3 compile options are provided for a number of platforms. In addition, the ***arch/configure_new.defaults*** file can be edited to add a new option if needed.

13

Configuration File

- The ***configure.wrf*** file is built from three pieces within the ***arch*** directory

1. **preamble_new**: uniform requirement for the code, such as maximum number of domains, word size, etc.
2. **configure_new.defaults**: selection of compiler, parallel, communication layer
   - User edits if a change to the compilation options or library locations is needed
3. **postamble_new**: standard make rules and dependencies

14

Sample *configure.wrf*

```
# Settings for Linux i486 i586 i686, PGI compiler with gcc  (dmpar)

DMPARALLEL  =   1
SFC         =   pgf90
SCC         =   gcc
DM_FC       =   mpif90 -f90=$(SFC)
DM_CC       =   mpicc -cc=$(SCC) -DMPI2_SUPPORT
FC          =   $(DM_FC)
CC          =   $(DM_CC) -DFSEEKO64_OK
LD          =   $(FC)
RWORDSIZE   =   $(NATIVE_RWORDSIZE)
FCOPTIM     =   -O2 -fast
FCNOOPT     =   -O0
FCDEBUG     =   # -g $(FCNOOPT)
```

15

Compiling WRF

- First set ***one*** core environment variable to 1:

  ARW: ***setenv WRF_EM_CORE 1***

  NMM: ***setenv WRF_NMM_CORE 1***

Note: If neither of these environment variables are set, the default is to compile ARW.

In addition, if running NMM with nesting:

setenv WRF_NMM_NEST 1

16

Compiling WRF

- Type the following command to compile:

  ./compile test_case >& compile_wrf.log

  where ***test_case*** is one of the following:

compile em_b_wave ***compile em_quarter_ss*** ***compile em_heldsuarez*** ***compile em_les***	***3D Ideal Case (ARW only)***
compile em_grav2d_x ***compile em_hill2d_x*** ***compile em_squall2d_x*** ***compile em_squall2d_y*** ***compile em_seabreeze_x***	***2D Ideal Case (ARW only)***
compile em_real ***compile nmm_real***	***Real Data Cases (ARW and NMM)***
compile -h →	***help message***

17

Compiling ARW: Idealized Cases

- If the choosen ideal case compilation is successful, it will create two executables under **main/**:
  - ✓***ideal.exe***: used for ARW initialization of ideal cases.
  - ✓***wrf.exe***: used for ARW model integration.
- These executables will be linked to the specific **test/em_*test_case*** and **run** directories.

18

Compiling WRF: Real Data Case

- If the real data case compilation is successful:
  - ARW: creates four executables in the ***main/*** directory:
    - ✓***real.exe***: used for initialization of real data cases.
    - ✓***wrf.exe***: used for model integration.
    - ✓***ndown.exe:*** used for one-way nesting
    - ✓***nup.exe*** (not used much)
  - NMM: creates two executables in the ***main/*** directory:
    - ✓***real_nmm.exe***: used for initialization of real data cases.
    - ✓***wrf.exe***: used for model integration.
- These executables will be linked to either ***test/em_real*** or ***test/nmm_real*** and ***run/*** directories.

19

Clean Compilation

- To remove all object files (except those in *external/*) and executables, type:

  clean

- To remove all built files, including ***configure.wrf***, type:

  clean –a

  - ➢ Recommended if
    - compilation failed
    - registry changed
    - want to compile different dynamic core
    - want to change configuration file

20

Compiling both WRF cores

Using two different WRFV3 directory trees

Set environment variables for each and configure and compile as usual

Using the same WRFV3 directory tree

Core "A"

- Set environment
- Configure, compile
- Save ***main/wrf.exe*** to ***main/wrf_coreA.exe***
- Copy ***main/*exe*** to a temporary location outside of WRFV3/

clean –a

Core "B"

- Set environment
- Configure, compile
- Save ***wrf.exe*** to ***wrf_coreB.exe***

Move Core "A" ***exe's** from temporary location back to ***WRFV3/main*** (and to ***test/test_case/*** if you run there)

21

Installing WPS

- Download static terrestrial data
- Download source code
- Configure and Compile WPS

Reminder: A successful compilation of WRF is required prior to WPS compilation!

22

Download Static Terrestrial Data

- The terrestrial fields interpolated by ***geogrid*** may be downloaded from same page as the code:

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

  – Two options for data: low-res and all resolutions

- Data are static: only need to be downloaded once

23

Download Static Terrestrial Data, Cont.

- The *geog.tar.gz* file (all resolutions) contains:
  - *albedo_ncep* – monthly surface albedo
  - *greenfrac* – monthly vegetation fraction
  - *islope* – slope index
  - *landuse* – land use category (30", 2', 5', and 10' res.)
  - *maxsnowalb* – maximum snow albedo (30", 2', 5', and 10' res.)
  - *soiltemp* – annual mean deep soil temperature (30", 2', 5', and 10' res.)
  - *soiltype_bot* – bottom-layer soil type (30", 2', 5', and 10' res.)
  - *soiltype_top* – top-layer soil type (30", 2', 5', and 10' res.)
  - *topo* – topography height (30", 2', 5', and 10' res.)

24

Download Static Terrestrial Data, Cont.

- Uncompress the data into a directory with ~10 GB of available space (264 MB for low-res only)!
  tar -zxvf geog.tar.gz

- Data can be shared by users on the same machine by placing files in a common directory
  – Recommended due to size!

25

Download WPS Source Code

- The WPS source code can be obtained from:
  http://www.mmm.ucar.edu/wrf/users/download/get_source.html

- WPS is designed to work with WRF (since v2.2 for ARW and v2.2.1 for NMM)
  – *WPS programs use WRF I/O API libraries to do file input and output*
  – *These I/O libraries are built when WRF is installed*

- For simplicity, install WPS/ in the same location as WRFV3/

- After ***gunzip*** and ***untar***, should see a directory WPS/
  tar –zxvf WPSV3.TAR.gz *(or the latest release available)*

  ls
  WPS/   WRFV3/

- ***cd*** to WPS/ directory

26

Configure WPS

- To create a WPS configuration file for your computer, type:
  ./configure

- This script offers the user choices for configuring WPS:
  - Type of compiler
  - Serial or Distributed memory
  - GRIB1 or GRIB2

- The *./configure* command will create a file called ***configure.wps***

  – This file contains compilation options, rules, etc. specific to your computer and can be edited to change compile options, if desired.

27

List of WPS Configure Options

```
Will use NETCDF in dir: /usr/local/netcdf-pgi
    $JASPERLIB or $JASPERINC not found in environment, configuring to build without
    grib2 I/O...
    ------------------------------------------------------------------------
Please select from among the following supported platforms.

      1.  PC Linux i486 i586 i686, PGI compiler      serial, NO GRIB2
      2.  PC Linux i486 i586 i686, PGI compiler      serial
      3.  PC Linux i486 i586 i686, PGI compiler      DM parallel, NO GRIB2
      4.  PC Linux i486 i586 i686, PGI compiler      DM parallel
      5.  PC Linux i486 i586 i686, Intel compiler    serial, NO GRIB2
      6.  PC Linux i486 i586 i686, Intel compiler    serial
      7.  PC Linux i486 i586 i686, Intel compiler    DM parallel, NO GRIB2
      8.  PC Linux i486 i586 i686, Intel compiler    DM parallel
      9.  PC Linux i486 i586 i686, g95 compiler,     serial, NO GRIB2
     10.  PC Linux i486 i586 i686, g95 compiler,     serial

    Enter selection [1-10] : 1
----------------------------------------------------------
Configuration successful. To build the WPS, type: compile
----------------------------------------------------------
```

28

Compile WPS

- If configuration was successful, compile WPS:
  ./compile >& compile_wps.log
- If the compilation is successful, it will create three executables:
  - ✓***geogrid.exe***: define size/location of domain(s)
  - ✓***ungrib.exe***: extract meteorological fields from GRIB files
  - ✓***metgrid.exe***: horizontally interpolate meteorological fields (from *ungrib*) to simulation grid(s) (defined by *geogrid*)

29

Compile WPS, Cont.

- If compilation is successful, it will create the following executables in ***util/***:
  - ✓ ***avg_tsfc.exe***
  - ✓ ***g1print.exe***
  - ✓ ***g2print.exe***
  - ✓ ***mod_levs.exe***
  - ✓ ***rd_intermediate.exe***
  - ✓ ***calc_ecmwf_p.exe***
- If NCAR Graphics libraries are available it will also create in ***util/***:
  - ✓ ***plotgrids.exe***
  - ✓ ***plotfmt.exe***

  – Each of these utilities are described in more detail in the WPS Overview talk

30

Sharing WPS Installation

- A single build of WPS will work for both ARW and NMM core
- Multiple users may share a single installation of the WPS; not every user needs to install
  – Make WPS installation directory read-only
  – Each user will run WPS programs in their own working directories
  – Output files created in user working directories

31

Additional Resources

- For more detailed information on installation of WRF and WPS, please see:
  – ARW and NMM Users Guides

  – Online Users Pages:
    - **ARW:** *http://www.mmm.ucar.edu/wrf/users/*
    - **NMM:** *http://www.dtcenter.org/wrf-nmm/users/*
- For further assistance regarding WRF and WPS:
  – WRF Users Forum: *http://forum.wrfforum.com*
  – WRF Email list: wrf_users@ucar.edu

  – WRF Help email: *wrfhelp@ucar.edu*

32

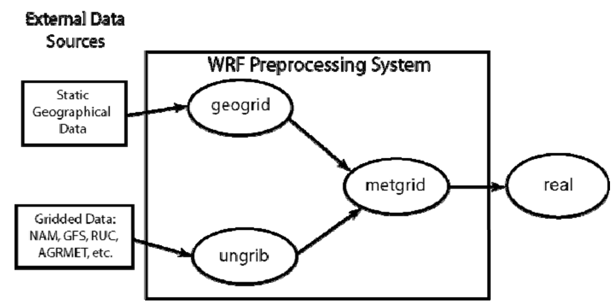
WPS: Set up & Run

Running the WRF Preprocessing System

Michael Duda

Review

- Briefly recall the programs in the WPS



Review

- geogrid (think <u>geo</u>graphical)
  - Define size/location of model domains and interpolate static terrestrial fields to simulation grids
- ungrib (think <u>un</u>+<u>grib</u>)
  - Extract meteorological fields from GRIB files
- metgrid (think <u>met</u>eorological)
  - Horizontally interpolate meteorological fields (from ungrib) to simulation grids (defined by geogrid)

Overview

- How to run through the WPS for basic cases
  - Basic steps for running WPS
    - Geogrid
    - Ungrib
    - Metgrid
- WPS utility programs
- Common WPS mistakes

Running geogrid

STEP 1: Edit `namelist.wps`

For geogrid, only the &share and &geogrid namelists need to be edited in namelist.wps

```
&share
 wrf_core = 'ARW',
 max_dom = 2,
 io_form_geogrid = 2,
/

&geogrid
 parent_id         =   1,    1,
 parent_grid_ratio =   1,    3,
 i_parent_start    =   1,   20,
 j_parent_start    =   1,   17,
 e_we              = 220,  181,
 e_sn              = 175,  181,
 geog_data_res     = '5m',  '2m',
 dx = 15000,
 dy = 15000,
 map_proj   = 'lambert',
 ref_lat    = 37.0,
 ref_lon    = -97.0,
 truelat1   = 45.0,
 truelat2   = 30.0,
 stand_lon  = -97.0,
 geog_data_path = '/data/static/geog/'
/
```


Running geogrid

STEP 1: Edit `namelist.wps`

```
&share
 wrf_core = 'ARW',
 max_dom = 2,
 io_form_geogrid = 2,
/
```

- wrf_core: Which WRF core? For ARW, set to 'ARW' For NMM, set to 'NMM'
- max_dom: Total number of model domains, including nests, for ARW; number of nesting levels for NMM.
- io_form_geogrid: Format for geogrid output files; 2=netCDF is recommended.

See p. 3-8 and 3-31

Running geogrid

STEP 1: Edit `namelist.wps`

```
&geogrid
 parent_id         =  1,   1,
 parent_grid_ratio =  1,   3,
 i_parent_start    =  1,  20,
 j_parent_start    =  1,  17,

 e_we              = 220,
181,
 e_sn              = 175,  181,
 dx = 15000,
 dy = 15000,

 geog_data_res     = '5m',  '2m',
 . . .
/
```

- **Nesting**: Who is the parent? What is the grid ratio for each nest? Where is it located in its parent?
- **Domain sizes**: How many grid points does the domain have? What is the grid spacing?
- **Static data**: What resolution of source data to interpolate from for each domain? '30s', '2m', '5m', or '10m'?

See p. 3-9, 3-16, and 3-33

Running geogrid

STEP 1: Edit `namelist.wps`

```
&geogrid
 …
 map_proj   = 'lambert',
 ref_lat    = 37.0,
 ref_lon    = -97.0,
 truelat1   = 45.0,
 truelat2   = 30.0,
 stand_lon  = -97.0,

 geog_data_path = '/data/static/geog/'
/
```

- **Map projection**: What projection to use? What are the parameters of the projection?

See p. 3-9 and 3-33

- **Static data**: Where are the data directories (e.g., topo_30s) located?

See p. 3-35

Running geogrid

STEP 2: Make sure GEOGRID.TBL is linked to the correct version of GEOGRID.TBL

- There are multiple GEOGRID.TBL files to support multiple dynamical cores in WRF
- GEOGRID.TBL.ARW must be used for ARW
- GEOGRID.TBL.NMM must be used for NMM

```
> ls geogrid/GEOGRID.TBL
GEOGRID.TBL -> GEOGRID.TBL.ARW
```


Running geogrid

STEP 3: Run geogrid.exe

```
Parsed 11 entries in GEOGRID.TBL
Processing domain 1 of 2
  Processing XLAT and XLONG
  Processing MAPFAC
  Processing F and E
  Processing ROTANG
  Processing LANDUSEF
  Calculating landmask from LANDUSEF
  Processing HGT_M

  …

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Successful completion of geogrid.        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Geogrid processes each domain individually. There will be one section of messages for each domain or nesting level.

As each field is processed, a message will be written to the screen and to the geogrid.log file.

Running geogrid

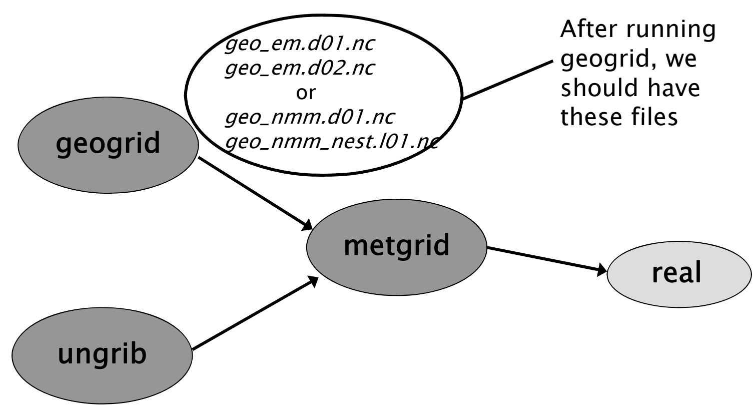
STEP 4: Check that geogrid ran successfully

If geogrid ran sucessfully, this message should be printed:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Successful completion of geogrid.        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

If there was an error, check for an ERROR or WARNING message in the `geogrid.log` file, or for a system error, like "Segmentation fault".

Running geogrid



Running ungrib

STEP 1: Edit `namelist.wps`

For ungrib, only the &share and &ungrib namelists need to be edited

```
&share
   wrf_core = 'ARW',
   max_dom = 2,
   start_date  = '2006-04-01_00:00:00',
   end_date   = '2006-04-01_12:00:00',
   interval_seconds = 21600
   io_form_geogrid = 2,
/
```

```
&ungrib
   out_format = 'WPS',
   prefix = 'GFS',
/
```


Running ungrib

STEP 1: Edit `namelist.wps`

```
&share
   wrf_core = 'ARW',
   max_dom = 2,

   start_date  = '2006-04-01_00:00:00',
   end_date   = '2006-04-01_12:00:00',

   interval_seconds = 21600

   io_form_geogrid = 2,
/
```

Data time range: Between which times should ungrib process GRIB data?

Data frequency: How many seconds between output files for ungrib? E.g., 10800 s = 3 hrs

See p. 3-11, and 3-32

Running ungrib

STEP 1: Edit `namelist.wps`

```
&ungrib
   out_format = 'WPS',

   prefix = 'GFS',
/
```

Intermediate file format: Which format to use for intermediate files? 'WPS', 'SI', or 'MM5' are possible; 'WPS' is recommended.

Intermediate file names: Gives prefix for intermediate files.
Prefix can include a path.
E.g., 'XZY' would give intermediate files named XYZ:*yyyy-mm-dd_hh.*

See p. 3-11, 3-18, and 3-36

Running ungrib

STEP 2: Link the correct Vtable to the file name "Vtable" in the run directory

- Some Vtables are provided with WPS in the **`WPS/ungrib/Variable_Tables`** directory
  - E.g., Vtable.GFS, Vtable.SST, Vtable.ECMWF
- Ungrib always expects to find a file named **`Vtable`** in the run directory

See p. 3-12

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

Running ungrib

STEP 3: Link GRIB files to the correct file names in the run directory

- Ungrib always expects GRIB files to be named `GRIBFILE.AAA`, `GRIBFILE.AAB`, `GRIBFILE.AAC`, etc., in the run directory
- The `link_grib.csh` script can be used to link GRIB files to these file names:

```
> link_grib.csh /data/GRIB/GFS/gfs*
> ls GRIBFILE.*
GRIBFILE.AAA -> /data/GRIB/GFS/gfs_060401_00_00
```

See p. 3-12

Running ungrib

STEP 4: Run ungrib.exe

```
 *** Starting program ungrib.exe ***
Start_date =  2006-08-16_12:00:00 ,      End_date = 2006-08-16_12:00:00
output format is WPS
Path to intermediate files is ./
 ungrib - grib edition num           2

#############################################################
Inventory for date = 2006-08-16 12:00:00

PRES   TT       UU       VV       RH       HGT
-------------------------------------------------------------
2013.0  O        O        O        O        O        O
2001.0  X        X        X        X        O        X
1000.0  X        X        X        X        X
 975.0  X        X        X        X        X
 950.0  X        X        X        X        X
 925.0  X        X        X        X        X
 900.0  X        X        X        X        X
```


Running ungrib

STEP 5: Check that ungrib ran successfully

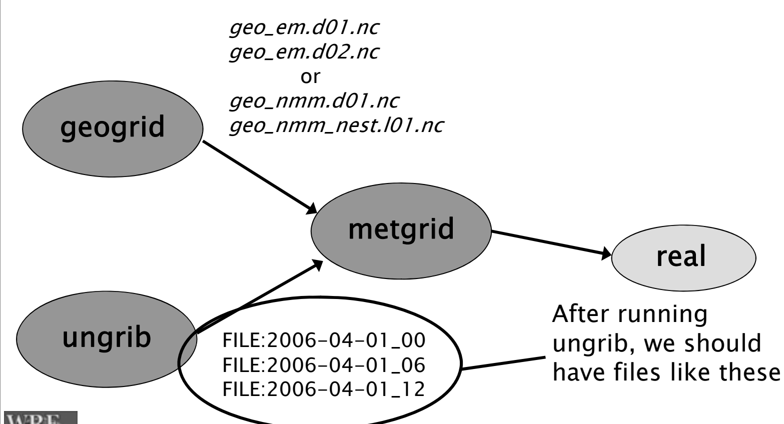
If ungrib ran successfully, this message should be printed:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Successful completion of ungrib.            !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

If there was an error, check for error message in ungrib's printout or in the `ungrid.log` file.

Common errors are related to incorrect date specifications in the `&share` namelist, or because GRIB2 data was used with a version of WPS compiled without GRIB2 libraries.

Running ungrib



Running metgrid

STEP 1: Edit `namelist.wps`

For metgrid, only the &share and &metgrid namelists need to be edited

```
&share
   wrf_core = 'ARW',
   max_dom = 2,
   start_date  = '2006-04-01_00:00:00',  '2006-04-01_00:00:00',
   end_date   = '2006-04-01_12:00:00',  '2006-04-01_00:00:00',
   interval_seconds = 21600
   io_form_geogrid = 2,
/
```

```
&metgrid
   fg_name = 'GFS',
   constants_name = 'SST:2006-04-01_00',
   io_form_metgrid = 2,
/
```


Running metgrid

STEP 1: Edit `namelist.wps`

```
&share
   wrf_core = 'ARW',
   max_dom = 2,

   start_date  = '2006-04-01_00:00:00',  '2006-04-01_00:00:00',
   end_date   = '2006-04-01_12:00:00',  '2006-04-01_00:00:00',

   interval_seconds = 21600
   io_form_geogrid = 2,
/
```

Data time range: Time range to process *for each domain*. Usually, only the initial time is needed for ARW nested domains. Only coarse domain needed for NMM.

See p. 3-14 and 3-32

Running metgrid

STEP 1: Edit `namelist.wps`

```
&metgrid
   fg_name = 'GFS',

   constants_name = 'SST:2006-04-01_00',

   io_form_metgrid = 2,
/
```

Intermediate file prefixes: Prefix (or prefixes) of intermediate files to interpolate to model domain. Should match prefix given to ungrib. *See p. 3-18*

Constant fields: Optional name of an intermediate file with fields to be used for every time period.

Metgrid I/O format: Which I/O format to use for metgrid output? 2=netCDF is recommended.

See p. 3-14, and 3-36

Running metgrid

STEP 2: Make sure METGRID.TBL is linked to the correct version of METGRID.TBL

- There are multiple METGRID.TBL files to support multiple dynamical cores in WRF
- Generally, METGRID.TBL.ARW must be used for ARW and METGRID.TBL.NMM for NMM

```
> ls metgrid/METGRID.TBL
METGRID.TBL -> METGRID.TBL.ARW
```

Running metgrid

STEP 3: Run metgrid.exe

```
Processing domain 1 of 2
    SST:2006-04-01_00

 Processing 2006-04-01_00
    GFS
 Processing 2006-04-01_06
    GFS
 Processing 2006-04-01_12
    GFS

Processing domain 2 of 2
    SST:2006-04-01_00
 Processing 2006-04-01_00
    GFS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Successful completion of metgrid.  !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Fields from constant files (given using `constants_name`) are processed before any time varying fields.

Metgrid processes all time period for one domain before processing for the next domain

Running metgrid

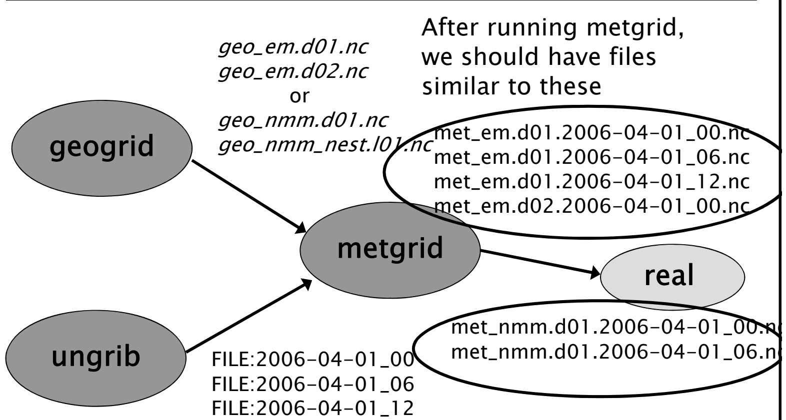
STEP 4: Check that metgrid ran successfully

If metgrid ran successfully, this message should be printed:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Successful completion of metgrid.        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

If there was an error, check for an ERROR or WARNING message in the `metgrid.log` file, or for a system error, like "Segmentation fault".

Running metgrid



Overview

- How to run through the WPS for basic cases
  - Basic steps for running WPS
    - Geogrid
    - Ungrib
    - Metgrid
- WPS utility programs
- Common WPS mistakes

WPS Utility Programs

- Besides geogrid, ungrib, and metgrid, some simple utility programs are distributed with WPS:
  - For checking contents of intermediate format files
  - For listing contents of GRIB1 & GRIB2 files
  - To assist in locating domains
- Some programs use NCAR Graphics libraries for plotting
  - For these utilities, *NCAR Graphics must be installed*

See p. 3-22

WPS Utility Programs

The utility programs that come with WPS can be helpful when diagnosing problems with WPS output

- All utilities are found in the **`WPS/util`** directory

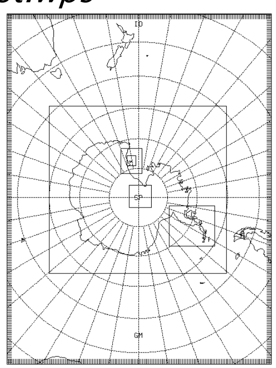
- Users are encouraged to make use of these utilities to examine WPS input and output files

Utility: plotgrids

The *plotgrids* program plots the location of grids defined in *namelist.wps*

- *plotgrids can be used to iteratively refine the locations of grids.*

- *plotgrids* uses the namelist.wps file only, so there is no need to run geogrid first!



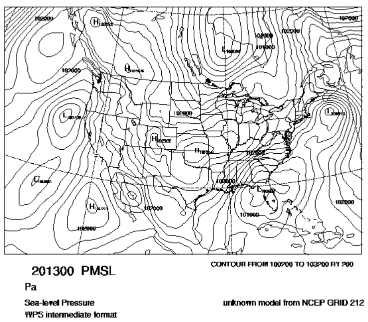
Utility: rd_intermediate

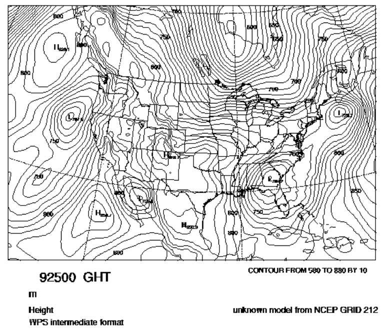
The rd_intermediate lists information about the fields found in an intermediate-format file

```
================================================
FIELD = TT
UNITS = K DESCRIPTION = Temperature
DATE = 2000-01-24_12:00:00 FCST = 0.000000
SOURCE = unknown model from NCEP GRID 212
LEVEL = 200100.000000
I,J DIMS = 185, 129
IPROJ = 1
  REF_X, REF_Y = 1.000000, 1.000000
  REF_LAT, REF_LON = 12.190000, -133.459000
  DX, DY = 40.635250, 40.635250
  TRUELAT1 = 25.000002
DATA(1,1)=295.910950
================================================
```

Utility: plotfmt

The plotfmt program plots the fields in the ungrib intermediate-formatted files





Utility: g1print and g2print

The *g1print* and *g2print* programs list the contents of a GRIB1 or GRIB2 file:

rec num	Prod Disc	Cat	Param num	Lvl code	Lvl one	Lvl two	Name	Time	Fcst hour
1	0	3	5	100	100000	0	HGT	2006-08-16_12:00:00	00
2	0	3	5	100	97500	0	HGT	2006-08-16_12:00:00	00
3	0	3	5	100	95000	0	HGT	2006-08-16_12:00:00	00
4	0	3	5	100	92500	0	HGT	2006-08-16_12:00:00	00
5	0	3	5	100	90000	0	HGT	2006-08-16_12:00:00	00
6	0	3	5	100	85000	0	HGT	2006-08-16_12:00:00	00
7	0	3	5	100	80000	0	HGT	2006-08-16_12:00:00	00
8	0	3	5	100	75000	0	HGT	2006-08-16_12:00:00	00
9	0	3	5	100	70000	0	HGT	2006-08-16_12:00:00	00
10	0	3	5	100	65000	0	HGT	2006-08-16_12:00:00	00

Overview

- How to run through the WPS for basic cases
  - Basic steps for running WPS
    - Geogrid
    - Ungrib
    - Metgrid
- WPS utility programs
- Common WPS mistakes

Common WPS Mistakes

1) All 3-d fields must have same number of levels in metgrid

```
WRF_DEBUG: Warning DIM            4 , NAME
 num_metgrid_levels REDIFINED  by var GHT           27
 26  in wrf_io.F90 line        2347
 ERROR: Error in ext_pkg_write_field
```

- This is usually corrected by ensuring that all 3-d meteorological fields have surface level data
- Try setting debug_level=1000 in &share namelist, and checking metgrid.log for a table showing which fields are on which levels

Common WPS Mistakes

2) When using a regional data set (e.g., NAM), ensure that model domain is completely covered by the data

- The metgrid program will stop if the model domain has grid points that are not covered by data

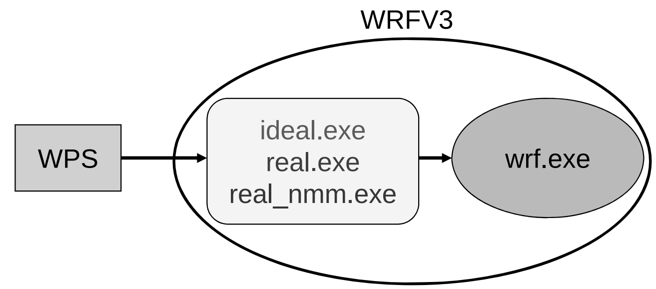
3) For native vertical coordinate data sets (e.g., RUCb, ECMWF), ensure that both pressure and geopotential height fields are available

WRF: Set up & Run

Set Up and Run WRF
(ARW-Ideal, ARW-real, and NMM)

Wei Wang
NCAR/ESSL/MMM

WRF System Flowchart



Outline

- Running WRF code
  - Before you run..
  - Running idealized case
  - Running ARW real-data case
  - Running NMM real-data case
- Basic runtime options for a single domain run (*namelist*)
- Check output
- Simple trouble shooting

This talk is complementary to 'Nesting' talk later.

Before You Run ..

- Check and make sure appropriate executables are created in **`WRFV3/main/`** directory:

  For ARW:
  - **`ideal.exe`**
  - **`real.exe`**
  - **`wrf.exe`**
  - **`ndown.exe`**

  For NMM:
  - **`real_nmm.exe`**
  - **`wrf.exe`**
- If you are running a real-data case, be sure that files from WPS are correctly generated:
  - **`met_em.d01.*,`** for ARW or
  - **`met_nmm.d01.*`** for NMM
- Prepare **`namelist.input`** for runtime options.

WRF test case directories

You have these choices in **`WRFV3/test/`**

(made at compile time):

`nmm_real`	3d real-data	*NMM*
`em_real`		
`em_quarter_ss`	3d ideal	*ARW*
`em_b_wave`		
`em_les`		
`em_heldsuarez`		
`em_hill2d_x`	2d ideal	
`em_squall2d_x`		
`em_squall2d_y`		
`em_grav2d_x`		
`em_seabreeze2d_x`		

Steps to Run

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

WRFV3/run directory

```
README.namelist
LANDUSE.TBL
ETAMPNEW_DATA
GENPARM.TBL
RRTM_DATA
SOILPARM.TBL
VEGPARM.TBL
urban_param.tbl
tr49t67
tr49t85
tr67t85
gribmap.txt
grib2map.tbl
```

…. (a few more)

these files are model physics data files: they are used to either initialize physics variables, or make physics computation more efficient (LANDUSE.TBL through tr67t85)

WRFV3/run directory after compile

```
LANDUSE.TBL
ETAMPNEW_DATA
GENPARM.TBL
RRTM_DATA
SOILPARM.TBL
VEGPARM.TBL
urban_param.tbl
tr49t67
tr49t85
tr67t85
gribmap.txt
grib2map.tbl
namelist.input -> ../test/em_real/namelist.input
real.exe -> ../main/real.exe
wrf.exe -> ../main/wrf.exe
ndown.exe -> ../main/ndown.exe
```

…. (a few more)

An example after ARW real case compile

Running an Idealized Case
ARW only

Running an *Idealized* Case

- If you have compiled an ideal case, you should have:
  `ideal.exe` - ideal case initialization program
  `wrf.exe` - model executable
- These executables are linked to:
  `WRFV3/run`
  and
  `WRFV3/test/`*`em_test-case`*

➔ One can go to either directory to run.

Running an *Idealized* Case

Go to the desired *ideal* test case directory: e.g.

```
cd test/em_quarter_ss
```

If there is **`'run_me_first.csh'`** in the directory, run it first - this links physics data files to the currect directory:

```
./run_me_first.csh
```


Running an *Idealized* Case

Then run the ideal initialization program:

```
./ideal.exe
```

The input to this program is typically a sounding file (file named ***`input_sounding`***), or a pre-defined 3D input (e.g. ***`input_jet`*** in **`em_b_wave`** case).

Running **`ideal.exe`** creates WRF initial condition file: **`wrfinput_d01`**

Note that wrfbdy file is not needed for idealized cases

Running an *Idealized* Case

- To run the model interactively, type
  ./wrf.exe >& wrf.out &
  for single processor (serial) or SMP run. Or
  mpirun -np *N* ./wrf.exe &
  for a MPI run (where ***N*** is the number of processors requested)
- Successful running of the model executable will create a model history file called **wrfout_d01_<*date*>**
  e.g. wrfout_d01_0001-01-01_00:00:00

Mesoscale & Microscale Meteorological Division / NCAR 13

Running an *Idealized* Case

- Edit **namelist.input** file to change options.
- For your own case, you may provide a different sounding.
- You may also edit `dyn_em/module_initialize_<case>.F` to change other aspects of the initialization.

***Note*:**
- For 2D cases and baroclinic wave case, **ideal.exe** must be run serially
- For all 2D cases, **wrf.exe** must be run serially or with SMP

Mesoscale & Microscale Meteorological Division / NCAR 14

Running ARW Real-Data Case

Mesoscale & Microscale Meteorological Division / NCAR 15

Running ARW 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.

Mesoscale & Microscale Meteorological Division / NCAR 16

WRFV3/test/em_*real* directory

```
LANDUSE.TBL -> ../../run/LANDUSE.TBL
ETAMPNEW_DATA -> ../../run/ETAMPNEW_DATA
GENPARM.TBL -> ../../run/GENPARM.TBL
RRTM_DATA -> ../../run/RRTM_DATA
SOILPARM.TBL -> ../../run/SOILPARM.TBL
VEGPARM.TBL -> ../../run/VEGPARM.TBL
urban_param.tbl -> ../../run/urban_param.tbl
tr49t67 -> ../../run/tr49t67
tr49t85 -> ../../run/tr49t85
tr67t85 -> ../../run/tr67t85
gribmap.txt -> ../../run/gribmap.txt
grib2map.tbl -> ../../run/grib2map.tbl
namelist.input      - require editing
real.exe -> ../../main/real.exe
wrf.exe -> ../../main/wrf.exe
ndown.exe -> ../../main/ndown.exe
```

…. (a few more)

Running WRF ARW Real-data Cases

- 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.d01.* .
```


Running WRF ARW Real-data Cases

- Edit **`namelist.input`** file for runtime options (at mininum, one must edit **`&time_control`** for start, end and integration times, and **`&domains`** for grid dimensions)
- Run the real-data initialization program:
  `./real.exe,` if compiled serially / SMP, or
  `mpirun -np N ./real.exe,` for a MPI job
  where ***N*** is the number of processors requested

Running WRF ARW Real-data Cases

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

`wrfinput_d01` ← *Single time level data at model's start time*

`wrfbdy_d01` ← *Multiple time level data at the lateral boundary, and only for domain 1*

Running WRF ARW Real-data Cases

- 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_2005-08-28_00:00:00

  And *restart* file if **restart_interval** is set to a time within the range of the forecast time:
  wrfrst_d01_2008-08-28_12:00:00

Running NMM Real-Data Case

Running NMM Real-Data Case

- If you have compiled the *nmm_real*, you should have:
  real_nmm.exe - NMM real date initialization program
  wrf.exe - NMM model executable

- These executables are linked to:
  WRFV3/run
  and
  WRFV3/test/*nmm_real*

➔ One can go to either directory to run.

WRFV3/test/*nmm_real* directory

```
LANDUSE.TBL -> ../../run/LANDUSE.TBL
ETAMPNEW_DATA -> ../../run/ETAMPNEW_DATA
GENPARM.TBL -> ../../run/GENPARM.TBL
RRTM_DATA -> ../../run/RRTM_DATA
SOILPARM.TBL -> ../../run/SOILPARM.TBL
VEGPARM.TBL -> ../../run/VEGPARM.TBL
urban_param.tbl -> ../../run/urban_param.tbl
tr49t67 -> ../../run/tr49t67
tr49t85 -> ../../run/tr49t85
tr67t85 -> ../../run/tr67t85
gribmap.txt -> ../../run/gribmap.txt
grib2map.tbl -> ../../run/grib2map.tbl
namelist.input       - require editing
real_nmm.exe -> ../../main/real_nmm.exe
wrf.exe -> ../../main/wrf.exe
```

Running WRF NMM Real-data Cases

- One must successfully run WPS, and create **met_nmm.*** file for more than one time period
- Link or copy WPS output files to the run directory:

```
cd test/nmm_real
ln -s ../../../WPS/met_nmm.d01.* .
```


Running WRF NMM Real-data Cases

- Edit **namelist.input** file for runtime options (at minimum, one must edit **&time_control** for start, end and integration time, and **&domains** for grid dimensions)
- Run the real-data initialization program (MPI only):

```
mpirun -np N ./real_nmm.exe
```

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

```
wrfinput_d01
wrfbdy_d01
```


Running WRF NMM Real-data Cases

- Run the model executable by typing (MPI only):

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

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

```
wrfout_d01_2005-08-28_00:00:00
```

And *restart* file if **restart_interval** is set to a time within the range of the forecast time:

```
wrfrst_d01_2008-08-28_12:00:00
```


Basic namelist Options

What is a namelist?

- A Fortran namelist contains a list of *runtime* options for the code 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

&time_control

```
run_days          = 0,
run_hours         = 24,
run_minutes       = 0,
run_seconds       = 0,
start year        = 2000, 2000, 2000,
start_month       = 01,   01,   01,
start_day         = 24,   24,   24,
start_hour        = 12,   12,   12,
start_minute      = 00,   00,   00,
start_second      = 00,   00,   00,
end_year          = 2000, 2000, 2000,
end_month         = 01,   01,   01,
end_day           = 25,   25,   25,
end_hour          = 12,   12,   12,
end_minute        = 00,   00,   00,
end_second        = 00,   00,   00,
interval_seconds  = 21600
history_interval  = 180,  60,   60,
frame_per_outfile = 1000, 1000, 1000,
restart_interval  = 360,
```

domain 1 option

nest options

Notes on **`&time_control`**

- *run_** time variables:
  – Model simulation length: *wrf.exe* and domain 1 only
- *start_** and *end_** time variables:
  – Program *real* will use WPS output between these times to produce lateral (and lower) boundary file
  – They can also be used to specify the start and end of simulation times for the coarse grid.

Notes on **`&time_control`**

- *Interval_seconds*:
  – Time interval between WPS output times, and LBC update frequency
- *history_interval*:
  – Time interval in minutes when a history output is written
  – The time stamp in a history file name is the time when the history file is first written, and multiple time periods may be written in one file. e.g. a history file for domain 1 that is first written for 1200 UTC Jan 24 2000 is

  `wrfout_d01_2000-01-24_12:00:00`

Notes on **`&time_control`**

- *frame_per_outfile*:
  - Number of history times written to one file.
- *restart_interval*:
  - Time interval in minutes when a restart file is written.
  - By default, restart file is not written at hour 0.
  - A restart file contains only one time level data, and its valid time is in its file name, e.g. a restart file for domain 1 that is valid for 0000 UTC Jan 25 2000 is

`wrfrst_d01_2000-01-25_00:00:00`

`&time_control`

```
io_form_history     = 2,
io_form_restart     = 2,
io_form_input       = 2,
io_form_boundary    = 2,
debug_level         = 0,
```

IO format options:
= 1, binary
= 2, netcdf (most common)
= 4, PHDF5
= 5, Grib 1
=10, Grib 2

`io_form_restart = 102` : write in patch sizes: fast for large grids and useful for restart file

Debug print control: Increasing values give more prints.

Notes on **`&time_control`**

- *frame_per_outfile*:
  - Number of history times written to one file.
- *restart_interval*:
  - Time interval in minutes when a restart file is written.
  - By default, restart file is not written at hour 0.
  - A restart file contains only one time level data, and its valid time is in its file name, e.g. a restart file for domain 1 that is valid for 0000 UTC Jan 25 2000 is

`wrfrst_d01_2000-01-25_00:00:00`

`&domains`

```
time_step              = 180
time_step_fract_num    = 0,
time_step_fract_den    = 1,
max_dom                = 1,
s_we                   = 1,     1,     1,
e_we                   = 74,    112,   94,
s_sn                   = 1,     1,     1,
e_sn                   = 61,    97,    91,
s_vert                 = 1,     1,     1,
e_vert                 = 28,    28,    28,
num_metgrid_levels     = 21
dx                     = 30000, 10000, 3333,
dy                     = 30000, 10000, 3333,
eta_levels             = 1.0,0.996,0.99,0.98,… 0.0
p_top_requested        = 5000,
```

nest options

Notes on **`&domains`**

- *time_step, time_step_fract_num, time_step_frac_den*:
  - Time step for model integration in seconds.
  - Fractional time step specified in separate integers of numerator and denominator.
  - ARW: 6xDX; NMM: 2.25xDX (DX is grid distance in km)
- *s_we, s_sn, s_vert*:
  - Starting indices in X, Y, and Z direction; 1 for domain 1.
- *e_we, e_sn, e_vert*:
  - Model grid dimensions (staggered) in X, Y and Z directions.
- *num_metgrid_levels*:
  - Number of *metgrid* (input) data levels.
- *dx, dy*:
  - grid distances in meters for ARW; in degrees for NMM.

Notes on **`&domains`**

- *p_top_requested*:
  - Pressure value at the model top.
  - Constrained by the available data from WPS.
  - Default is 5000 hPa
- *eta_levels*:
  - Specify your own model levels from 1.0 to 0.0.
  - If not specified, program *real* will calculate a set of levels for you
- *ptsgm* (NMM only):
  - Pressure level (Pa) at which the WRF-NMM hybrid coordinate transitions from sigma to pressure

Where do I start?

- Always start with a *namelist* template provided in a test case directory, whether it is a ideal case, ARW or NMM.
  - 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
  - User’s Guide, Chapter 5
  - Full list of namelists and their default values can be found in Registry files: Registry.EM (ARW), Registry.NMM and registry.io_boilerplate (IO options, shared)

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 2000-01-24_12:03:00 on domain   1:    3.25000 elapsed seconds.
Timing for main: time 2000-01-24_12:06:00 on domain   1:    1.50000 elapsed seconds.
Timing for main: time 2000-01-24_12:09:00 on domain   1:    1.50000 elapsed seconds.
Timing for main: time 2000-01-24_12:12:00 on domain   1:    1.55000 elapsed seconds.
```

- Time taken to write history and restart file:

```
Timing for Writing wrfout_d01_2000-01-24_18:00:00 for domain   1:   0.14000 elapsed
     seconds.
```

- Any model error prints: (example from ARW run)

```
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

Simple Trouble Shooting

Often-seen runtime problems

- `module_configure: initial_config: error reading namelist: &dynamics`

  > Typos or erroneous namelist variables exist in namelist record &dynamics in *namelist.input* file

- `input wrf.F: SIZE MISMATCH:  namelist ide,jde,num_metgrid_levels= 70 61 27 ; input data ide,jde,num_metgrid_levels= 74 61 27`

  > Grid dimensions in error

Often-seen runtime problems

– `Segmentation fault (core dumped)`

> Often typing **`'unlimit'`** or **`'ulimit -s unlimited'`** or equivalent can help when this happens quickly in a run.

– `121 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`

> Model becomes unstable due to various reasons. If it happens soon after the start time, check input data, and/or reduce time step.

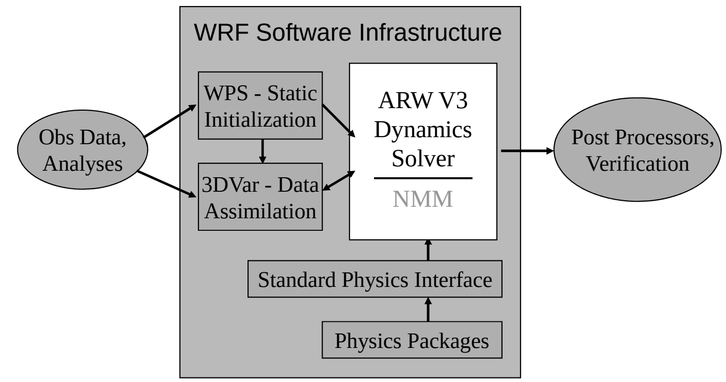
References

- Information on compiling and running WRF, and a more extensive list of namelist options and their definition / explanations can be found in the ARW and NMM User's Guide, Chapter 5
- Also see '*Nesting Setup and Run*' talk.

ARW Dynamics & Numerics

The Advanced Research WRF (ARW) Dynamics Solver

Bill Skamarock
skamaroc@ucar.edu
Jimy Dudhia
dudhia@ucar.edu



WRF ARW Tech Note
A Description of the Advanced Research WRF Version 3
http://www.mmm.ucar.edu/wrf/users/pub-doc.html

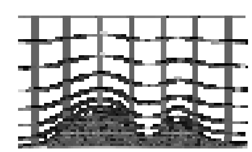
ARW Dynamical Solver

– Terrain representation
– Vertical coordinate
– Equations / variables
– Time integration scheme
– Grid staggering
– Advection scheme
– Time step parameters
– Filters
– Boundary conditions
– Nesting
– Map projections

ARW, Terrain Representation

Lower boundary condition for the geopotential $(\phi = gz)$ specifies the terrain elevation, and specifying the lowest coordinate surface to be the terrain results in a terrain-following coordinate.

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} + \omega\frac{\partial \phi}{\partial \eta} = gw$$



Vertical coordinate:

hydrostatic pressure π $\quad \eta = \frac{(\pi - \pi_t)}{\mu}, \quad \mu = \pi_s - \pi_t$

Flux-Form Equations in ARW

Hydrostatic pressure coordinate:

hydrostatic pressure π

$$\eta = \frac{(\pi - \pi_t)}{\mu}, \qquad \mu = \pi_s - \pi_t \qquad \mu(x)\Delta\eta = \Delta\pi = -g\rho\Delta z$$

Conserved state variables:

$$\mu, \quad U = \mu u, \quad V = \mu v, \quad W = \mu w, \quad \Theta = \mu\theta$$

Non-conserved state variable: $\phi = gz$

January 2009

Flux-Form Equations in ARW

Inviscid, 2-D equations without rotation:

$$\frac{\partial U}{\partial t} + \mu\alpha\frac{\partial p}{\partial x} + \frac{\partial p}{\partial \eta}\frac{\partial \phi}{\partial x} = -\frac{\partial Uu}{\partial x} - \frac{\partial \Omega u}{\partial \eta}$$

$$\frac{\partial W}{\partial t} + g\left(\mu - \frac{\partial p}{\partial \eta}\right) = -\frac{\partial Uw}{\partial x} - \frac{\partial \Omega w}{\partial \eta}$$

$$\frac{\partial \Theta}{\partial t} + \frac{\partial U\theta}{\partial x} + \frac{\partial \Omega\theta}{\partial \eta} = \mu Q$$

$$\frac{\partial \mu}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial \Omega}{\partial \eta} = 0$$

$$\frac{d\phi}{dt} = gw$$

Diagnostic relations:

$$\frac{\partial \phi}{\partial \eta} = -\mu\alpha, \qquad p = \left(\frac{R\theta}{p_0\alpha}\right)^{\gamma}, \qquad \Omega = \mu\dot{\eta}$$

January 2009

Moist Equations in ARW

Moist Equations:

$$\frac{\partial U}{\partial t} + \alpha\mu_d\frac{\partial p}{\partial x} + \frac{\alpha}{\alpha_d}\frac{\partial p}{\partial \eta}\frac{\partial \phi}{\partial x} = -\frac{\partial Uu}{\partial x} - \frac{\partial \Omega u}{\partial \eta}$$

$$\frac{\partial W}{\partial t} + g\left(\mu_d - \frac{\alpha}{\alpha_d}\frac{\partial p}{\partial \eta}\right) = -\frac{\partial Uw}{\partial x} - \frac{\partial \Omega w}{\partial \eta}$$

$$\frac{\partial \mu_d}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial \Omega}{\partial \eta} = 0$$

$$\frac{\partial(\mu_d q_{v,l})}{\partial t} + \frac{\partial(U q_{v,l})}{\partial x} + \frac{\partial(\Omega q_{v,l})}{\partial \eta} = \mu Q_{v,l}$$

Diagnostic relations:

$$\frac{\partial \phi}{\partial \eta} = -\alpha_d\mu_d, \qquad p = \left(\frac{R\Theta}{p_o\mu_d\alpha_v}\right)^{\gamma}$$

January 2009

Time Integration in ARW

3rd Order Runge-Kutta time integration

advance $\phi^t \rightarrow \phi^{t+\Delta t}$

$$\phi^* = \phi^t + \frac{\Delta t}{3}R(\phi^t)$$

$$\phi^{**} = \phi^t + \frac{\Delta t}{2}R(\phi^*)$$

$$\phi^{t+\Delta t} = \phi^t + \Delta t\, R(\phi^{**})$$

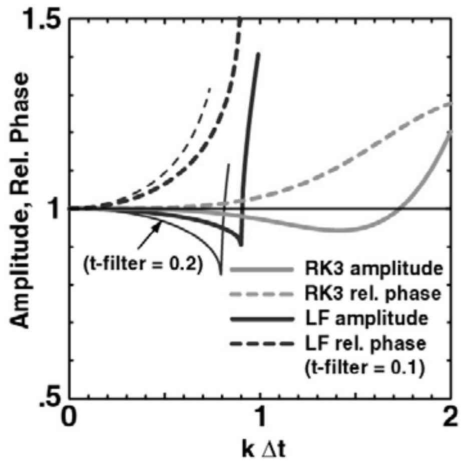
Amplification factor $\phi_t = ik\phi; \quad \phi^{n+1} = A\phi^n; \quad |A| = 1 - \frac{(k\Delta t)^4}{24}$

January 2009

Phase and amplitude errors for LF, RK3

Oscillation equation analysis

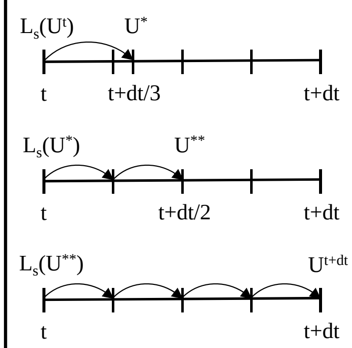
$$\phi_t = ik\phi$$



Time-Split Runge-Kutta Integration Scheme

$$U_t = L_{fast}(U) + L_{slow}(U)$$

3rd order Runge-Kutta, 3 steps



- RK3 is 3rd order accurate for linear eqns, 2nd order accurate for nonlinear eqns.
- Stable for centered and upwind advection schemes.
- Stable for Courant number Udt/dx < 1.73
- Three L_{slow}(U) evaluations per timestep.

WRF ARW Model Integration Procedure

Begin time step

Runge-Kutta loop (steps 1, 2, and 3)

(i) advection, p-grad, buoyancy using $(\phi^t, \phi^*, \phi^{**})$

(ii) physics if step 1, save for steps 2 and 3

(iii) mixing, other non-RK dynamics, save…

(iv) assemble dynamics tendencies

Acoustic step loop

(i) advance U,V, then ο.!S, then w, h

(ii) time-average U,V, Y

End acoustic loop

Advance scalars using time-averaged U,V, Y

End Runge-Kutta loop

Adjustment physics (currently microphysics)

End time step

Flux-Form Perturbation Equations

Introduce the perturbation variables:

$$\phi = \bar{\phi}(z) + \phi', \ \mu = \bar{\mu}\square + \mu';$$
$$p = \bar{p}(z) + p', \alpha = \bar{\alpha}(z) + \alpha'$$

Note – $\phi = \bar{\phi}(z) = \bar{\phi}(x, y, \eta)$,

likewise $\bar{p}(x, y, \eta), \bar{\alpha}(x, y, \eta)$

Momentum and hydrostatic equations become:

$$\frac{\partial U}{\partial t} + \mu\alpha\frac{\partial p'}{\partial x} + \eta\mu\alpha'\frac{\partial \bar{\mu}}{\partial x} + \mu\frac{\partial \phi'}{\partial x} + \frac{\partial \phi'}{\partial x}\left(\frac{\partial p'}{\partial \eta} - \mu'\right) = -\frac{\partial Uu}{\partial x} - \frac{\partial \Omega u}{\partial \eta}$$

$$\frac{\partial W}{\partial t} + g\left(\mu' - \frac{\partial p'}{\partial \eta}\right) = -\frac{\partial Uw}{\partial x} - \frac{\partial \Omega w}{\partial \eta}$$

$$\frac{\partial \phi'}{\partial \eta} = -\bar{\mu}\alpha' - \bar{\alpha}\mu'$$

Flux-Form Perturbation Equations: Acoustic Step

Acoustic mode separation:

Recast Equations in terms of perturbation about time t

$$U' = U'^{\,t} + U'', \; V' = V'^{\,t} + V'', \; W' = W'^{\,t} + W'',$$
$$\Theta' = \Theta'^{\,t} + \Theta'', \quad \mu' = \mu'^{\,t} + \mu'', \; \phi' = \phi'^{\,t} + \phi'';$$
$$p' = p'^{\,t} + p'', \; \alpha' = \alpha'^{\,t} + \alpha''$$

Linearize ideal gas law about time t

$$p'' = \frac{c_s^2}{\alpha^t}\left(\frac{\Theta''}{\Theta^t} - \frac{\alpha''}{\alpha^t} - \frac{\mu''}{\mu^t}\right)$$

$$\alpha'' = \frac{1}{\mu^t}\left(\frac{\partial \phi''}{\partial \eta} + \alpha^t \mu''\right)$$

Vertical pressure gradient becomes

$$\frac{\partial p''}{\partial \eta} = \frac{\partial}{\partial \eta}\left(\frac{c_s^2}{\mu^t {\alpha^t}^2}\frac{\partial \phi''}{\partial \eta} + \frac{c_s^2}{\mu^t}\frac{\Theta''}{\Theta^t}\right)$$

Acoustic Integration in ARW

Forward-backward scheme, first advance the horizontal momentum

$$\delta_\tau U'' + \mu^t \alpha^t \frac{\partial p''}{\partial x} + \eta \mu^t \frac{\partial \overline{\mu}}{\partial x}\alpha'' + \mu^t \frac{\partial \phi''}{\partial x} + \frac{\partial \phi^t}{\partial x}\left(\frac{\partial p''}{\partial \eta} - \mu''\right) = R_u^{\,t}$$

Second, advance continuity equation, diagnose omega, and advance thermodynamic equation

$$\delta_\tau \mu'' + \left(\nabla \cdot \mathrm{V}''\right)_\eta^{\tau+\Delta\tau} = R_\mu^{\,t}$$
$$\delta_\tau \Theta'' + \left(\nabla \cdot \mathrm{V}''\theta^t\right)_\eta^{\tau+\Delta\tau} = R_\Theta^{\,t}$$

Finally, vertically-implicit integration of the acoustic and gravity wave terms

$$\delta_\tau W'' + g\,\overline{\left[\mu'' - \frac{\partial}{\partial \eta}\left(\frac{c_s^2}{\mu^t {\alpha^t}^2}\frac{\partial \phi''}{\partial \eta} + \frac{c_s^2}{\alpha^t}\frac{\Theta''}{\Theta^t}\right)\right]}^{\tau} = R_w^{\,t}$$

$$\delta_\tau \phi'' + \frac{1}{\mu^t}\left[\left(\mathrm{V}''\cdot\nabla\phi^t\right)_\eta^{\tau+\Delta\tau} - g\overline{W''}^{\tau}\right] = R_\varphi^{\,t}$$

Hydrostatic Option

Instead of solving vertically implicit equations for W'' and ϕ''

Integrate the hydrostatic equation to obtain $p''^{\tau+\Delta\tau}$:

$$\frac{\partial p''}{\partial \eta} = \left(\frac{\alpha_d}{\alpha}\right)^t \mu_d''$$

Solve the linearized ideal gas law for $\alpha_d''^{\tau+\Delta\tau}$:

$$p'' = \frac{c_s^2}{\alpha^t}\left(\frac{\Theta''}{\Theta^t} - \frac{\alpha_d''}{\alpha_d^t} - \frac{\mu_d''}{\mu_d^t}\right)$$

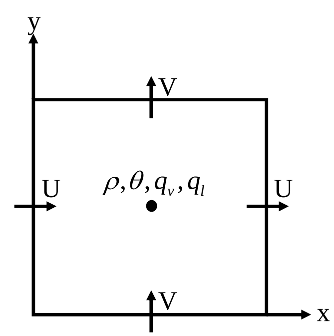
and recover $\phi''^{\tau+\Delta\tau}$ from:

$$\alpha_d'' = \frac{1}{\mu_d^t}\left(\frac{\partial \phi''}{\partial \eta} + \alpha_d^t \mu_d''\right)$$

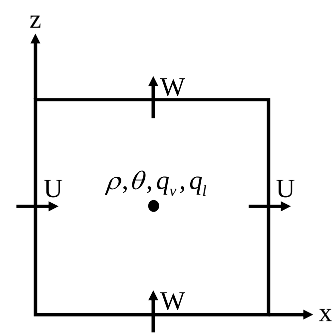
W'' is no longer required during the integration.

ARW model, grid staggering

C-grid staggering



horizontal



vertical

Advection in the ARW Model

2nd, 3rd, 4th, 5th and 6th order centered and upwind-biased schemes are available in the ARW model.

Example: 5th order scheme

$$\frac{\partial(U\phi)}{\partial x}=\frac{1}{\Delta x}\left(F_{i+\frac{1}{2}}(U\phi)-F_{i-\frac{1}{2}}(U\phi)\right)$$

where

$$F_{i-\frac{1}{2}}(U\phi)= U_{i-\frac{1}{2}}\left\{\frac{37}{60}(\phi_i+\phi_{i-1})-\frac{2}{15}(\phi_{i+1}+\phi_{i-2})+\frac{1}{60}(\phi_{i+2}+\phi_{i-3})\right\}$$
$$-sign(1,U)\,\frac{1}{60}\,\left\{(\phi_{i+2}-\phi_{i-3})-5(\phi_{i+1}-\phi_{i-2})+10(\phi_i-\phi_{i-1})\right\}$$

Advection in the ARW Model

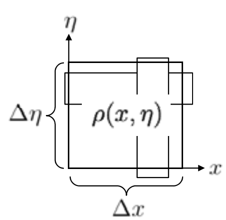
For constant U, the 5th order flux divergence tendency becomes

$$\Delta t\frac{\delta(U\phi)}{\Delta x}\bigg|_{5th}=\Delta t\frac{\delta(U\phi)}{\Delta x}\bigg|_{6th}-\underbrace{\left|\frac{U\Delta t}{\Delta x}\right|\frac{1}{60}\left(-\phi_{i-3}+6\phi_{i-2}-15\phi_{i-1}+20\phi_i-15\phi_{i+1}+6\phi_{i+2}-\phi_{i+3}\right)}_{\frac{Cr}{60}\frac{\partial^6\phi}{\partial x^6}+H.O.T}$$

The odd-ordered flux divergence schemes are equivalent to the next higher ordered (even) flux-divergence scheme plus a dissipation term of the higher even order with a coefficient proportional to the Courant number.

Mass Conservation in the ARW Model

control volume (2D example) →



Mass in a control volume is proportional to

$$(\Delta x\Delta\eta)(\mu)^t$$

since $\mu(x)\Delta\eta=\Delta\pi=-g\rho\Delta z$

Mass Conservation in the ARW Model

Mass in a control volume $(\Delta x\Delta\eta)(\mu)^t$
2D example

Mass conservation equation

$$\boxed{\Delta t^{-1}(\Delta x\Delta\eta)\cdot\left[(\mu)^{t+\Delta t}-(\mu)^t\right]}=\boxed{\begin{array}{l}\left[(\mu u\Delta\eta)_{x-\Delta x/2,\eta}-(\mu u\Delta\eta)_{x+\Delta x/2,\eta}\right]+\\ \left[(\mu\omega\Delta x)_{x,\eta-\Delta\eta/2}-(\mu\omega\Delta x)_{x,\eta+\Delta\eta/2}\right]\end{array}}$$

Change in mass over a time step ↑

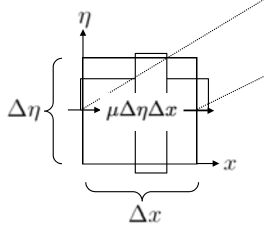
mass fluxes through control volume faces ↑

Mass Conservation in the ARW Model

Mass in a control volume $(\Delta x \Delta \eta)(\mu)^t$

Mass conservation equation

$$\Delta t^{-1}(\Delta x \Delta \eta)\cdot\left[(\mu)^{t+\Delta t} - (\mu)^t\right] = \left[(\mu u \Delta \eta)_{x-\Delta x/2,\eta} - (\mu u \Delta \eta)_{x+\Delta x/2,\eta}\right] + \left[(\mu \omega \Delta x)_{x,\eta-\Delta \eta/2} - (\mu \omega \Delta x)_{x,\eta+\Delta \eta/2}\right]$$



Horizontal fluxes through the vertical control-volume faces

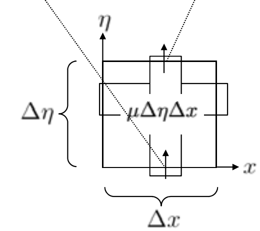
Mass Conservation in the ARW Model

Mass in a control volume $(\Delta x \Delta \eta)(\mu)^t$

Mass conservation equation

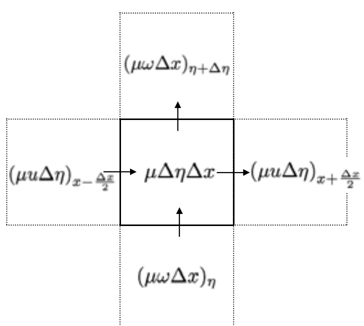
$$\Delta t^{-1}(\Delta x \Delta \eta)\cdot\left[(\mu)^{t+\Delta t} - (\mu)^t\right] = \left[(\mu u \Delta \eta)_{x-\Delta x/2,\eta} - (\mu u \Delta \eta)_{x+\Delta x/2,\eta}\right] + \left[(\mu \omega \Delta x)_{x,\eta-\Delta \eta/2} - (\mu \omega \Delta x)_{x,\eta+\Delta \eta/2}\right]$$

Vertical fluxes through the horizontal control-volume faces



Mass Conservation in the ARW Model

The same mass fluxes are used for neighboring grid cells - hence mass is conserved locally and globally.



Scalar Mass Conservation in the ARW Model

Mass in a control volume $(\Delta x \Delta \eta)(\mu)^t$

Scalar mass $(\Delta x \Delta \eta)(\mu\phi)^t$

Mass conservation equation:

$$\Delta t^{-1}(\Delta x \Delta \eta)\cdot\left[(\mu)^{t+\Delta t} - (\mu)^t\right] = \left[(\mu u \Delta \eta)_{x-\Delta x/2,\eta} - (\mu u \Delta \eta)_{x+\Delta x/2,\eta}\right] + \left[(\mu \omega \Delta x)_{x,\eta-\Delta \eta/2} - (\mu \omega \Delta x)_{x,\eta+\Delta \eta/2}\right]$$

change in mass over a time step — mass fluxes through control volume faces

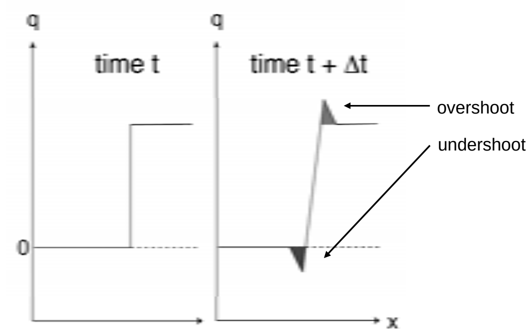
Scalar mass conservation equation:

$$\Delta t^{-1}(\Delta x \Delta \eta)\cdot\left[(\mu\phi)^{t+\Delta t} - (\mu\phi)^t\right] = \left[(\mu u \phi \Delta \eta)_{x-\Delta x/2,\eta} - (\mu u \phi \Delta \eta)_{x+\Delta x/2,\eta}\right] + \left[(\mu \omega \phi \Delta x)_{x,\eta-\Delta \eta/2} - (\mu \omega \phi \Delta x)_{x,\eta+\Delta \eta/2}\right]$$

change in tracer mass over a time step — tracer mass fluxes through control volume faces

Moisture Transport in ARW

1D advection



ARW scheme is conservative,
but not positive definite nor monotonic.
Removal of negative q■
results in spurious source of q■ .

January 2009

Positive-Definite Flux Renormalization

Scalar update, last RK3 step

$$(\mu\phi)^{t+\Delta t} = (\mu\phi)^{t} - \Delta t \sum_{i=1}^{n} \delta_{x_i}[f_i] \qquad (1)$$

(1) Decompose flux: $f_i = f_i^{upwind} + f_i^c$

(2) Renormalize high-order correction fluxes f_i^c such that solution is positive definite: $f_i^c = R(f_i^c)$

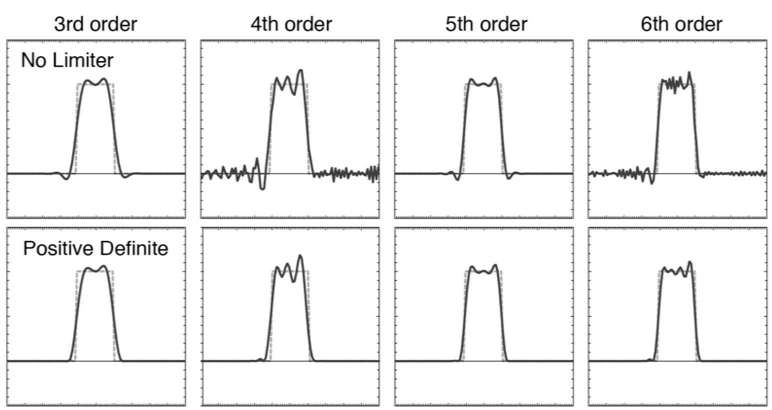
(3) Update scalar eqn. (1) using $f_i = f_i^{upwind} + R(f_i^c)$

Skamarock, MWR 2006, 2241-2250

January 2009

PD Limiter in ARW - 1D Example Top-Hat Advection

Cr = 0.5, 1 revolution (200 steps)



January 2009

ARW Model: Dynamics Parameters

3rd order Runge-Kutta time step

Courant number limited, 1D: $C_r = \frac{U\Delta t}{\Delta x} < 1.73$

Generally stable using a timestep approximately twice as large as used in a leapfrog model.

Acoustic time step

2D horizontal Courant number limited: $C_r = \frac{C_s \Delta \tau}{\Delta h} < \frac{1}{\sqrt{2}}$

$\Delta \tau_{sound} = \Delta t_{RK} / (\text{number of acoustic steps})$

Guidelines for time step

Δt in seconds should be about 6**Δx* (grid size in kilometers). Larger *Δt* can be used in smaller-scale dry situations, but *time_step_sound* (default = 4) should increase proportionately if larger *Δt* is used.

January 2009

Maximum Courant Number for Advection

$$C_a = U \Delta t / \Delta x$$

Time Integration Scheme	*Advection Scheme*				
	2nd	*3rd*	*4th*	*5th*	*6th*
Leapfrog (α=0.1)	0.91	U	0.66	U	0.57
RK2	U	0.90	U	0.39	U
RK3	1.73	1.63	1.26	1.43	1.09

U = unstable

(Wicker & Skamarock, 2002)

ARW Filters: Divergence Damping

Purpose: filter acoustic modes

$$p^{*\tau} = p^{\tau} + \gamma_d \left(p^{\tau} - p^{\tau - \Delta\tau}\right) \quad \text{since} \quad p_t \sim c^2 \nabla \cdot \rho \mathbf{V}$$

$$\delta_\tau U'' + \mu^{t^*} \alpha^{t^*} \partial_x p''^{\tau} + (\mu^{t^*} \partial_x \bar{p}) \alpha''^{\tau} + (\alpha/\alpha_d)[\mu^{t^*} \partial_x \phi''^{\tau} + (\partial_x \phi^{t^*})(\partial_\eta p'' - \mu'')^{\tau}] = R_U{}^{t^*}$$

$$\delta_\tau V'' + \mu^{t^*} \alpha^{t^*} \partial_y p''^{\tau} + (\mu^{t^*} \partial_y \bar{p}) \alpha''^{\tau} + + (\alpha/\alpha_d)[\mu^{t^*} \partial_y \phi''^{\tau} + (\partial_y \phi^{t^*})(\partial_\eta p'' - \mu'')^{\tau}] = R_V{}^{t^*}$$

γ_d = 0.1 recommended (default)

ARW Filters: External Mode Filter

Purpose: filter the external mode
(primarily for real-data applications)

Additional terms:

$$\delta_\tau U'' = \ldots \quad \underline{- \gamma_e \left(\Delta x^2 / \Delta\tau\right) \delta_x (\delta_{\tau - \Delta\tau} \mu_d'')}$$

$$\delta_\tau V'' = \ldots \quad \underline{- \gamma_e \left(\Delta y^2 / \Delta\tau\right) \delta_y (\delta_{\tau - \Delta\tau} \mu_d'')}$$

$$\delta_\tau \mu_d = m^2 \int_1^0 [\partial_x U'' + \partial_y V'']^{\tau + \Delta\tau} d\eta$$

γ_e = 0.01 recommended (default)

ARW Filters: Vertically Implicit Off-Centered Acoustic Step

Purpose: damp vertically-propagating acoustic modes

$$\delta_\tau W'' - m^{-1} g \overline{\left[(\alpha/\alpha_d)^{t^*} \partial_\eta (C \partial_\eta \phi'') + \partial_\eta \left(\frac{c_s^2}{\alpha^{t^*}} \frac{\Theta''}{\Theta^{t^*}}\right) - \mu_d''\right]}^{\tau} = R_W{}^{t^*}$$

$$\delta_\tau \phi'' + \frac{1}{\mu_d^{t^*}} [m \Omega^{\tau + \Delta\tau} \phi_\eta - \overline{g W''}^{\tau}] = R_\phi{}^{t^*}.$$

$$\overline{a}^{\tau} = \frac{1 + \beta}{2} a^{\tau + \Delta\tau} + \frac{1 - \beta}{2} a^{\tau}$$

β = 0.1 recommended (default)

ARW Filters: Vertical Velocity Damping

Purpose: damp anomalously-large vertical velocities

(usually associated with anomalous physics tendencies)

Additional term:

$$\partial_t W = ... - \underline{\mu_d \, \mathrm{sign}(W) \gamma_w (Cr - Cr_\beta)}$$

$$Cr = \left| \frac{\Omega dt}{\mu d\eta} \right|$$

$Cr_\beta = 1.0$ typical value (default)

$\gamma_w = 0.3$ m/s^2 recommended (default)

January 2009

ARW Filters: 2nd-Order Horizontal Mixing, Horizontal-Deformation-Based K_h

Purpose: mixing on horizontal coordinate surfaces (real-data applications)

$$K_h = C_s^2 \, l^2 \left[0.25(D_{11} - D_{22})^2 + \overline{D_{12}^2}^{xy} \right]^{\frac{1}{2}}$$

where $l = (\Delta x \Delta y)^{1/2}$

$$D_{11} = 2\,m^2 \left[\partial_x(m^{-1}u) - z_x \partial_z(m^{-1}u)\right]$$
$$D_{22} = 2\,m^2 \left[\partial_y(m^{-1}v) - z_y \partial_z(m^{-1}v)\right]$$
$$D_{12} = m^2 \left[\partial_y(m^{-1}u) - z_y \partial_z(m^{-1}u) + \partial_x(m^{-1}v) - z_x \partial_z(m^{-1}v)\right]$$

$C_s = 0.25$ (Smagorinsky coefficient, default value)

January 2009

ARW Filters: Upper Level Gravity-Wave Absorbers

(1) Absorbing layer using spatial filtering

Horizontal and vertical 2nd order diffusion operators with eddy viscosities that increase with height.

$$K_{dh} = \frac{\Delta x^2}{\Delta t} \gamma_g \cos\left(\frac{\pi}{2} \frac{z_{top} - z}{z_d} \right)$$

$$K_{dv} = \frac{\Delta z^2}{\Delta t} \gamma_g \cos\left(\frac{\pi}{2} \frac{z_{top} - z}{z_d} \right)$$

z_d - depth of the damping layer

K_{dh}, K_{dv} - horizontal and vertical eddy viscosities

γ_g - dimensionless damping coefficient, typical value 0.003

Not recommended !

January 2009

ARW Filters: Upper Level Gravity-Wave Absorbers

(2) Traditional Rayleigh Damping - idealized cases only!

$$\frac{\partial u}{\partial t} = -\tau(z)\,(u - \overline{u})$$
$$\frac{\partial v}{\partial t} = -\tau(z)\,(v - \overline{v})$$
$$\frac{\partial w}{\partial t} = -\tau(z) w,$$
$$\frac{\partial \theta}{\partial t} = -\tau(z)\,(\theta - \overline{\theta})$$

$$\tau(z) = \begin{cases} \gamma_r \sin^2\left[\frac{\pi}{2}\left(1 - \frac{z_{top} - z}{z_d}\right)\right] & \text{for } z \geq (z_{top} - z_d); \\ 0 & \text{otherwise,} \end{cases}$$

$\tau(z)$ - damping rate (t^{-1})

z_d - depth of the damping layer

γ_r - dimensionless damping coefficient

January 2009

ARW Filters: Upper Level Gravity-Wave Absorbers

(3) Implicit Rayleigh W - damping (nonhydrostatic equations only!)

$$\tilde{W}''^{\tau+\Delta\tau} - W''^{\tau} - g\Delta\tau \frac{\alpha}{\alpha_d}\frac{\partial}{\partial_\eta}\left(C\frac{\partial\overline{\phi''}^{\tau}}{\partial\eta}\right) = \Delta\tau R_W^* \quad (1)$$

$$W''^{\tau+\Delta\tau} - \tilde{W}''^{\tau+\Delta\tau} = -\tau(z)\Delta\tau W''^{\tau+\Delta\tau} \quad (2)$$

$$\phi''^{\tau+\Delta\tau} - \phi''^{\tau} - g\Delta\tau\frac{1}{\mu}\overline{W''}^{\tau} = \Delta\tau R_\phi^* \quad (3)$$

Vertically implicit solution procedure:
Eliminate $\phi''^{\tau+\Delta\tau}$ from (1) using (3), solve for $\tilde{W}''^{\tau+\Delta\tau}$.
Apply implicit Rayleigh damping - solve for $W''^{\tau+\Delta\tau}$ using (2).
Recover the geopotential using $W''^{\tau+\Delta\tau}$ in (3).

$$\tau(z) = \begin{cases} \gamma_r \sin^2\left[\frac{\pi}{2}\left(1 - \frac{z_{top}-z}{z_d}\right)\right] & \text{for } z \geq (z_{top} - z_d); \\ 0 & \text{otherwise,} \end{cases}$$

$\tau(z)$ - damping rate (t^{-1})
z_d - depth of the damping layer
γ_r - dimensionless damping coefficient

Global WRF - Latitude-Longitude Grid

WRF Version 3 Release

Additions to WRF Version 2

- Map factors are generalized - m_x and m_y
  - Computational grid poles need not be geographic poles.
  - Limited area and nesting capable.
- Polar boundary conditions
- Polar filtering

ARW Map Projections

ARW map factors

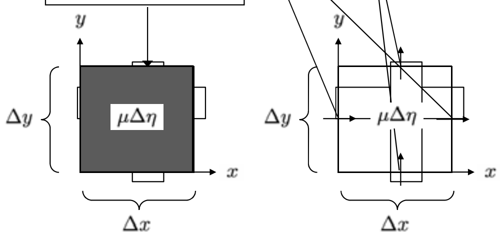
Map-scale factor: $m_x = \frac{\Delta x}{\text{distance on the earth}}, \quad m_y = \frac{\Delta y}{\text{distance on the earth}}$

Continuity equation:

$$\frac{\partial\mu}{\partial t} + m_x m_y\left[\frac{\partial}{\partial x}\left(\frac{\mu u}{m_y}\right) + \frac{\partial}{\partial y}\left(\frac{\mu v}{m_x}\right)\right] + \frac{\partial}{\partial\eta}(\mu\omega) = 0$$

$\propto \frac{1}{\text{control volume area}}$

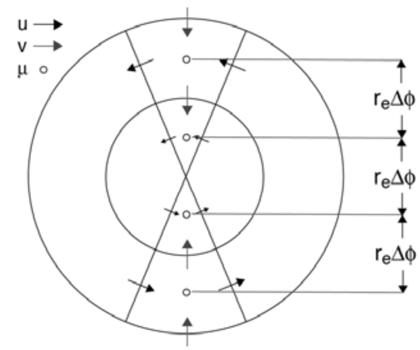
Control volume:



Lat-Long Grid Global WRF

Lat-Long WRFV3

Polar boundary condition (pole point).



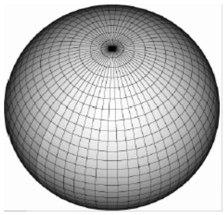
Meridional velocity (v) is undefined at the poles.

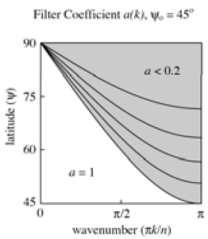
Zero meriodional flux at the poles (cell-face area is zero).

v (poles) only needed for meridional derivative of v near the poles (some approximation needed).

All other meriodional derivatives are well-defined near/at poles.

ARW Filters: Polar Filter





Converging gridlines severely limit timestep.
The polar filter removes this limitation.

Filter procedure - Along a grid latitude circle:
1. Fourier transform variable.
2. Filter Fourier coefficients.
3. Transform back to physical space.

$$\hat{\phi}(k)_{filtered} = a(k)\,\hat{\phi}(k), \quad \text{for all } k$$

$$a(k) = \min\left[1., \max\left(0., \left(\frac{\cos\psi}{\cos\psi_o}\right)^2 \frac{1}{\sin^2(\pi k/n)}\right)\right]$$

$k =$ dimensionless wavenumber
$\hat{\phi}(k) =$ Fourier coefficients from forward transform
$a(k) =$ filter coefficients
$\psi =$ latitude $\psi_o =$ polar filter latitude, filter when $|\psi| > \psi_o$

January 2009

WRF ARW Model Integration Procedure

Begin time step
- Runge-Kutta loop (steps 1, 2, and 3)
  - (i) advection, p-grad, buoyancy, physics
  - Acoustic step loop
    - (i) advance U,V
    - (ii) advance ○ .'S """
    - (iii) advance w, h"'
  - End acoustic loop
  - Advance scalars
- End Runge-Kutta loop
- Adjustment physics (currently microphysics)

End time step

January 2009

WRF ARW Model Integration Procedure

Begin time step
- Runge-Kutta loop (steps 1, 2, and 3)
  - (i) advection, p-grad, buoyancy, physics
  - Acoustic step loop
    - (i) advance U,V (Fourier Filter U,V)
    - (ii) advance ○ .'S """(Fourier Filter ○ .'S)
    - (iii) advance w, h"(Fourier Filter w, h)
  - End acoustic loop
  - Advance scalars (Fourier Filter Sc)"'
- End Runge-Kutta loop
- Adjustment physics (currently microphysics) (Fourier Filter Sc)

End time step

Timestep limited by minimum Δx outside of polar-filter region.

January 2009

ARW Model: Coordinate Options

1. Cartesian geometry:
   idealized cases
2. Lambert Conformal:
   mid-latitude applications
3. Polar Stereographic:
   high-latitude applications
4. Mercator:
   low-latitude applications
5. Latitude-Longitude (new in ARW V3)
   global
   regional

Projections 1-4 are isotropic ($m_x = m_y$)
Latitude-Longitude projection is anistropic ($m_x \neq m_y$)

January 2009

ARW Model: Boundary Condition Options

Lateral boundary conditions

1. Specified (Coarse grid, real-data applications).
2. Open lateral boundaries (gravity-wave radiative).
3. Symmetric lateral boundary condition (free-slip wall).
4. Periodic lateral boundary conditions.
5. Nested boundary conditions (specified).

Top boundary conditions

1. Constant pressure.

Bottom boundary conditions

1. Free slip.
2. Various B.L. implementations of surface drag, fluxes.

ARW Model: Nesting

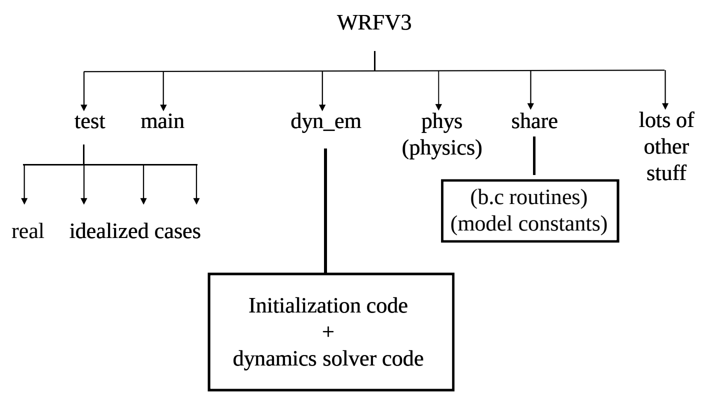
2-way nesting

1. Multiple domains run concurrently
2. Multiple levels, multiple nests per level
3. Any integer ratio grid size and time step
4. Parent domain provides nest boundaries
5. Nest feeds back interior values to parent

1-way nesting

1. Parent domain is run first
2. *ndown* uses coarse output to generate nest boundary conditions
3. Nest initial conditions from fine-grid input file
4. Nest is run after *ndown*

WRF ARW code



WRF ARW Tech Note
A Description of the Advanced Research WRF Version 3 (June 2008)
http://www.mmm.ucar.edu/wrf/users/pub-doc.html

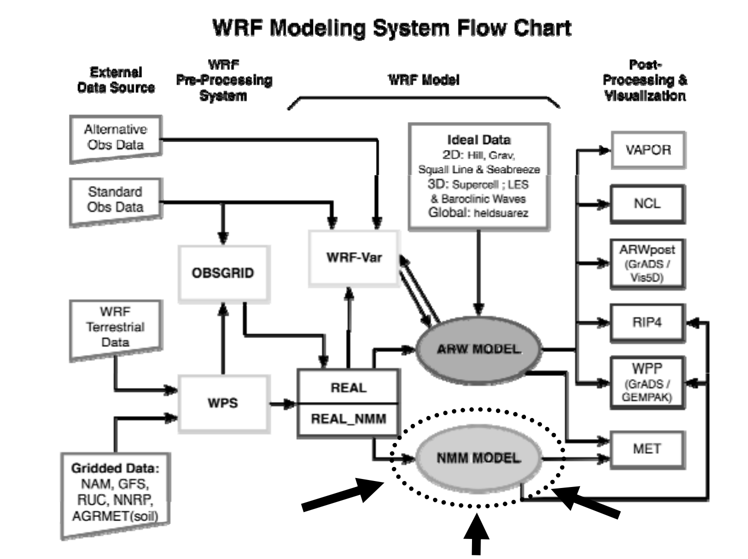
NMM Dynamics & Numerics

National Weather Service
National Centers for Environmental Prediction

The WRF NMM Core

Zavisa Janjic

Talk modified and presented by
Matthew Pyle



NMM Dynamic Solver

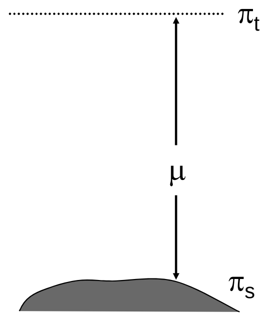
- Basic Principles
- Equations / Variables
- Model Integration
- Horizontal Grid
- Spatial Discretization
- Vertical Grid
- Boundary Conditions
- Dissipative Processes
- Summary

Basic Principles

- Use full compressible equations split into hydrostatic and nonhydrostatic contributions
  - Easy comparison of hydro and nonhydro solutions
  - Reduced computational effort at lower resolutions
- Use modeling principles proven in NWP and regional climate applications
- Use methods that minimize the generation of small-scale noise
- Robust, computationally efficient

Mass Based Vertical Coordinate

For simplicity, consider the sigma coordinate as representative of a vertical coordinate based on hydrostatic pressure (π):



$$\mu = \pi_s - \pi_t$$

$$\sigma = \frac{\pi - \pi_t}{\mu}$$

π_t = model top π
π_s = surface π

WRF-NMM dynamical equations in inviscid, adiabatic, sigma form (Janjic et al., 2001, MWR)

Analogous to a hydrostatic system, except for p and ε, where p is the total (nonhydrostatic) pressure and ε is defined below.

Momentum eqn. $$\frac{\partial \mathbf{v}}{\partial t} = -\mathbf{v}\cdot\nabla_\sigma \mathbf{v} - \dot{\sigma}\frac{\partial \mathbf{v}}{\partial \sigma} - (1+\varepsilon)\nabla_\sigma \Phi - \alpha \nabla_\sigma p + f\mathbf{k}\times\mathbf{v}$$

Thermodynamic eqn. $$\frac{\partial T}{\partial t} = -\mathbf{v}\cdot\nabla_\sigma T - \dot{\sigma}\frac{\partial T}{\partial \sigma} + \frac{\alpha}{c_p}\left[\frac{\partial p}{\partial t} + \mathbf{v}\cdot\nabla_\sigma p + \dot{\sigma}\frac{\partial p}{\partial \sigma}\right]$$

Continuity eqn. $$\frac{\partial \mu}{\partial t} + \nabla_\sigma \cdot (\mu \mathbf{v}) + \frac{\partial(\mu\dot{\sigma})}{\partial \sigma} = 0$$

$$\varepsilon \equiv \frac{1}{g}\frac{dw}{dt}$$

$$\alpha = RT/p$$

Hypsometric eqn. $$\frac{\partial \Phi}{\partial \sigma} = -\mu\frac{RT}{p}$$

Nonhydro var. definition (restated) $$\varepsilon \equiv \frac{1}{g}\frac{dw}{dt}$$

3rd eqn of motion $$\frac{\partial p}{\partial \pi} = 1 + \varepsilon$$

Nonhydrostatic continuity eqn. $$w = \frac{1}{g}\frac{d\Phi}{dt} = \frac{1}{g}\left(\frac{\partial \Phi}{\partial t} + \mathbf{v}\cdot\nabla_\sigma \Phi + \dot{\sigma}\frac{\partial \Phi}{\partial \sigma}\right)$$

Vertical boundary conditions for model equations

Top: $\dot{\sigma} = 0$, $p - \pi = 0$

Surface: $\dot{\sigma} = 0$, $\frac{\partial(p-\pi)}{\partial \sigma} = 0$

Properties of system

- Φ, w, ε are not independent, no independent prognostic equation for w!
- ε<<1 in meso and large scale atmospheric flows
- Impact of nonhydrostatic dynamics becomes detectable at resolutions <10km, important at 1km.

WRF-NMM predictive variables

- Mass variables:
  - PD – hydrostatic pressure depth (time and space varying component) (Pa)
  - PINT – nonhydrostatic pressure (Pa)
  - T – sensible temperature (K)
  - Q – specific humidity (kg/kg)
  - CWM – total cloud water condensate (kg/kg)
  - Q2 – 2 * turbulent kinetic energy (m^2/s^2)
- Wind variables:
  - U, V – wind components (m/s)

Model Integration

General Philosophy

- Explicit time differencing preferred where possible, as allows for better phase speeds and more transparent coding:
  - horizontal advection of u, v, T
  - passive substance advection of q, cloud water, TKE
- Implicit time differencing for very fast processes that would require a restrictively short time step for numerical stability:
  - vertical advection of u, v, T and vertically propagating sound waves

Model Integration

Horizontal advection of u, v, T

2^{nd} order Adams-Bashforth:

$$\frac{y^{\tau+1}-y^{\tau}}{\Delta t}=\frac{3}{2}f(y^{\tau})-\frac{1}{2}f(y^{\tau-1})$$

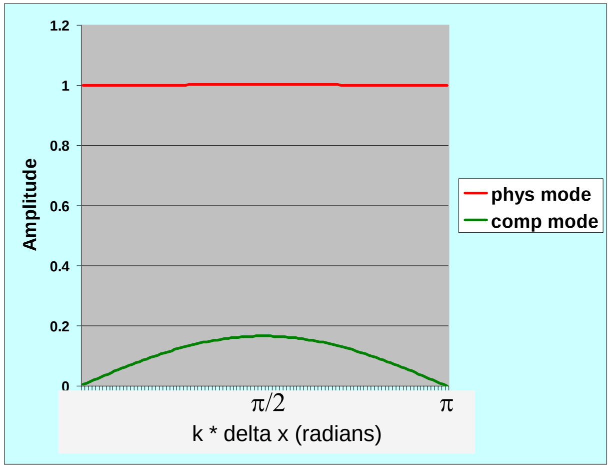
Stability/Amplification:

A-B has a weak linear instability (amplification) which can be tolerated in practice or stabilized by a slight off-centering as is done in the WRF-NMM.

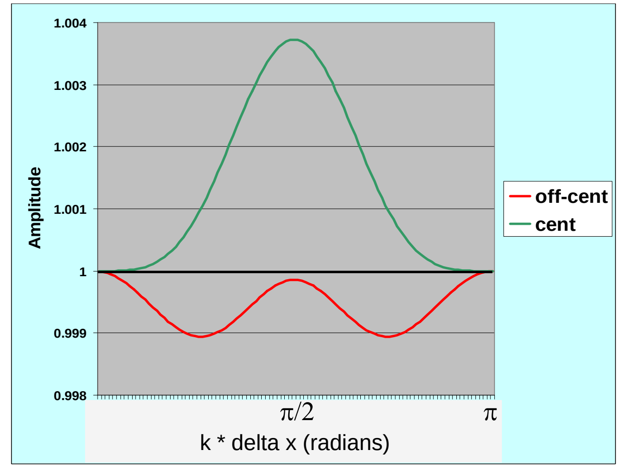
$$\frac{y^{\tau+1}-y^{\tau}}{\Delta t}=1.533f(y^{\tau})-0.533f(y^{\tau-1})$$

Adams-Bashforth amplification factor derived from

$$\frac{\Delta u}{\Delta t} + c\frac{\Delta u}{\Delta x} = 0$$ with c∆t/∆x = 0.33



Adams-Bashforth amplification factor,
Impact of off-centering



Model Integration

Vertical advection of u, v, & T

Crank-Nicolson:

$$\frac{y^{\tau+1} - y^{\tau}}{\Delta t} = \frac{1}{2}[f(y^{\tau+1}) + f(y^{\tau})]$$

Stability:

An implicit method, it is absolutely stable numerically.

Model Integration

Advection of TKE (Q2) and moisture (Q, CWM)

- Similar to Janjic (1997) scheme used in Eta model:
  - Starts with an initial upstream advection step
  - anti-diffusion/anti-filtering step to reduce dispersiveness
  - conservation enforced after each anti-filtering step – maintain global sum of advected quantity, and prevent generation of new extrema.

Model Integration

Fast adjustment processes

Forward-Backward (Ames, 1968; Janjic and Wiin-Nielsen, 1977; Janjic 1979): Mass field computed from a forward time difference, while the velocity field comes from a backward time difference.

In a shallow water equation sense:

$$\frac{\partial u}{\partial t} = -g\frac{\partial h}{\partial x}, \frac{\partial h}{\partial t} = -H\frac{\partial u}{\partial x}$$

$$h^{\tau+1} = h^{\tau} - \Delta t\, H \frac{\partial u^{\tau}}{\partial x}$$

$$u^{\tau+1} = u^{\tau} - \Delta t\, g \frac{\partial h^{\tau+1}}{\partial x}$$

} Mass field forcing to update wind from τ+1 time

Model Integration

Vertically propagating sound waves

In case of linearized equations, equivalent to implicit solution of:

$$\frac{\partial^2 p'}{\partial t^2} \rightarrow \frac{p'^{\tau+1} - 2p'^{\tau} + p'^{\tau-1}}{\Delta t^2} = \frac{c_p}{c_v} R T_0 \frac{\partial^2 p'^{\tau+1}}{\partial z_0{}^2}$$

Where p' is the perturbation pressure from a hydrostatic basic state, and τ represents the time level. (Janjic et al., 2001; Janjic, 2003). *Embedded into full equations, not actually used in the model in this form.*

Model Integration

- Sequence of events within a solve_nmm loop (ignoring physics):

(0.6%) ▪ PDTE – integrates mass flux divergence, computes vertical velocity and updates hydrostatic pressure.

(26.4%) ▪ ADVE – horizontal and vertical advection of T, u, v, Coriolis and curvature terms applied.

(1.2%) ▪ VTOA – updates nonhydrostatic pressure, applies $\omega\alpha$ term to thermodynamic equation

(8.6%) ▪ VADZ/HADZ – vertical/horizontal advection of height. w=dz/dt updated.

(10.6%) ▪ EPS – vertical and horizontal advection of dz/dt, vertical sound wave treatment.

(relative % of dynamics time spent in these subroutines)

Model Integration

- Sequence of events within a solve_nmm loop (cont):

(19.5%) ▪ VAD2/HAD2 (every other step) – vertical/horizontal advection of q, CWM, TKE

(11.8%) ▪ HDIFF – horizontal diffusion

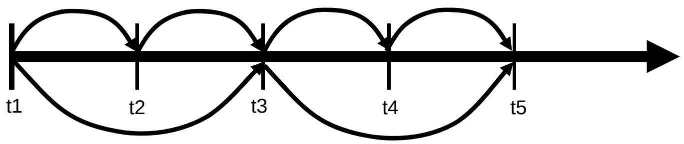
(1.2%) ▪ BOCOH – boundary update at mass points

(17.5%) ▪ PFDHT – calculates PGF, updates winds due to PGF, computes divergence.

(2.3%) ▪ DDAMP – divergence damping

(0.3%) ▪ BOCOV – boundary update at wind points

All dynamical processes every fundamental time step, except….



…passive substance advection, every other time step

Model time step "dt" specified in model namelist.input is for the fundamental time step.

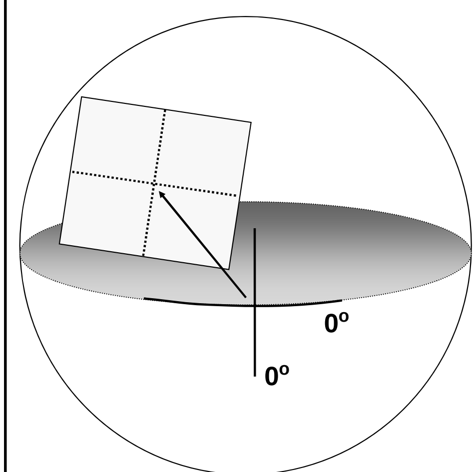
Generally about 2.25X the horizontal grid spacing (km), or 350X the namelist.input "dy" value (degrees lat).

Now we'll take a look at two items specific to the WRF-NMM horizontal grid:

- Rotated latitude-longitude map projection (only projection used with the WRF-NMM)
- The Arakawa E-grid stagger

Rotated Latitude-Longitude

- Rotates the earth's latitude/longitude grid such that the intersection of the equator and prime meridian is at the center of the model domain.
- The rotation minimizes the convergence of meridians over the domain, and maintains a more uniform earth-relative grid spacing than exists for a regular lat-lon grid.



For a domain spanning 10N to 70N:

$$\Delta x \propto \cos(lat)$$

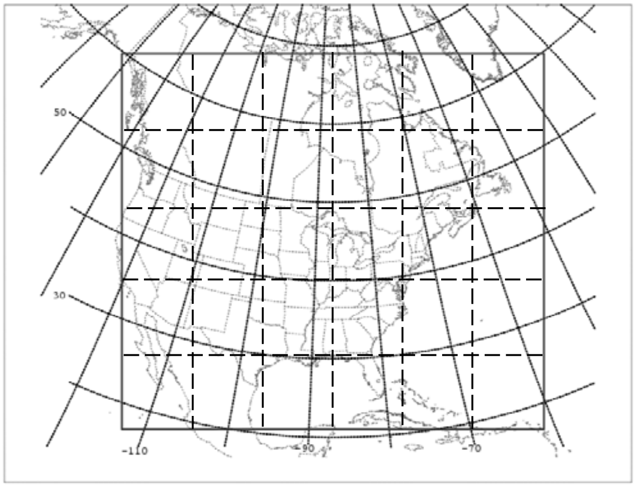
Regular lat-lon grid

$$\cos(70^0)/\cos(10^0) = 0.347$$

Rotated lat-lon grid

$$\cos(30^0)/\cos(0^0) = 0.866$$

Sample rotated lat-lon domain (red), with earth lat-lon lines (black)



The E-grid Stagger

H	v	H	v	H	v	H	(v)
v	H	v	H	v	H	v	(H)
H	v	H	v	H	v	H	(v)
v	H	v	H	v	H	v	(H)
H	v	H	v	H	v	H	(v)

H=mass point, v=wind point
red=(1,1) blue=(1,2)

The E-grid Stagger

H	v	H	v	H	v	H	(v)
v	H	v	H	v	H	v	(H)
H	v	H	v	H	v	H	(v)
v	H	v	H	v	H	v	(H)
H	v	H	v	H	v	H	(v)

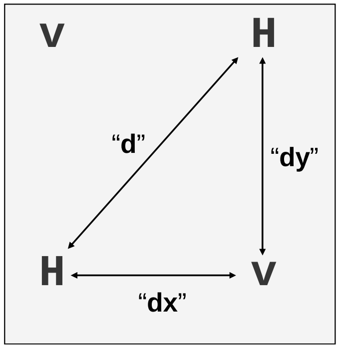
XDIM=4 (# of mass points on odd numbered row)
YDIM=5 (number of rows)

The E-grid Stagger - properties

- Due to the indexing convention, the X-dimension is half as large as would be expected from a C-grid domain (typically XDIM < YDIM for the E-grid).
- “Think diagonally” –the shortest distance between adjacent like points is along the diagonals of the grid.

- E-grid energy and enstrophy conserving momentum advection scheme (Janjic, 1984, MWR) controls the spurious nonlinear energy cascade (accumulation of small scale computational noise due to nonlinearity) more effectively than schemes on the C grid – an argument in favor of the E grid.

The E-grid Stagger



- Conventional grid spacing is the diagonal distance "**d**".

- Grid spacings in the WPS and WRF namelists are the "**dx**" and "**dy**" values, *specified in fractions of a degree for the WRF-NMM*.

- "WRF domain wizard" takes input grid spacing "**d**" in km and computes the angular distances "**dx**" and "**dy**" for the namelist.

Spatial Discretization

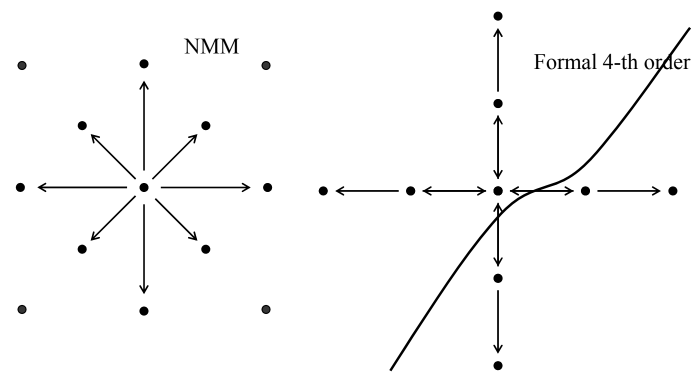
- Basic discretization principle is conservation of important properties of the continuous system.
  - "Mimetic" approach

    http://www.math.unm.edu/~stanly/mimetic/mimetic.html
  - Major novelty in applied mathematics, …
  - … but well established in atmospheric modeling (Arakawa, 1966, 1972, 1977 …; Sadourny, 1975 …; Janjic, 1977, 1984 …; Tripoli, 1992, …)

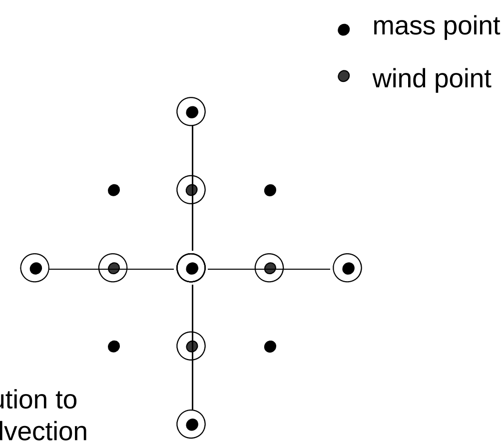
Spatial Discretization

General Philosophy

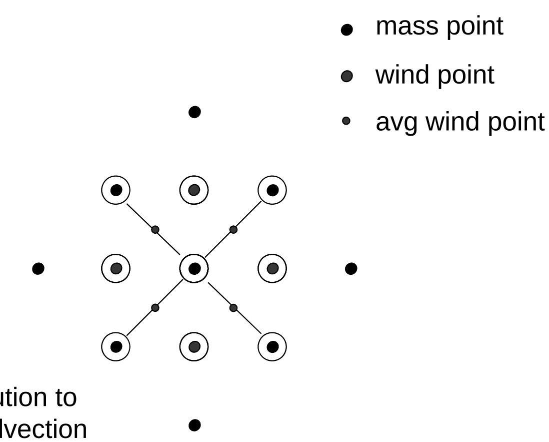
- Conserve energy and enstrophy in order to control nonlinear energy cascade; eliminate the need for numerical filtering to the extent possible.
- Conserve a number of first order and quadratic quantities (mass, momentum, energy, ...).
- Use consistent order of accuracy for advection and divergence operators and the omega-alpha term; consistent transformations between KE and PE.
- Preserve properties of differential operators.



Advection and divergence operators – each point talks to all eight neighboring points (isotropic)

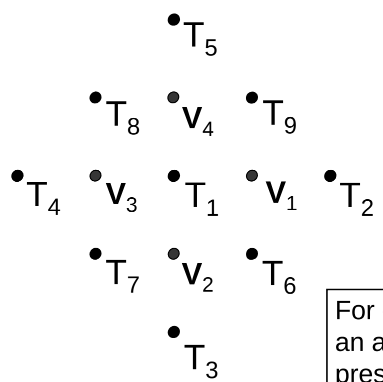


1/3 of contribution to divergence/advection comes from these N/S and E/W fluxes.



2/3 of contribution to divergence/advection comes from these diagonal fluxes.

Example of horizontal temperature advection in detail



For each T_n, there is an associated layer pressure depth (dp_n). There also is a dx_n specific to each point

Example of horizontal temperature advection in detail

Temperature fluxes in E/W, N/S, and diagonal directions:

$$TEW = u_3 dy\,(dp_1+dp_4)(T_1-T_4) + u_1 dy\,(dp_1+dp_2)(T_2-T_1)$$

$$TNS = v_2 dx_2\,(dp_1+dp_3)(T_1-T_3) + v_4 dx_4\,(dp_1+dp_5)(T_5-T_1)$$

$$TNE = [(u_1 dy + v_1 dx_1 + u_4 dy + v_4 dx_4)\,(dp_1+dp_9)\,(T_9-T_1) + (u_3 dy + v_3 dx_3 + u_2 dy + v_2 dx_2)\,(dp_1+dp_7)\,(T_1-T_7)]$$

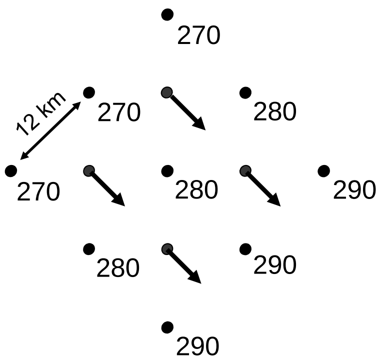
$$TSE = [(u_1 dy - v_1 dx_1 + u_2 dy - v_2 dx_2)\,(dp_1+dp_6)\,(T_6-T_1) + (u_3 dy - v_3 dx_3 + u_4 dy - v_4 dx_4)\,(dp_1+dp_8)\,(T_1-T_8)]$$

Advective tendency, ADT, combines the fluxes:

$$ADT = (TEW + TNS + TNE + TSE) * (-dt/24) * (1/dx_1 * dy * dp_1)$$

Example of horizontal temperature advection in detail

Consider pure 20 m/s northwesterly flow advecting colder temperatures to the center point of interest. Grid spacing is 12000 m.



Example of horizontal temperature advection in detail

$u_1=u_2=u_3=u_4=14.14$ m/s ; $v_1=v_2=v_3=v_4=-14.14$ m/s

To simplify, further assume that the layer pressure depth (dp) and spacing (dx) is the same at all points. Take dx=dy=8485.3 m, and dt=26.666 s.

The horizontal advective tendency, ADT, has contributions along 3 of 4 axes, and reduces to:

ADT = -.4444 K/time step

From the original schematic:

$-v \cdot \nabla T =$ -(20 m/s)*(10K /12000 m) = -.016666 K/s

-.016666 K/s * dt = -.4444 K/time step (agrees…phew)

NMM Vertical Coordinate

Pressure-sigma hybrid (Arakawa and Lamb, 1977)

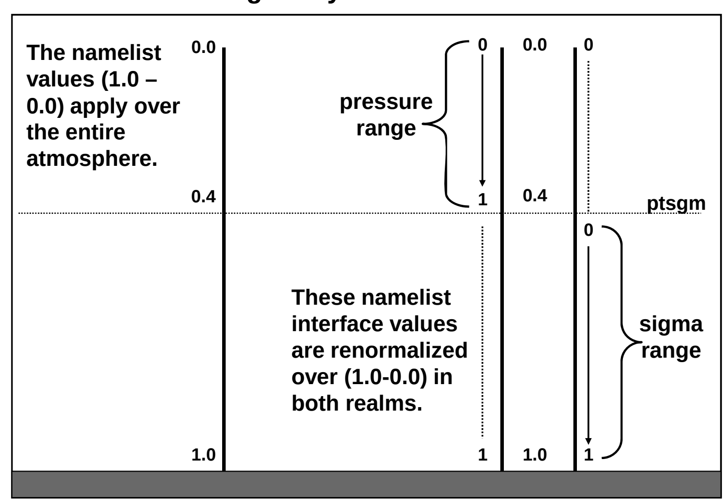
Has the desirable properties of a terrain-following, pressure coordinate:

- Exact mass (etc.) conservation
- Nondivergent flow on pressure surfaces
- No problems with weak static stability
- No discontinuities or internal boundary conditions

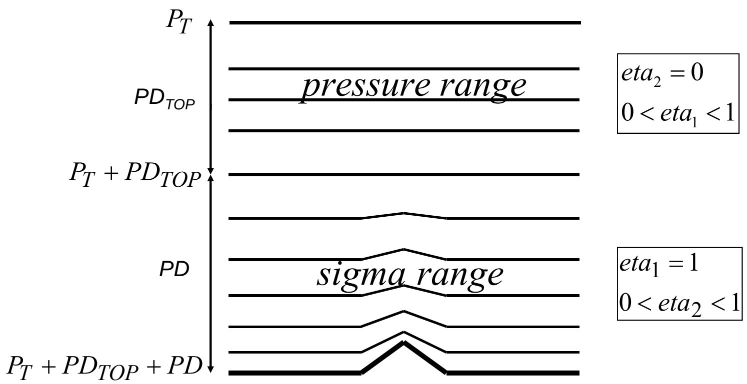
And an additional benefit from the hybrid:

- Flat coordinate surfaces at high altitudes where sigma problems worst (e.g., Simmons and Burridge, 1981)

Pressure-Sigma Hybrid Vertical Coordinate

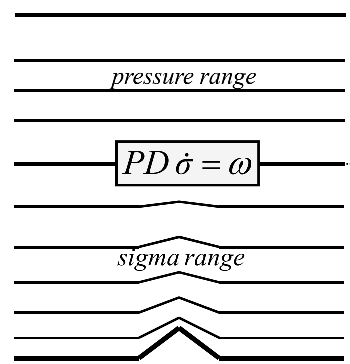


Pressure-Sigma Hybrid Vertical Coordinate



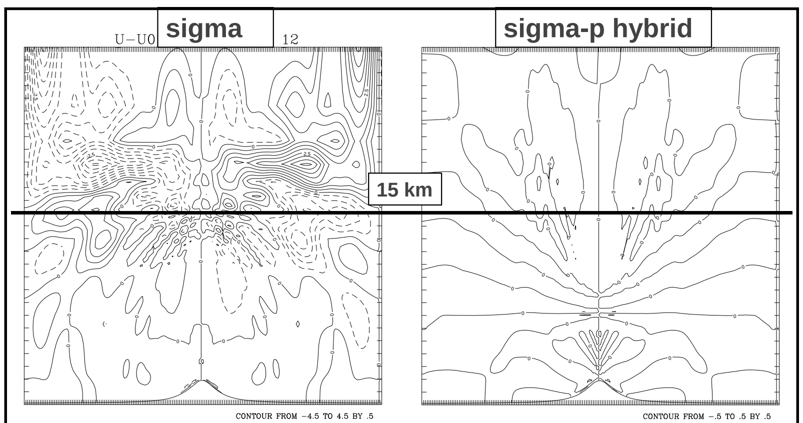
$$p = eta_1 * PD_{TOP} + eta_2 * PD + P_T$$

Equations in Hybrid Coordinate



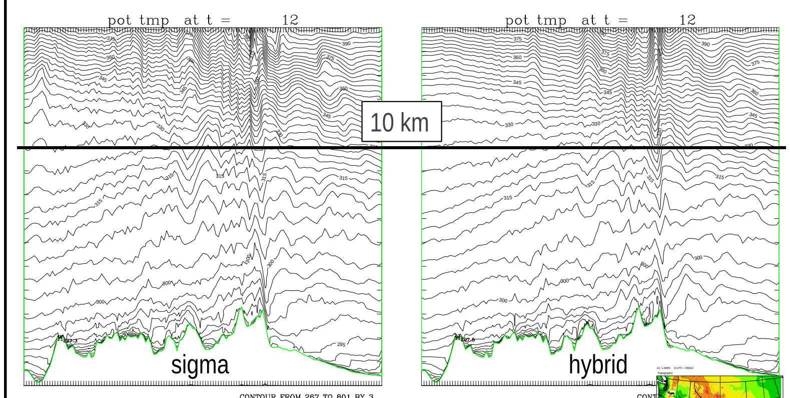
$$\nabla_p \bullet (\mathbf{v}) + \frac{\partial \omega}{\partial p} = 0$$

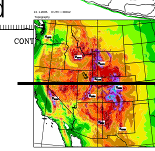
$$\frac{\partial PD}{\partial t} + \nabla_\sigma \bullet (PD\ \mathbf{v}) + \frac{\partial (PD\,\dot{\sigma})}{\partial \sigma} = 0$$



Wind component developing due to the spurious pressure gradient force in an idealized integration. The hybrid coordinate boundary between the pressure and sigma domains is at at about 400 hPa.

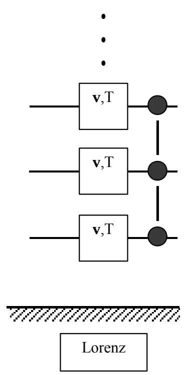
Example of nonphysical small scale energy source





12 h potential temperature forecasts (CINT=3 C) from 00Z January 13, 2005.

Vertical discretization



Vertical advection combines the advective fluxes computed above and below the layer of interest.

Boundary Conditions

- Lateral boundary information prescribed only on outermost row:

...	H	v	H	v	H	...	Pure boundary information
...	v	H	v	H	v	...	Average of four surrounding points (blend of boundary and interior)
...	H	v	H	v	H	...	Freely evolving
...	v	H	v	H	v	...	

- Upstream advection in three rows next to the boundary
  - No computational outflow boundary condition for advection
- Enhanced divergence damping close to the boundaries.

Dissipative Processes – lateral diffusion

A 2nd order, nonlinear Smagorinsky-type horizontal diffusion is utilized:

- Diffusion strength a function of the local TKE and 3D wind field deformation, gradients of the field being diffused, and a code-specified constant (COAC*).
- Lateral diffusion is zeroed for sloping model surfaces (> ~ 54 m / 12 km grid point).

* COAC has a default value of 1.6 and is specified in ./dyn_nmm/module_initialize_real.F. Larger values generate more diffusive smoothing.

Dissipative Processes - divergence damping

- Horizontal divergence damping with enhanced damping of the external mode.
- Internal mode damping (on each vertical layer)

$$v_j = v_j + \frac{(\nabla \cdot dp_{j+1}\mathbf{v}_{j+1} - \nabla \cdot dp_{j-1}\mathbf{v}_{j-1})}{(dp_{j+1} + dp_{j-1})} \cdot DDMPV$$

- External mode damping (vertically integrated)

$$v_j = v_j + \frac{(\int \nabla \cdot dp_{j+1}\mathbf{v}_{j+1} - \int \nabla \cdot dp_{j-1}\mathbf{v}_{j-1})}{(\int dp_{j+1} + \int dp_{j-1})} \cdot DDMPV$$

$$DDMPV \approx \sqrt{2} \cdot dt \cdot CODAMP$$

CODAMP is a code-specified parameter = 6.4 by default.

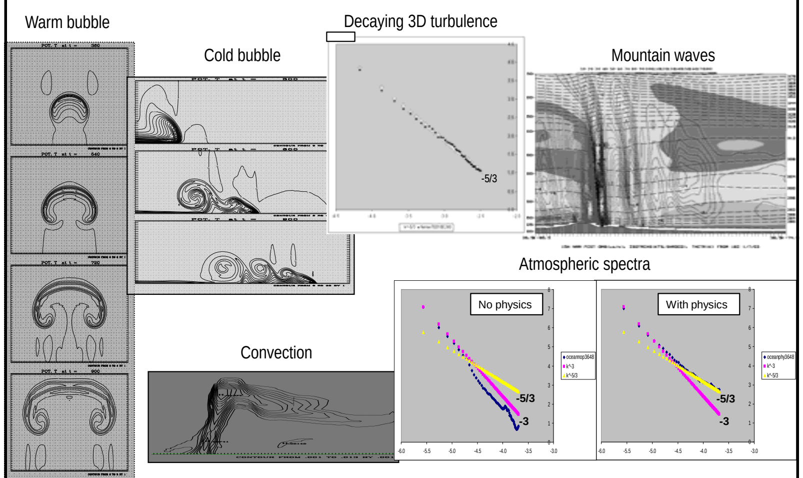
New for WRFV3.1

- Gravity Wave Drag & Mountain Blocking
  - Accounts for sub-grid scale mountain effects
  - Benefits overall synoptic patterns (especially in longer-range integrations), and near-surface wind and temperature forecasts.
  - Run within the WRF-NMM based NAM system at NCEP since March 2008.

Gravity Wave Drag (GWD) & Mountain Blocking (MB)

- Gravity wave drag (Alpert et al., 1988, 1996; Kim & Arakawa,1995)
  - Mountain wave stress, pressure drag
  - Vertical distribution of the wave stress, changes winds aloft (momentum deposition)
- "Mountain blocking" (Lott & Miller, 1997)
  - Wind flow around subgrid orogrography
  - Low-level flow is blocked below a dividing streamline (air flows around, not over barrier)

Dynamics formulation tested on various scales



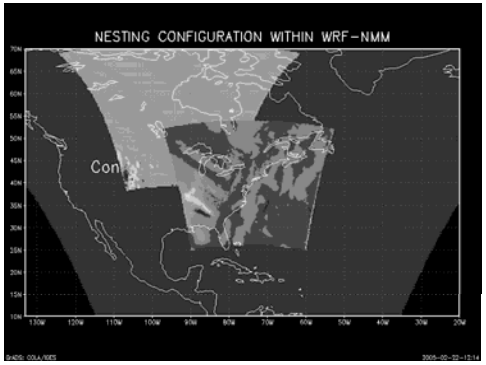
Summary

- Robust, reliable, fast
- NWP on near-cloud scales successful more frequently and with stronger signal than if only by chance
- Replaced the Eta as NAM at NCEP on June 20, 2006
- Near-cloud-scale runs (~4 km grid spacing) operational at NCEP for severe weather forecasting
- Operational as Hurricane WRF in 2007
- Operational and quasi-operational elsewhere.

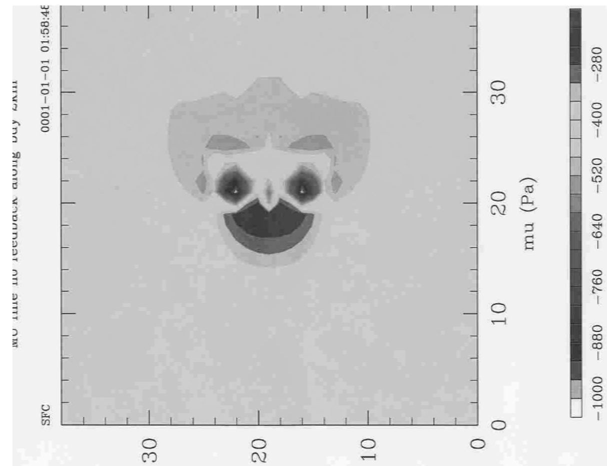
Nesting in WRF

WRF Nesting

Dave Gill
Matthew Pyle



ARW Nesting



Nesting Basics - What is a nest

- A nest is a ***finer-resolution*** model run. It may be ***embedded*** simultaneously within a coarser-resolution (parent) model run, or ***run independently*** as a separate model forecast.
- The nest ***covers a portion*** of the parent domain, and is driven along its ***lateral boundaries*** by the parent domain.
- Nesting enables running at finer resolution without the following problems:
  - Uniformly high resolution over a large domain - prohibitively expensive
  - High resolution for a very small domain with mismatched time and spatial lateral boundary conditions

Nesting Basics - NMM

- The focus is on static, one- or two-way nesting
  - Static: The nest location is fixed in space
  - One-way: Information exchange between the parent and the nest is strictly down-scale. The nest solution does not feedback to the coarser/parent solution.
  - Two-way: Information exchange between the parent and the nest is bi-directional. The nest feedback impacts the coarse-grid domain’s solution.
  - Fine grid input is for non-meteorological variables.

Nesting Basics - ARW

- One-way nesting via multiple model forecasts
- One-way nesting with a single model forecast, without feedback

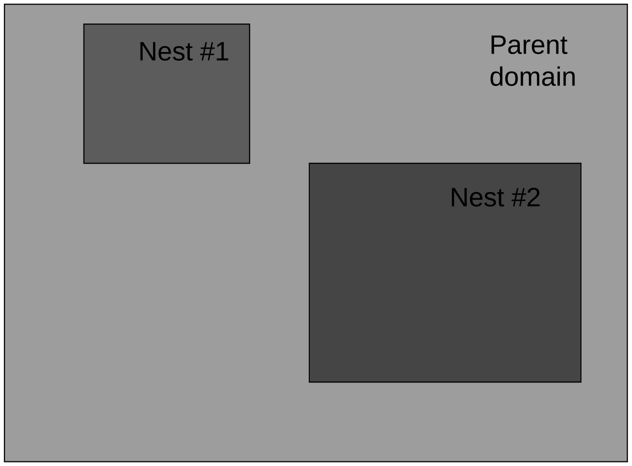
- One-way/two-way nesting with a single input file, all fields interpolated from the coarse grid
- One-way/two-way nesting with multiple input files, each domain with a full input data file
- One-way/two-way nesting with the coarse grid data including all meteorological fields, and the fine-grid domains including only the static files

- One-way/two-way nesting with a specified move for each nest
- One-way/two-way nesting with a automatic move on the nest determined through 500 mb low tracking

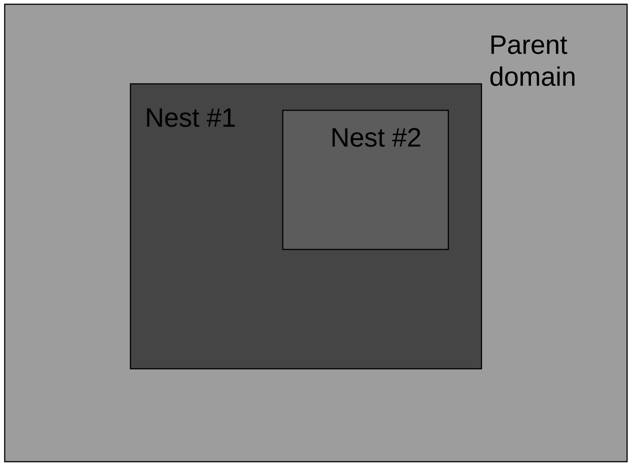
Some Nesting Hints

- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting

Two nests on the same "level", with a common parent domain

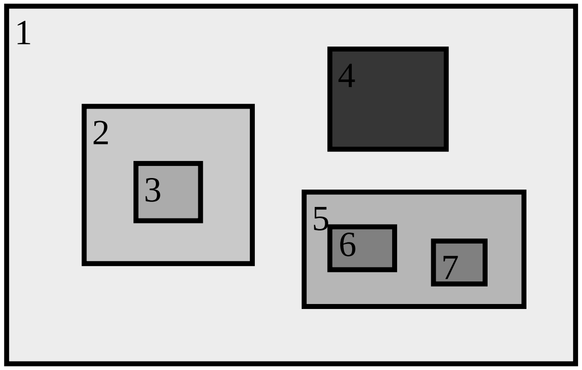


Two levels of nests, with nest #1 acting as the parent for nest #2



These are all OK

Telescoped to any depth
Any number of siblings

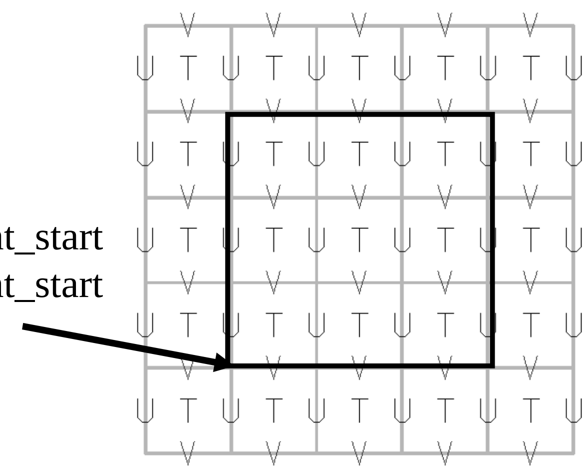


Some Nesting Hints

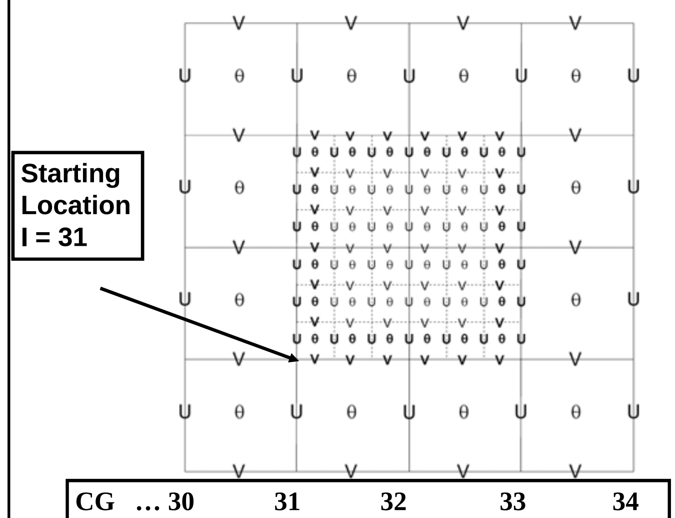
- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting

ARW Coarse Grid Staggering

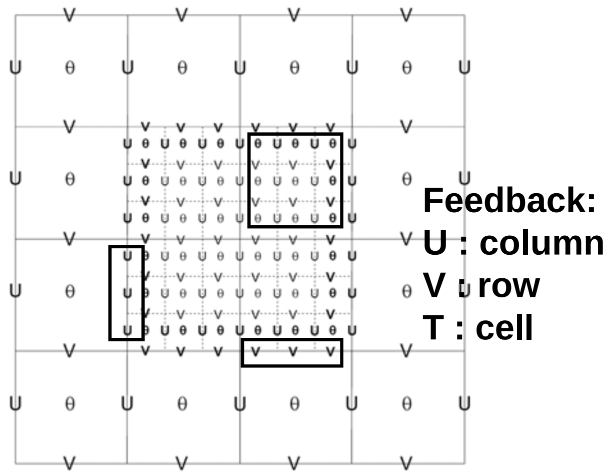
i_parent_start
j_parent_start



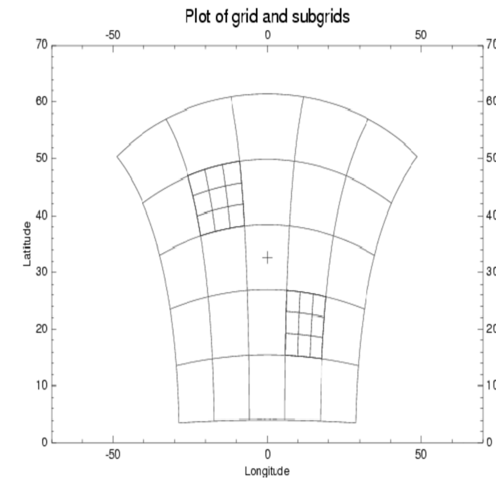
ARW Coarse Grid Staggering 3:1 Ratio



ARW Coarse Grid Staggering 3:1 Ratio



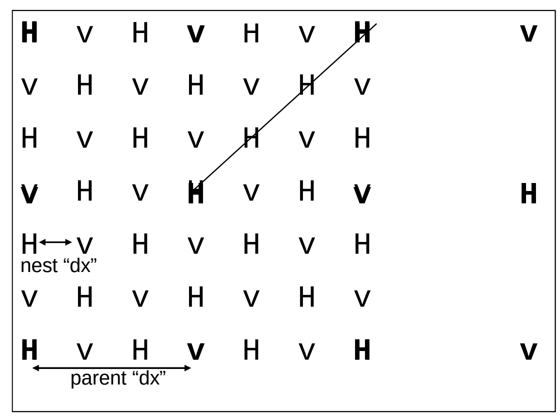
NMM Coarse/Fine Overlay



NMM Telescopic E-Grid

- Interpolations are done on the rotated latitude/longitude projection. The fine grid is coincident with a portion of the high-resolution grid that covers the entire coarse grid.
- The nested domain can be placed anywhere within the parent domain and the nested grid cells will exactly overlap the parent cells at the coincident cell boundaries.
- Coincident parent/nest grid points eliminate the need for complex, generalized remapping calculations, and enhances model performance and portability.
- The grid design was created with moving nests in mind.

An odd grid ratio introduces parent/nest points being coincident, and a 3:1 ratio is preferred as it has been extensively tested.

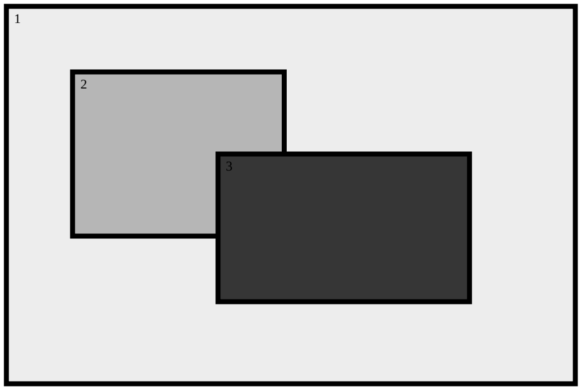


Some Nesting Hints

- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting

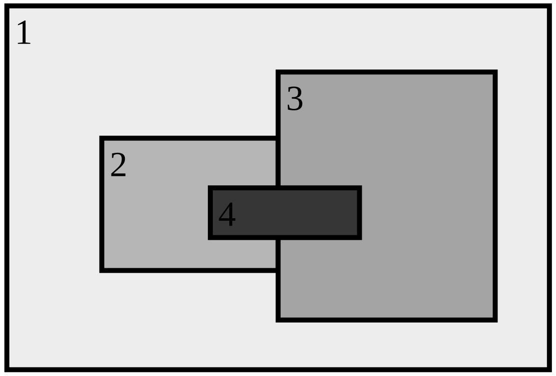
Not OK for 2-way

Child domains *may not* have overlapping points in the parent domain (1-way nesting excluded).



Not OK either

Domains have one, and only one, parent -
(domain 4 is NOT acceptable even with 1-way nesting)



Some Nesting Hints

- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting

Nesting Performance

- The size of the nested domain may need to be chosen with computing performance in mind.
- Assuming a 3:1 ratio and the same number of grid cells in the parent and nest domains, the fine grid will require 3x as many time steps to keep pace with the coarse domain.
- A simple nested domain forecast is approximately 4x the cost of just the coarse domain.
- Don't be *cheap* on the coarse grid, 2x as many CG points in only a 25% nested forecast time increase.

NMM: Initial Conditions

- Simple horizontal bilinear interpolation of the parent initial conditions is used to initialize all meteorological fields on the nest.
- A nearest-neighbor approach is adopted for prescribing most of the land-state variables.
- Topography and land-sea mask are redefined over the nested domain using the appropriate "nest level" of WPS info from geogrid.
- Quasi-hydrostatic mass balancing is carried out after introducing the high-resolution topography.

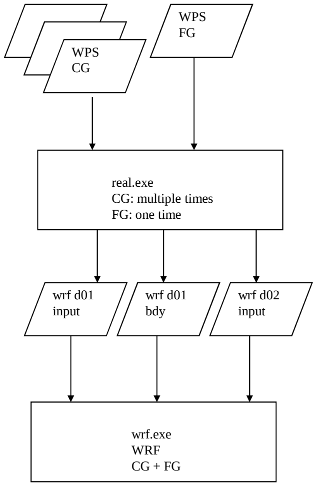
ARW: 2-Way Nest with 2 Inputs

Coarse and fine grid domains must start at the same time, fine domain may end at any time

Feedback may be shut off to produce a 1-way nest (cell face and cell average)

Any integer ratio for coarse to fine is permitted, odd is usually chosen for real-data cases

Options are available to ingest only the static fields from the fine grid, with the coarse grid data horizontally interpolated to the nest



ARW: 2-Way Nest with 2 Inputs

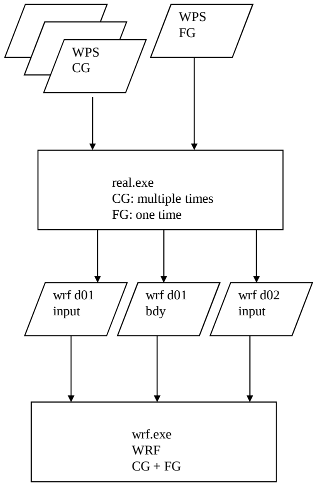
No vertical nesting

Usually the same physics are run on all of the domains (excepting cumulus)

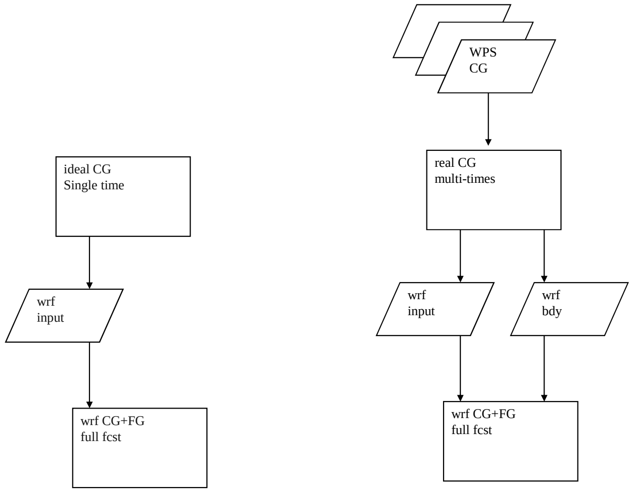
The grid distance ratio is not strictly tied to the time step ratio

Topography smoothly ramps from coarse grid to the fine grid along the interface along the nest boundary

All fine grids must use the nested lateral boundary condition



ARW: 2-Way Nest with 1 Input



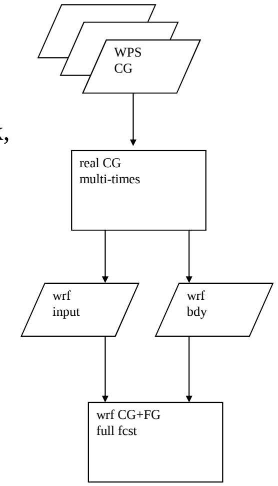
ARW: 2-Way Nest with 1 Input

A single namelist column entry is tied to each domain

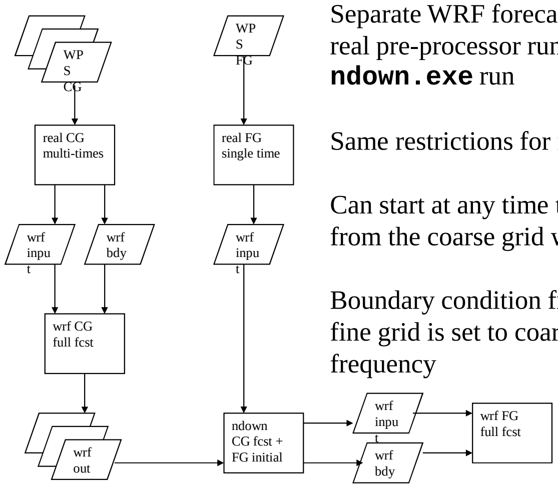
The horizontal interpolation method, feedback, and smoothing are largely controlled through the Registry file

For a 3:1 time step ratio, after the coarse grid is advanced, the lateral boundaries for the fine grid are computed, the fine grid is advanced three time steps, then the fine grid is fed back to the coarse grid (recursively, depth first)

Helpful run*.tar files are located in the **`./WRFV3/test/em_real`** directory



ndown: 1-Way Nest with 2 Inputs



Separate WRF forecast runs, separate real pre-processor runs, intervening **`ndown.exe`** run

Same restrictions for nest ratios

Can start at any time that an output time from the coarse grid was created

Boundary condition frequency for the fine grid is set to coarse grid output frequency

Some Nesting Hints

- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting
- Nest logic in WRF source code
- Nest information in the Registry

Allocate and Initialize a Nest

```
DO WHILE ( nests_to_open( grid , nestid , kid ) )
  a_nest_was_opened = .true.
  CALL med_pre_nest_initial ( grid , nestid , &
         config_flags )
  CALL alloc_and_configure_domain ( &
         domain_id  = nestid ,   &
         grid       = new_nest , &
         parent     = grid ,     &
         kid        = kid        )
   CALL Setup_Timekeeping (new_nest)
   CALL med_nest_initial ( grid , new_nest, &
         config_flags )
END DO
```

All Siblings get Processed

```
DO WHILE ( ASSOCIATED( grid_ptr ) )
  CALL set_current_grid_ptr( grid_ptr )
  CALL solve_interface ( grid_ptr )
  CALL domain_clockadvance ( grid_ptr )
  CALL domain_time_test( grid_ptr, &
  'domain_clockadvance' )
  grid_ptr => grid_ptr%sibling
END DO
```

Recursive Nest Depth

```
DO kid = 1, max_nests
  IF ( ASSOCIATED( grid_ptr%nests(kid)%ptr ) ) THEN
    CALL set_current_grid_ptr( grid_ptr%nests(kid)%ptr )
    CALL med_nest_force ( grid_ptr , &
         grid_ptr%nests(kid)%ptr , config_flags )
    grid_ptr%nests(kid)%ptr%start_subtime = &
         domain_get_current_time(grid) - &
         domain_get_time_step(grid)
    grid_ptr%nests(kid)%ptr%stop_subtime = &
         domain_get_current_time(grid)
    CALL integrate ( grid_ptr%nests(kid)%ptr )
    CALL med_nest_feedback ( grid_ptr , &
         grid_ptr%nests(kid)%ptr , config_flags )
  END IF
END DO
```

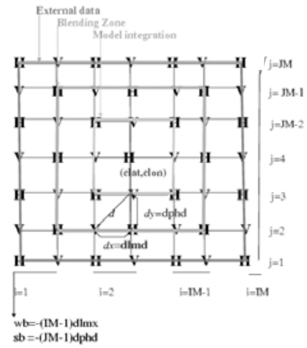
Input vs Interpolating

```
CALL med interp domain( parent, nest )

CALL init_domain_constants ( parent, nest )

IF ( nest_config_flags%input_from_file ) THEN

   IF ( nest_config_flags%input_from_file ) THEN
      CALL med_initialdata_input_ptr( nest , &
         nest_config_flags )
   ENDIF
```

Feedback and Domain Sync-ing

```
CALL med_nest_feedback ( parent , nest , &
  config_flags )

CALL start_domain ( nest , .TRUE. )

CALL start_domain ( parent , .TRUE. )
```

NMM Nested LBCs



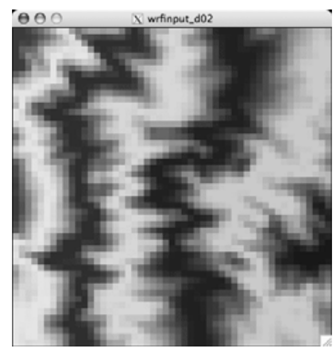
* Given wb,sb, clat and clon, the above rotated lat-lon grid system can be transformed to a lat-lon grid system.

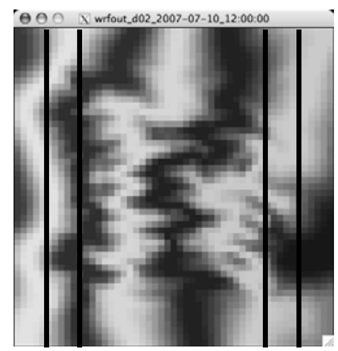
- Nest boundaries generally are treated in the same way as the standard parent domain boundaries:
  - outermost row is prescribed
  - two rows in from boundary is freely integrating
  - in between is a blending zone (average of outermost and freely integrating points)
- The one key difference is frequency of boundary updates: *nested boundaries are updated at every time step of the parent domain.*

NMM Mass Balancing for LBCs

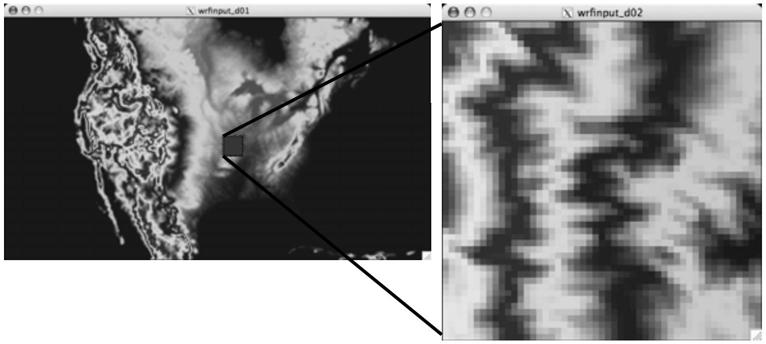
- The parent domain geopotential height, temperature, and moisture are all vertically interpolated (cubic splines) from the hybrid surfaces onto standard isobaric levels.
- Using horizontally interpolated information of the height field from the parent domain, and high-resolution topography from the nest level, mass is adjusted and revised hybrid surfaces are constructed.
- T and q: 1) horizontally interpolated to the nest domain on standard pressure levels, 2) vertically interpolated onto the new hybrid surfaces
- Approach produces an effective way of updating the nest interface without much distortion or noise

ARW Lateral Smoothing





Intermediate Domains



The intermediate domain between a parent and a child is the resolution of the coarse grid over the size of the fine grid. It allows the model to re-decompose the domain among all of the processors.

Intermediate Domains - Part 1

```
grid => nested_grid%intermediate_grid
CALL alloc_space_field ( grid, grid%id , 1 , 2 , .TRUE. ,
grid => parent_grid
CALL model_to_grid_config_rec ( grid%id , &
   model_config_rec , config_flags )
CALL couple_or_uncouple_em ( grid , config_flags ,.true. &
#         include "em_actual_new_args.inc" )
grid => nested_grid
CALL model_to_grid_config_rec ( grid%id , &
   model_config_rec , config_flags )
CALL couple_or_uncouple_em ( grid , config_flags ,.true. &
#         include "em_actual_new_args.inc" )
```

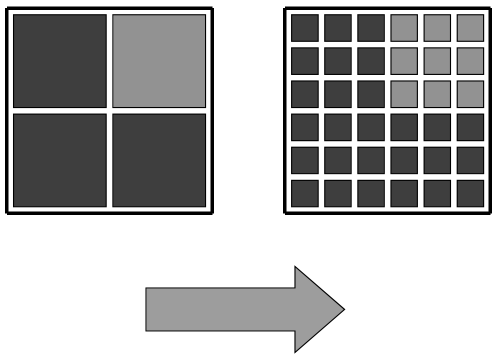
Intermediate Domains - Part 1

```
grid => parent_grid
CALL model_to_grid_config_rec ( grid%id ,
   model_config_rec , config_flags )
CALL interp_domain_em_part1 ( grid , &
   nested_grid%intermediate_grid, nested_grid, &
   config_flags   &
#         include "em_actual_new_args.inc" )
```

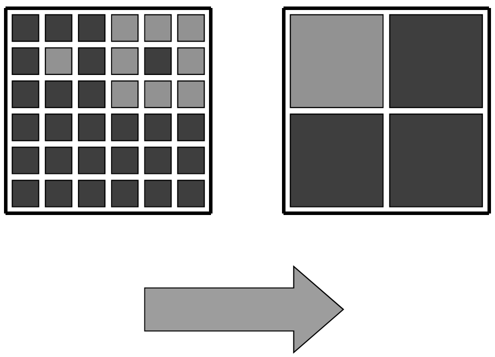
Intermediate Domains - Part 2

```
grid => nested_grid%intermediate_grid
CALL model_to_grid_config_rec ( nested_grid%id , &
   model_config_rec , config_flags )
CALL force_domain_em_part2 ( grid, nested_grid, &
   config_flags   &
#          include "em_actual_new_args.inc")

grid => nested_grid
CALL model_to_grid_config_rec ( grid%id , &
   model_config_rec , config_flags )
CALL couple_or_uncouple_em ( grid , config_flags ,.false. &
#          include "em_actual_new_args.inc" )

grid => parent_grid
CALL model_to_grid_config_rec ( grid%id , &
   model_config_rec , config_flags )
CALL couple_or_uncouple_em ( grid , config_flags ,.false. &
#          include "em_actual_new_args.inc" )
```

ARW Masked Interpolation



ARW Masked Feedback



Some Nesting Hints

- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting
- Nest logic in WRF source code
- Nest information in the Registry

What are those "usdf" Options

```
state real u ikjb dyn_em 2 X \
  i01rhusdf=(bdy_interp:dt) \
  "U" "x-wind component" "m s-1"
```

"f" defines what lateral boundary forcing routine (found in **share/interp_fcn.F**) is utilized, colon separates the additional fields that are required (fields must be previously defined in the Registry)

What are those “usdf” Options

```
state real landmask ij misc 1 -  \
  i012rhd=(interp_fcnm)u=(copy_fcnm)\
  "LANDMASK" "LAND MASK (1=LAND, 0=WATER)"
```

“u” and “d” define which feedback (up-scale) and horizontal interpolation (down-scale) routines (found in `share/interp_fcn.F`) are utilized

Default values (i.e. not a subroutine name listed in the parentheses) assume non-masked fields

At compile-time, users select options

What are those “usdf” Options

```
state real ht ij  misc  1  - i012rhdus  "HGT" \
  "Terrain Height"   "m"
```

“s” if the run-time option for smoothing is activated, this field is to be smoothed - only used for the parent of a nest domain, smoothing is in the area of the nest, excluding the outer row and column of the nest coverage

Whether or not smoothing is enabled is a run-time option from the namelist

Special IO Stream #2 Fields

```
state real msft ij  misc  1 -  \
  i012rhdu=(copy_fcnm)  "MAPFAC_M"   \
  "Map scale factor on mass grid" ""

state real msfu ij  misc  1 X  \
  i012rhdu=(copy_fcnm)  "MAPFAC_U"   \
  "Map scale factor on u-grid" ""

state real msfv ij  misc  1 Y  \
  i012rhdu=(copy_fcnm)  "MAPFAC_V"   \
  "Map scale factor on v-grid" ""
```

Post-processing Tools (1):
NCL

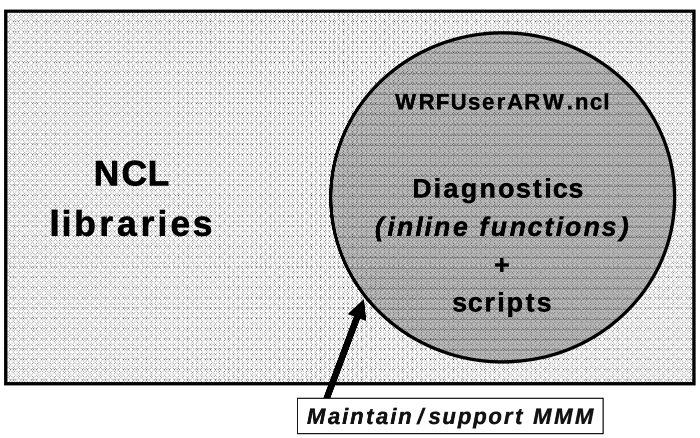
Post-processing Tools: NCL
(WRF-ARW Only)

Cindy Bruyère

NCL

- **NCAR Command Language**
- ***http://www.ncl.ucar.edu***
- **Read WRF-ARW data directly**
- **Generate a number of graphical plots**
  - *Horizontal, cross-section, skewT, meteogram, panel*

NCL & WRF



Download NCL

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

Home Directory → ~/.hluresfile ← Very Important

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

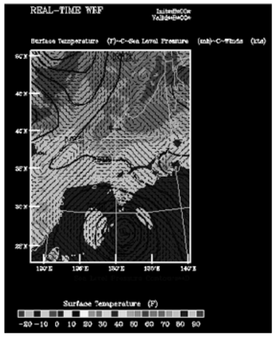
- ***http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml***

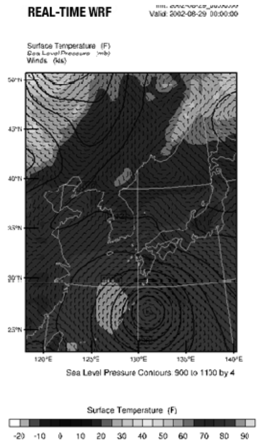
~/.hluresfile

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

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

~/.hluresfile



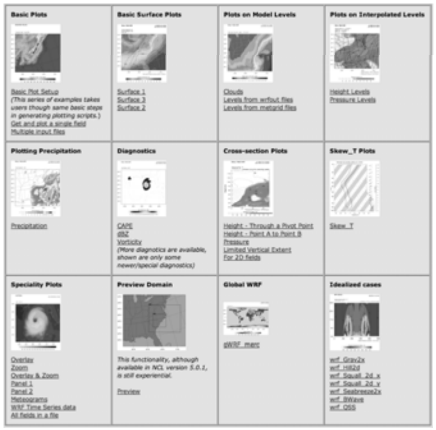


Generate Plots

- **Create a script**
  - *wrf_real.ncl*
- **Set NCARG_ROOT environment variable:**
  - *setenv NCARG_ROOT /usr/local/ncl*
- **Ensure you have an ~/.hluresfile file**
- **Run NCL script**
  - *ncl wrf_real.ncl*

Generate Plots: *A good start - OnLine Tutorial*

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

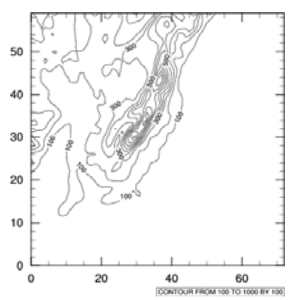


Creating a Plot : NCL script

```
load ncl library scripts

begin

 ; Open graphical output
 ; Open input file(s)

 ; Read variables

 ; Set up plot resources & Create plots
 ; Output graphics

end
```

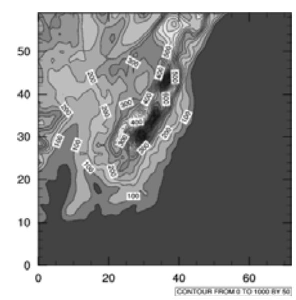
Generate Plots

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

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

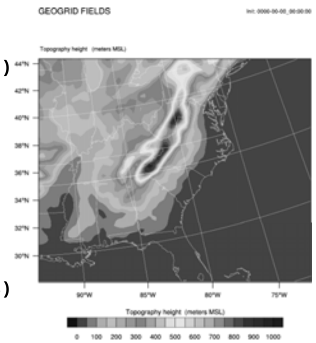


Generate Plots

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

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



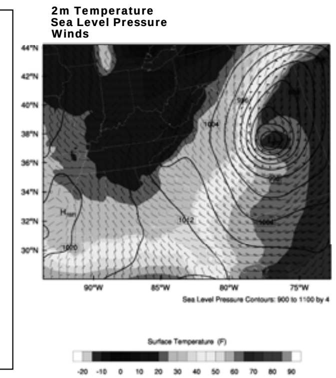
Generate Plots




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

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

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

Creating a Plot : NCL script

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

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

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



Functions

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

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

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

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

WRF / Built-in Functions

Use NCL inline functions, in place of wrf_user_getvar

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

WRF / Built-in Functions

Use NCL inline functions, in place of wrf_user_getvar

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

WRF / Built-in Functions

Use NCL inline functions, in place of wrf_user_getvar

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

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

Special WRF Functions

- **wrf_user_getvar**
  Get fields from input file

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

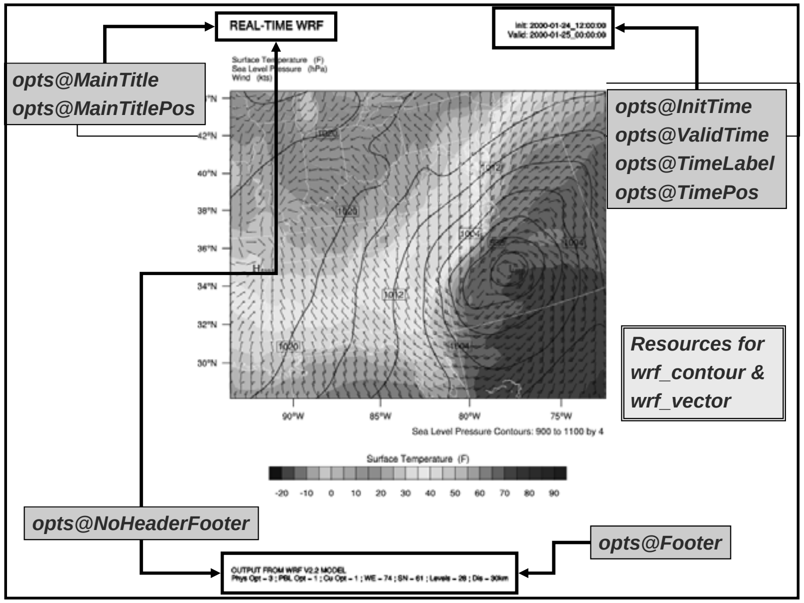
  Diagnostics
  avo/pvo: Absolute/Potential Vorticity, ***cape_2d***: 2D mcape/mcin/lcl/lfc, ***cape_3d***: 3D cape/cin, ***dbz/mdbz***: Reflectivity, ***geopt/geopotential***: Geopotential, ***p/pres/pressure***: Pressure**,** ***rh/rh2***: Relative Humidity, ***slp***: Sea Level Pressure, ***td/td2***: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

Special WRF Functions

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

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

  opts@MainTitle : Main title on the plot.
  opts@MainTitlePos : Main title position
  opts@NoHeaderFooter : Switch off Headers & Footers.
  opts@Footer : Add model information as a footer.
  opts@InitTime : Plot initial time on graphic.
  opts@ValidTime : Plot valid time on graphic.
  opts@TimeLabel : Valid time.
  opts@TimePos : Time position.
  opts@ContourParameters : Countour parameters.
  opts@FieldTitle : Overwrite the field title.
  opts@UnitLabel : Overwrite the field units.
  opts@PlotLevelID : Add level information to field title.
  opts@NumVectors : Density of wind vectors. *(wrf_vector)*



Special WRF Functions

wrf_map_overlays / wrf_overlays
Overlay plots created with *wrf_contour* and *wrf_vector*

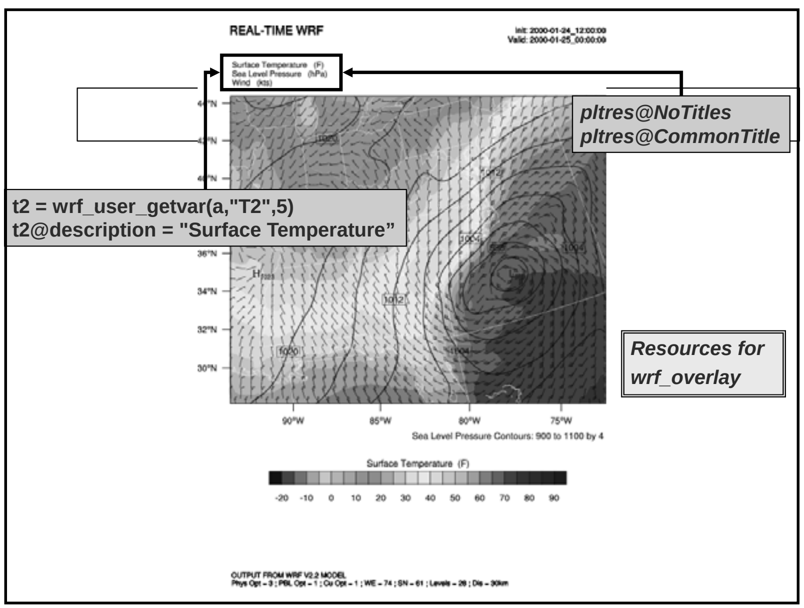
*plot = wrf_map_overlays (a, wks, \
(/contour,vector/), pltres, mpres)*

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

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

To Zoom set:
mpres@ZoomIn = True, and
mpres@Xstart, mpres@Xend, mpres@Ystart, mpres@Yend, to the corner x/y positions of the zoomed plot.

pltres@NoTitles; pltres@CommonTitle; pltres@PlotTitle;
pltres@PanelPot; pltres@FramePlot



Special WRF Functions

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

  times = wrf_user_list_times (a)

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

  map = wrf_map(a, wks, opts)

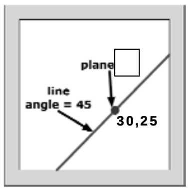
- **wrf_user_unstagger (varin, unstagDim)**
  Unstaggers an array

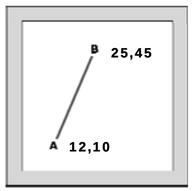
  ua = wrf_user_unstagger (U, “X”)
  ua = wrf_user_getvar(a,”ua”,time)

Special WRF Functions

- **wrf_user_intrp3d**
  Interpolate horizontally to a given pressure or height level
  Interpolate vertically *(pressure/height)*, along a given line
  tc_plane = wrf_user_intrp3d(tc, p, "v", (/30,25/), 45., False)

- **wrf_user_intrp2d**
  Interpolate along a given line
  t2_plane = wrf_user_intrp2d(t2, (/12,10, 25,45/), 0., True)





Special WRF Functions

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

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

  res@useTime - Default is 0
  Set if want the reference longitude/latitudes must come from a specific time - one will only use this for moving nest output which has been stored in a single file.

  res@returnInt - Default is True
  If set to False, the return values will be real.
  (*wrf_user_ll_to_ij only)*

WRF / Built-in Functions

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

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

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

res@KNOWNI     = 37.0 }  Center of domain {74 x 64}
res@KNOWNJ     = 30.5 }
xx = 1.0
yy = 1.0
loc = wrf_ij_to_ll (xx,yy,res)
```

Special WRF Functions

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

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

Resources

- The special WRF functions have unique resources: http://www.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_functions.htm
- All general NCL resources can also be used to control the plot: http://www.ncl.ucar.edu/Document/Graphics/Resources

Adding FORTRAN code

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

myTK.f

```
subroutine compute_tk (tk,pressure,theta, nx, ny, nz)
  implicit none
  integer nx,ny,nz
  real    pi, tk(nx,ny,nz)
  real    pressure(nx,ny,nz), theta(nx,ny,nz)

integer i,j,k

  do k=1,nz
    do j=1,ny
    do i=1,nx
       pi=(pressure(i,j,k) / 1000.)**(287./1004.)
       tk(i,j,k) = pi*theta(i,j,k)
    enddo
    enddo
  enddo

  end
```

myTK.f

```
C NCLFORTSTART
subroutine compute_tk (tk,pressure,theta, nx, ny, nz)
  implicit none
  integer nx,ny,nz
  real    pi, tk(nx,ny,nz)
  real    pressure(nx,ny,nz), theta(nx,ny,nz)
C NCLEND

 integer i,j,k

  do k=1,nz
    do j=1,ny
    do i=1,nx
       pi=(pressure(i,j,k) / 1000.)**(287./1004.)
       tk(i,j,k) = pi*theta(i,j,k)
    enddo
    enddo
  enddo

  end
```

myTK.so

WRAPIT myTK.f

setenv NCARG_ROOT */usr/local/ncl*

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

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

edit your copy

./WRAPIT myTK.f

myTK.so - use in NCL script

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”
external myTK "./myTK.so"

begin

      t = wrf_user_getvar(a,”T”,5)
      t = t + 300
      p = wrf_user_getvar(a,”pressure”,5)

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

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

end
```

FORTRAN 90 code

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

FORTRAN 90 code

myTK.f90

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

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

end subroutine compute_tk
```

FORTRAN 90 code

myTK.f90

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

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

end subroutine compute_tk
```

myTK90.stub

```
C NCLFORTSTART
  subroutine compute_tk (tk, pres, theta, nx, ny, nz)
  implicit none
  integer nx,ny,nz
  real    tk(nx,ny,nz)
  real    pres(nx,ny,nz), theta(nx,ny,nz)
C NCLEND
```

myTK90.so

WRAPIT myTK90.stub myTK.f90

setenv NCARG_ROOT */usr/local/ncl*

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

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

edit your copy - link to a f90 compiler

./WRAPIT myTK.f

myTK90.so - use in NCL script

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
external myTK90 "./myTK90.so"

begin

    t = wrf_user_getvar(a,"T",5)
    t = t + 300
    p = wrf_user_getvar(a,"pressure",5)

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

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

end
```

Post-processing Tools (2):
WPP

NCEP's WRF POST PROCESSOR (WPP)

Hui-Ya Chuang

Outline

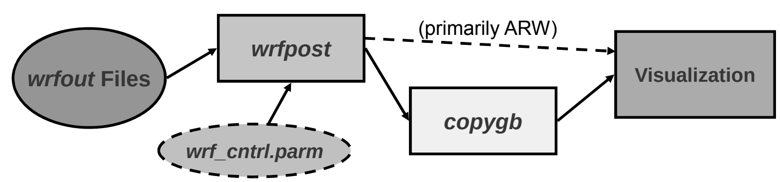
- Overview
- Components and Functions
- Sample fields generated
- Installation
- Running *wrfpost*
  - Controlling output generation
- Running *copygb*
  - Specifying target grid
- Visualization

The critical big picture overview

- Processes model output from <u>both</u> the NMM and the ARW <u>dynamical cores</u>.
- The WRF post processor (WPP) <u>generates</u> output in <u>GRIB</u>.
- The WPP enables product generation on <u>any output grid</u>.

Components of the WPP

The WPP has two components: wrfpost and copygb.



Functions and features of *wrfpost*

- Performs **vertical** interpolation onto isobaric and other non-model surfaces
- Computes diagnostic fields
- Destaggers wind onto mass points (ARW)
- An MPI-parallel code

Functions of *copygb*

- Performs **horizontal** interpolation and de-staggering (NMM core) onto a defined output grid
  - Many visualization packages cannot properly handle staggered grids, so copygb is an important step for processing NMM core output (optional for ARW).
- Useful for both cores in creating an output grid not fixed by the model integration domain.

Ingesting WRF model output

- wrfpost reads in WRF model output in either binary or netCDF format using the WRF I/O package.
- Users are encouraged to use netCDF formatted model output for simplicity. NCEP uses binary output for speed.

Ingesting WRF model output

- The model fields read in by *wrfpost* for both dynamical cores can be found in your user guide (listed by WRF Registry file variable names) .
- These fields are automatically provided by the default WRF model Registry files.

Fields generated by the WPP

- The WPP currently outputs 288 fields.
  - Complete list in the Post Processing Utilities Chapter of the user guide
- Sample fields generated by WPP:

1) T, Z, humidity, wind, cloud water, cloud ice, rain, and snow on isobaric levels
2) Shelter level T, humidity, and wind fields
3) SLP (two kinds)
4) Precipitation-related fields

Fields generated by the WPP

- Sample fields generated by WPP (cont.):

5) PBL-related fields
6) Diagnostic fields
7) Radiative fluxes
8) Surface fluxes
9) Cloud related fields
10) Aviation products

Computation of fields

- Documentation for how most fields are computed can be found in ETA post documentation online:

http://www.emc.ncep.noaa.gov/mmb/papers/chuang/1/OF438.html

- A field not included in the online documentation is simulated radar reflectivity. Different algorithms are used depending on the microphysics (MP) option used in the model run:
  - Ferrier MP scheme: consistent with assumptions made in Ferrier MP scheme [details in Ferrier,1994: *J. Atmos. Sci*, **51**, 249-280].
  - Other MP schemes: adopted from RIP4. More information can be found online:

    http://www.mmm.ucar.edu/wrf/users/docs/ripug.htm

WPP download and compile

Downloading the WPP source code

- The WPP source code can be obtained from:
  http://www.dtcenter.org/wrf-nmm/users/downloads
- The latest version available is:
  wrfpostproc_v3.0.1.tar.gz
- Unpack the downloaded file:
  tar –zxvf wrfpostproc_v3.0.1.tar.gz
- *cd* to newly created WPPV3/ directory

WPPV3 directory contents

- **sorc**/: source codes
- **scripts**/: sample scripts for running WPP and generating graphics
- **lib**/: libraries used in the build
- **parm**/: control file used when running the wrfpost
- **configure**: sets up makefiles based on user-specified computing platform and paths to software
- **makefile**: master makefile to compile lib/ and sorc/

Compile source codes

- Prepare master makefile to build WPP with on your computer by executing the configure file:
  ./configure
- At the prompt, specify:
  1) platform: "1" for LINUX (pg compiler); "2" for LINUX (ifort compiler); "3" for AIX/IBM.
  2) path to a netCDF installation
  3) path to a compiled WRF model source directory
- Compile all libraries and source code by executing the master makefile in the top directory:
  make >& compile_wpp.log &

Compile source codes (cont.)

- If compilation is successful, these three executables will be present in exec/ :

  copygb.exe
  ndate.exe
  wrfpost.exe

Running wrfpost and copygb

wrfpost needs three input files to run:

– *itag*: specifies details of WRF model output to process

wrfout_d01_2005-04-27_00:00:00	← *WRF history filename*
netcdf	← *WRF output format (netcdf/binary)*
2005-04-27_00:00:00	← *validation time*
NMM	← *model name (NMM/NCAR)*

– *wrf_cntrl.parm*: control file specifying fields to output

– *eta_micro_lookup.dat*: binary look-up table for Ferrier MP

* In the sample run_wrfpost* scripts, these files are generated on the fly or are automatically linked.

wrfpost control file: *wrf_cntrl.parm*

- Users specify which fields or which level(s) of fields to output by modifying control file, e.g.,

```
(PRESS ON MDL SFCS  ) SCAL=(6.0)
L=(11000 00000 00000 00000 00000 00000 00000…
(HEIGHT ON MDL SFCS ) SCAL=(6.0)
L=(11000 00000 00000 00000 00000 00000 00000…
```

GRIB packing precision (SCAL value)

Each column represents a single/model/isobaric/ level: "1" = output, "0" = no output

Product description – wrfpost code keys on these character strings.

wrfpost control file: *wrf_cntrl.parm*

- The included wrf_cntrl.parm file has entries for every possible output field.
- The "Fields produced by *wrfpost*" table in the user's guide may help understand the character string abbreviations used in the control file.

Outputting fields on different vertical coordinates

- *wrfpost* outputs on several vertical coordinates:
  - Native model levels
  - 47 isobaric levels
  - 7 flight levels above MSL: 914, 1524, 1829, 2134, 2743, 3658, and 6000 m
  - 6 PBL layers: each averaged over 30 hPa AGL layer
  - 2 AGL levels: 1000 & 4000 m (radar reflectivity).
- Except for AGL and isobaric levels, vertical levels are counted from the ground surface up in *wrf_cntrl.parm*.

Examples

- Output T every 50 hPa from 50 hPa to 1000 hPa:

```
(TEMP ON PRESS SFCS  ) SCAL=( 3.0)
L=(00000 01001 01010 10101 01010 10101 01010 10101 01010 10000…)
```

From left to right, the isobaric levels increase 2, 5, 7, 10, 20, 30, 50, 70, then 75-1000 hPa every 25 hPa.

- Output instantaneous surface sensible heat flux:

```
(INST SFC SENHEAT FX ) SCAL=( 3.0)
L=(10000 00000 00000 00000 00000 00000 00000 00000 00000 00000…)
```

Examples

- Do not output cloud top height:

```
(CLOUD TOP HEIGHT    ) SCAL=( 3.0)
L=(00000 00000 00000 00000 00000 00000 00000 00000 00000 00000…)
```

- Output the U-wind component at the 5 lowest model levels:

```
(U WIND ON MDL SFCS  ) SCAL=( 4.0)
L=(11111 00000 00000 00000 00000 00000 00000 00000 00000 00000…)
```

copygb target grid definition

- The generic command to run copygb and horizontally interpolate onto a new grid is:

  copygb.exe –xg"${grid}" in.grb out.grb

- Three options on how to specify the target $grid:
  1. Pre-defined NCEP standard grid number
  2. Grid navigation file created by *wrfpost* (NMM only)
  3. User-defined grid definition

Run *copygb* – Option 1

- Interpolate to a pre-defined NCEP standard grid (restrictive but simple)
  - For example, to interpolate onto NCEP grid 212:
    copygb.exe –xg212 in.grb out.grb

Descriptions of NCEP grids are available online:

http://www.nco.ncep.noaa.gov/pmb/docs/on388/tableb.html

Run *copygb* – Option 2

- Read in grid navigation file created by *wrfpost* (NMM only, simple, restrictive)
  - Running *wrfpost* on WRF-NMM output produces two ASCII files containing grid navigation information which is similar in domain and grid spacing to the model integration domain.
    - *copygb_gridnav.txt* for a Lambert Conformal grid
    - *copygb_hwrf.txt* for a regular Lat-Lon grid

For example:

read nav < 'copygb_gridnav.txt'

copygb.exe –xg"${nav}" in.grb out.grb

Run *copygb* – Option 3

- Create a user-defined grid by specifying a full set of grid parameters. To interpolate onto a Lambert conformal grid:

indicates user-defined grid → 255; map type (3=LC) → 3; # of points → NX NY; SW corner (millidegrees) → STARTLAT STARTLON; central lon (millidegrees) → CENLON

copygb.exe –xg"255 3 NX NY STARTLAT STARTLON 8 CENLON
DX DY 0 64 TRUELAT1 TRUELAT2 " in.grb out.grb

horizontal spacing (meters) → DX DY; true latitudes (millidegrees) → TRUELAT1 TRUELAT2

copygb –xg"255 3 185 129 12190 -133459 8 -95000
40635 40635 0 64 25000 25000" in.grb out.grb

Run *copygb* – Option 3

- Create a user-defined grid by specifying a full set of grid parameters. To interpolate onto a Polar stereographic grid:

indicates user-defined grid → 255; map type (5=STR) → 5; # of points along lon, lat → NX NY; SW corner (millidegrees) → STARTLAT STARTLON; central lon (millidegrees) → CENLON

copygb.exe –xg"255 5 NX NY STARTLAT STARTLON 8 CENLON
DX DY 0 64" in.grb out.grb

horizontal **resolution (meters)** → DX DY

copygb –xg"255 5 580 548 10000 -128000 8 -105000
15000 15000 0 64" in.grb out.grb

Run *copygb* – Option 3

- Create a user-defined grid by specifying a full set of grid parameters. To interpolate onto a regular Latlon grid:

indicates user-defined grid | map type (0=LTLN) | # of points along lon, lat | SW corner (millidegrees) | NE lat (millidegrees)

copygb.exe –xg"255 0 NX NY STARTLAT STARTLON 8 ENDLAT ENDLON DLAT DLON 64" in.grb out.grb

NE lon (millidegrees) | grid spacing (millidegrees)

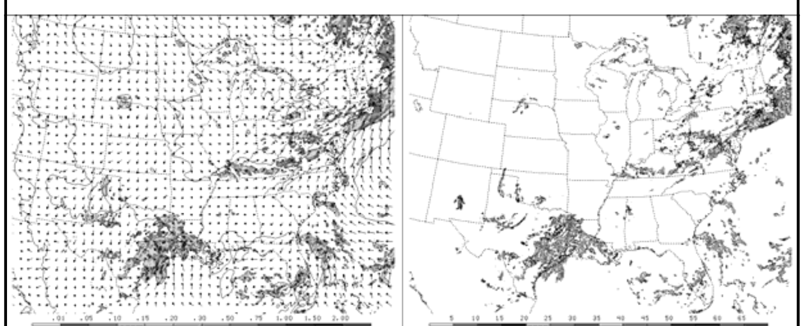
copygb –xg"255 0 401 401 10000 -130000 8 50000 -90000 100 100 64" in.grb out.grb

GRIB file visualization with GEMPAK

- The GEMPAK utility "nagrib" reads GRIB files from any non-staggered grid and generates GEMPAK-binary files that are readable by GEMPAK plotting programs
- GEMPAK can plot horizontal maps, vertical cross-sections, meteograms, and sounding profiles.
- Package download and user guide are available online: http://my.unidata.ucar.edu/content/software/gempak/index.html
- A sample script named *run_wrfpostandgempak* is included in scripts/ that can be used to run *wrfpost*, *copygb*, and then plot various fields using GEMPAK.
- Further details on this script and using GEMPAK are available in the user's guide.

Forecast plotted with GEMPAK :

Precipitation and derived Radar reflectivity

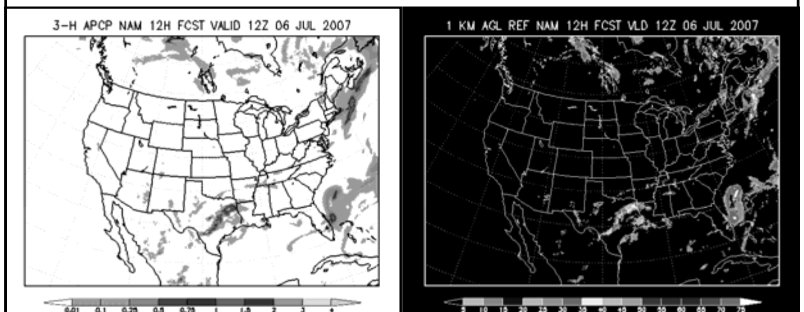


GRIB file visualization with GrADS

- GrADS also has utilities to read GRIB files on any non-staggered grids and generate GrADS "control" files. The utilities grib2ctl and gribmap are available via: http://www.cpc.ncep.noaa.gov/products/wesley/grib2ctl.html
- Package download and user guide for GrADS are available online: http://grads.iges.org/grads/gadoc/
- A sample script named *run_wrfpostandgrads* is included in scripts/ that can be used to the run *wrfpost*, *copygb*, and then plot various fields using GrADS.

Forecast plotted with GrADS:

Precipitation and derived Radar reflectivity

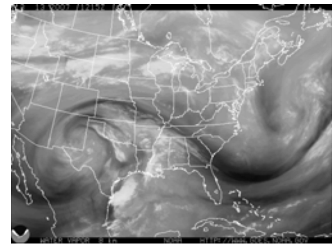


Future plan

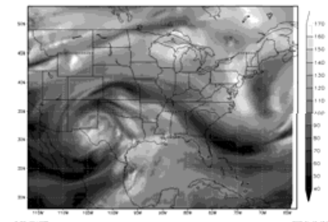
- The planned upgrade for WPP:

  1) add new products including simulated brightness temperature for GOES IR and water vapor channels;

  2) Include options to process and output on global grids.



observed water vapor ch



simulated WRF water vapor ch

WRF Registry & Examples

WRF Registry and Examples

John Michalakes, NCAR

Michael Duda, NCAR

Dave Gill, NCAR

WRF Software Architecture Working Group

Outline

- **Registry Mechanics**

- - - - - - - - - - - -

- **Examples**

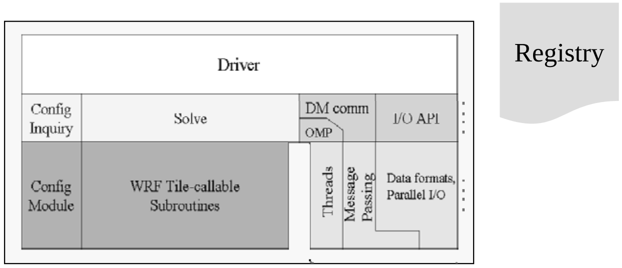
Introduction – Intended Audience

- **Intended audience for this tutorial session: scientific users and others who wish to:**
  - **Understand overall design concepts and motivations**
  - **Work with the code**
  - **Extend/modify the code to enable their work/research**
  - **Address problems as they arise**
  - **Adapt the code to take advantage of local computing resources**

Introduction – WRF Resources

- **WRF project home page**
  - **http://www.wrf-model.org**
- **WRF users page (linked from above)**
  - **http://www.mmm.ucar.edu/wrf/users**
- **On line documentation (also from above)**
  - **http://www.mmm.ucar.edu/wrf/WG2/software_v2**
- **WRF user services and help desk**
  - **wrfhelp@ucar.edu**

WRF Software Architecture



- **Hierarchical software architecture**
  - **Insulate scientists' code from parallelism and other architecture/implementation-specific details**
  - **Well-defined interfaces between layers, and external packages for communications, I/O, and model coupling facilitates code reuse and exploiting of community infrastructure, e.g. ESMF.**

WRF Registry

- **"Active data-dictionary" for managing WRF data structures**
  - **Database describing attributes of model state, intermediate, and configuration data**
    - **Dimensionality, number of time levels, staggering**
    - **Association with physics**
    - **I/O classification (history, initial, restart, boundary)**
    - **Communication points and patterns**
    - **Configuration lists (e.g. namelists)**
    - **Nesting up- and down-scale interpolation**

WRF Registry

- **"Active data-dictionary" for managing WRF data structures**
  - **Program for auto-generating sections of WRF from database:**
    - **<u>2000 - 3000</u> Registry entries ⇒ <u>160-thousand</u> lines of automatically generated WRF code**
    - **Allocation statements for state data and I1 data**
    - **Interprocessor communications: Halo and periodic boundary updates, transposes**
    - **Code for defining and managing run-time configuration information**
    - **Code for forcing, feedback, shifting, and interpolation of nest data**

WRF Registry

- **Why?**
  - **Automates time consuming, repetitive, error-prone programming**
  - **Insulates programmers and code from package dependencies**
  - **Allow rapid development**
  - **Documents the data**
- **A Registry file is available for each of the dynamical cores, plus special purpose packages**
- **Reference: Description of WRF Registry, <u>http://www.mmm.ucar.edu/wrf/WG2/software_v2</u>**

Registry Data Base

- **Currently implemented as a text file: Registry/Registry.EM**
- **Types of entry:**
  - ***Dimspec* – Describes dimensions that are used to define arrays in the model**
  - ***State* – Describes state variables and arrays in the domain structure**
  - ***I1* – Describes local variables and arrays in solve**
  - ***Typedef* – Describes derived types that are subtypes of the domain structure**

Registry Data Base

- **Types of entry:**
  - ***Rconfig* – Describes a configuration (e.g. namelist) variable or array**
  - ***Package* – Describes attributes of a package (e.g. physics)**
  - ***Halo* – Describes halo update interprocessor communications**
  - ***Period* – Describes communications for periodic boundary updates**
  - ***Xpose* – Describes communications for parallel matrix transposes**
  - ***include* – Similar to a CPP #include file**

Registry State Entry

```
#      Type Sym  Dims   Use    Tlev Stag IO         Dname      Descrip
state  real  u   ikjb   dyn_em   2   X    i01rhusdf   "U"    "X WIND COMPONENT“
```

- **Elements**
  - ***Entry*: The keyword “state”**
  - ***Type*: The type of the state variable or array (real, double, integer, logical, character, or derived)**
  - ***Sym*: The symbolic name of the variable or array**
  - ***Dims*: A string denoting the dimensionality of the array or a hyphen (-)**
  - ***Use*: A string denoting association with a solver or 4D scalar array, or a hyphen**
  - ***NumTLev*: An integer indicating the number of time levels (for arrays) or hypen (for variables)**

Registry State Entry

```
#      Type Sym  Dims   Use    Tlev Stag IO         Dname      Descrip
state  real  u   ikjb   dyn_em   2   X    i01rhusdf   "U"    "X WIND COMPONENT“
```

- **Elements**
  - ***Stagger*: String indicating staggered dimensions of variable (X, Y, Z, or hyphen)**
  - ***IO*: String indicating whether and how the variable is subject to I/O and Nesting**
  - ***DName*: Metadata name for the variable**
  - ***Units*: Metadata units of the variable**
  - ***Descrip*: Metadata description of the variable**

Registry State Entry

```
#      Type Sym  Dims    Use   Tlev Stag     IO      Dname      Descrip
state  real  u   ikjb   dyn_em   2   X    i01rhusdf   "U"    "X WIND COMPONENT“
```

- **This single entry results in over 100 lines of code automatically added to more than 40 different locations in the WRF model, the real and ideal initialization programs, and in the WRF-Var package**
- **Nesting code to interpolate, force, feedback, and smooth u**
- **Addition of u to the input, restart, history, and LBC I/O streams**

Registry State Entry

```
#      Type Sym  Dims    Use   Tlev Stag     IO      Dname      Descrip
state  real  u   ikjb   dyn_em   2   X    i01rhusdf   "U"    "X WIND COMPONENT“
```

Declaration and dynamic allocation of arrays in TYPE(domain)

Two 3D state arrays corresponding to the 2 time levels of U
u_1 (ims:ime , kms:kme , jms:jme)
u_2 (ims:ime , kms:kme , jms:jme)

Eight LBC arrays for boundary and boundary tendencies (dimension example for x BC)
u_b[xy][se] (jms:jme, kms:kme, spec_bdy_width, 4)
u_bt[xy][se] (jms:jme, kms:kme, spec_bdy_width, 4)

State Entry: Defining a variable-set for an I/O stream

- **Fields are added to a variable-set on an I/O stream in the Registry**

```
#      Type Sym  Dims    Use   Tlev  Stag    IO      Dname      Descrip
state  real  u   ikjb   dyn_em  2     X   i01rhusdf   "U"   "X WIND COMPONENT“
```

IO is a string that specifies if the variable is to be subject to initial, restart, history, or boundary I/O. The string may consist of 'h' (subject to history I/O), 'i' (initial dataset), 'r' (restart dataset), or 'b' (lateral boundary dataset). The 'h', 'r', and 'i' specifiers may appear in any order or combination.

The ‘h’ and ‘i’ specifiers may be followed by an optional integer string consisting of ‘0’, ‘1’, … , ‘9’ Zero denotes that the variable is part of the principal input or history I/O stream. The characters ‘1’ through ‘9’ denote one of the auxiliary input or history I/O streams.

usdf refers to nesting options: **u = UP, d = DOWN, s = SMOOTH, f = FORCE**

State Entry: Defining Variable-set for an I/O stream

`irh` -- The state variable will be included in the WRF model input, restart, and history I/O streams

`irh13` -- The state variable has been added to the first and third auxiliary history output streams; it has been removed from the principal history output stream, because zero is not among the integers in the integer string that follows the character 'h'

`rh01` -- The state variable has been added to the first auxiliary history output stream; it is also retained in the principal history output

`i205hr` -- Now the state variable is included in the principal input stream as well as auxiliary inputs 2 and 5. Note that the order of the integers is unimportant. The variable is also in the principal history output stream

`ir12h` -- No effect; there is only 1 restart data stream

`i01` -- Data goes into real and into WRF

`i1` -- Data goes into real only

Rconfig Entry

```
#         Type    Sym            How set            Nentries  Default
rconfig   integer spec_bdy_width namelist,bdy_control    1           1
```

- **This defines namelist entries**
- **Elements**
  - ***Entry*: the keyword "rconfig"**
  - ***Type*: the type of the namelist variable (integer, real, logical, string)**
  - ***Sym*: the name of the namelist variable or array**
  - ***How set*: indicates how the variable is set: e.g. namelist or derived, and if namelist, which block of the namelist it is set in**

Rconfig Entry

```
#         Type    Sym            How set            Nentries  Default
rconfig   integer spec_bdy_width namelist,bdy_control    1           1
```

- **This defines namelist entries**
- **Elements**
  - ***Nentries*: specifies the dimensionality of the namelist variable or array. If 1 (one) it is a variable and applies to all domains; otherwise specify max_domains (which is an integer parameter defined in module_driver_constants.F).**
  - ***Default*: the default value of the variable to be used if none is specified in the namelist; hyphen (-) for no default**

Rconfig Entry

```
#         Type    Sym            How set            Nentries  Default
rconfig   integer spec_bdy_width namelist,bdy_control    1           1
```

- **Result of this Registry Entry:**
  - **Define an namelist variable "spec_bdy_width" in the bdy_control section of namelist.input**
  - **Type integer (others: real, logical, character)**
  - **If this is first entry in that section, define "bdy_control" as a new section in the namelist.input file**
  - **Specifies that bdy_control applies to all domains in the run**

```
--- File: namelist.input ---

&bdy_control
 spec_bdy_width       = 5,
 spec_zone            = 1,
 relax_zone           = 4,
    . . .
 /
```

Rconfig Entry

```
#         Type    Sym            How set            Nentries  Default
rconfig   integer spec_bdy_width namelist,bdy_control    1           1
```

- **Result of this Registry Entry:**
  - **if Nentries is "max_domains" then the entry in the namelist.input file is a comma-separate list, each element of which applies to a separate domain**
  - **The single entry in the Registry file applies to each of the separate domains**

```
--- File: namelist.input ---

&bdy_control
 spec_bdy_width       = 5,
 spec_zone            = 1,
 relax_zone           = 4,
    . . .
 /
```

Rconfig Entry

```
#         Type      Sym              How set            Nentries  Default
rconfig   integer spec_bdy_width  namelist,bdy_control     1          1
```

- **Result of this Registry Entry:**
  - **Specify a default value of "1" if nothing is specified in the namelist.input file**
  - **In the case of a multi-process run, generate code to read in the bdy_control section of the namelist.input file on one process and broadcast the value to all other processes**

```
 --- File: namelist.input ---

&bdy_control
 spec_bdy_width        = 5,
 spec_zone             = 1,
 relax_zone            = 4,
    . . .
 /
```

Package Entry

- **Elements**
  - ***Entry*: the keyword "package",**
  - ***Package name*: the name of the package: e.g. "kesslerscheme"**
  - ***Associated rconfig choice*: the name of a rconfig variable and the value of that variable that choses this package**

```
# specification of microphysics options
package   passiveqv     mp_physics==0    -       moist:qv
package   kesslerscheme mp_physics==1    -       moist:qv,qc,qr
package   linscheme     mp_physics==2    -       moist:qv,qc,qr,qi,qs,qg
package   ncepcloud3    mp_physics==3    -       moist:qv,qc,qr
package   ncepcloud5    mp_physics==4    -       moist:qv,qc,qr,qi,qs

# namelist entry that controls microphysics option
rconfig   integer     mp_physics   namelist,physics     max_domains    0
```

Package Entry

- **Elements**
  - ***Package state vars*: unused at present; specify hyphen (-)**
  - ***Associated* variables: the names of 4D scalar arrays (moist, chem, scalar) and the fields within those arrays this package uses, and the state variables (state:u_gc, ...)**

```
# specification of microphysics options
package   passiveqv     mp_physics==0    -       moist:qv
package   kesslerscheme mp_physics==1    -       moist:qv,qc,qr
package   linscheme     mp_physics==2    -       moist:qv,qc,qr,qi,qs,qg
package   ncepcloud3    mp_physics==3    -       moist:qv,qc,qr
package   ncepcloud5    mp_physics==4    -       moist:qv,qc,qr,qi,qs

# namelist entry that controls microphysics option
rconfig   integer     mp_physics   namelist,physics     max_domains    0
```

Package Entry

```
USE module_state_descriptions

...

Micro_select : SELECT CASE ( mp_physics )

   CASE ( KESSLERSCHEME )
      CALL kessler ( ...

   CASE ( THOMPSON )
      CALL mp_gt_driver ( ...

   ...

END SELECT micro_select
```

Packages define automatically enumerated types to avoid the usual tests on option #17 for microphysics

Halo Entry

- **Elements**
  - ***Entry*: the keyword “halo”,**
  - ***Communication name*: given to the particular communication, must be identical in the source code (case matters!)**
  - ***Associated dynamical core*: dyn_em XOR dyn_nmm are acceptable**
  - ***Stencil size:* 4, or (2n+1)^2-1 (i.e. 8, 24, 48; semi-colon separated)**
  - ***Which variables*: names of the variables (comma separated)**

```
# Halo update communications
halo      HALO_EM_TKE_C dyn_em 4:ph_2,phb
```

HALO Entry

Place communication in dyn_em/solve_em.F

```
#ifdef DM_PARALLEL
#     include "HALO_EM_TKE_C.inc"
#endif
```


```
# Halo update communications
halo      HALO_EM_TKE_C dyn_em 4:ph_2,phb
```


PERIOD and XPOSE Entry

- **Elements**
  - ***Entry*: the keyword “period” or “xpose” (transpose)**
  - ***Communication name*: given to the particular communication, must be identical in the source code (case matters!)**
  - ***Associated dynamical core*: dyn_em XOR dyn_nmm are acceptable**
  - ***Stencil size for period:* # rows and columns to share for periodic lateral BCs**
  - ***Which variables for period*: names of the variables (comma separated)**
  - ***Which variables for xpose:* original variable (3d), x-transposed and y-transposed fields**

```
# Period update communications
period  PERIOD_EM_COUPLE_A dyn_em 2:mub,mu_1,mu_2
```

```
# Transpose update communications
xpose XPOSE_POLAR_FILTER_TOPO dyn_em t_init,t_xxx,dum_yyy
```

Registry IO: registry.io_boilerplate

- **include – method to populate Registry without duplicating information which is prone to administrative mismanagement**
  - ***Entry*: the keyword “include”**
  - ***Name*: file name to include in the Registry file**

```
Entry             Name
include registry.io_boilerplate
```

Registry IO: registry.io_boilerplate

- **rconfig - namelist entries**
  - ***Entry*: the keyword "rconfig",**
  - ***Type*: integer, logical, real**
  - ***WRF symbol*: name of variable in namelist**
  - ***Namelist record:* name of the resident record**
  - ***Number of entries*: either "1" or "max_domains"**
  - ***Default value*: what to define if not in namelist.input file**
  - ***NOT REQUIRED name and description*: for self documentation purposes**

```
Entry      Type        Sym              How set
rconfig   character  auxinput5_inname  namelist,time_control

Num Entries              Default
    1           "auxinput5_d<domain>_<date>"
```

<domain> expanded to 2-digit domain identifier
<date> expanded to the usual WRF "years down to seconds" date string

Registry IO: registry.io_boilerplate

```
Entry      Type        Sym                 How set

rconfig character auxinput5_outname      namelist,time_control
rconfig character auxinput5_inname       namelist,time_control
rconfig integer   auxinput5_interval_mo  namelist,time_control
rconfig integer   auxinput5_interval_d   namelist,time_control
rconfig integer   auxinput5_interval_h   namelist,time_control
rconfig integer   auxinput5_interval_m   namelist,time_control
rconfig integer   auxinput5_interval_s   namelist,time_control
rconfig integer   auxinput5_interval     namelist,time_control   <----
rconfig integer   auxinput5_begin_y      namelist,time_control
rconfig integer   auxinput5_begin_mo     namelist,time_control
rconfig integer   auxinput5_begin_d      namelist,time_control
rconfig integer   auxinput5_begin_h      namelist,time_control
rconfig integer   auxinput5_begin_m      namelist,time_control
rconfig integer   auxinput5_begin_s      namelist,time_control
rconfig integer   auxinput5_end_y        namelist,time_control
rconfig integer   auxinput5_end_mo       namelist,time_control
rconfig integer   auxinput5_end_d        namelist,time_control
rconfig integer   auxinput5_end_h        namelist,time_control
rconfig integer   auxinput5_end_m        namelist,time_control
rconfig integer   auxinput5_end_s        namelist,time_control
rconfig integer   io_form_auxinput5      namelist,time_control
```

Registry IO: registry.io_boilerplate

```
Entry      Type        Sym                 How set

rconfig   integer     io_form_input        namelist,time_control
rconfig   integer     io_form_history      namelist,time_control
rconfig   integer     io_form_restart      namelist,time_control
rconfig   integer     io_form_boundary     namelist,time_control

rconfig   integer     io_form_auxinput1    namelist,time_control
rconfig   integer     io_form_auxinput2    namelist,time_control
rconfig   integer     io_form_auxinput3    namelist,time_control
rconfig   integer     io_form_auxinput4    namelist,time_control
rconfig   integer     io_form_auxinput5    namelist,time_control
rconfig   integer     io_form_auxinput6    namelist,time_control
rconfig   integer     io_form_auxinput7    namelist,time_control
rconfig   integer     io_form_auxinput8    namelist,time_control
rconfig   integer     io_form_auxinput9    namelist,time_control

rconfig   integer     io_form_gfdda        namelist,fdda
rconfig   integer     io_form_auxinput11   namelist,time_control
```

For any given WRF model fcst, users have access to these input streams

Registry IO: registry.io_boilerplate

```
Entry      Type        Sym                 How set

rconfig   integer     io_form_auxhist1     namelist,time_control
rconfig   integer     io_form_auxhist2     namelist,time_control
rconfig   integer     io_form_auxhist3     namelist,time_control
rconfig   integer     io_form_auxhist4     namelist,time_control
rconfig   integer     io_form_auxhist5     namelist,time_control
rconfig   integer     io_form_auxhist6     namelist,time_control
rconfig   integer     io_form_auxhist7     namelist,time_control
rconfig   integer     io_form_auxhist8     namelist,time_control
rconfig   integer     io_form_auxhist9     namelist,time_control
rconfig   integer     io_form_auxhist10    namelist,time_control
rconfig   integer     io_form_auxhist11    namelist,time_control
```

… and access to these output streams

Registry Data Base - Review

- **Currently implemented as a text file: Registry/Registry.EM**
- **Types of entry:**
  - ***Dimspec* – Describes dimensions that are used to define arrays in the model**
  - ***State* – Describes state variables and arrays in the domain structure**
  - ***I1* – Describes local variables and arrays in solve**
  - ***Typedef* – Describes derived types that are subtypes of the domain structure**

Registry Data Base - Review

- **Types of entry:**
  - ***Rconfig* – Describes a configuration (e.g. namelist) variable or array**
  - ***Package* – Describes attributes of a package (e.g. physics)**
  - ***Halo* – Describes halo update interprocessor communications**
  - ***Period* – Describes communications for periodic boundary updates**
  - ***Xpose* – Describes communications for parallel matrix transposes**
  - ***include* – Similar to a CPP #include file**

Outline

- **Registry Mechanics**

- - - - - - - - - - - -

- **Examples**
  - **1) Add a variable to the namelist**
  - **2) Add an array**
  - **3) Compute a diagnostic**
  - **4) Add a physics package**

Example 1: Add a variable to the namelist

- **Use the examples for the rconfig section of the Registry**
- **Find a namelist variable similar to what you want**
  - **Integer *vs* real *vs* logical *vs* character**
  - **Single value *vs* value per domain**
  - **Select appropriate namelist record**
- **Insert your mods in all appropriate Registry files**
- **Remember that ALL Registry changes require that the WRF code be cleaned and rebuilt**

Example 1: Add a variable to the namelist

- **Adding a variable to the namelist requires the inclusion of a new line in the Registry file:**

```
rconfig integer my_option  namelist,time_control  1 0 - "my_option"
"test namelist option"
```

- **Accessing the variable is through an automatically generated function:**

```
USE module_configure
INTEGER :: my_option

CALL nl_get_my_option( 1, my_option )
CALL nl_set_my_option( 1, my_option )
```

Example 1: Add a variable to the namelist

- **You also have access to the namelist variables from the grid structure …**

```
SUBROUTINE foo ( grid , ... )

  USE module_domain
  TYPE(domain) :: grid

  print *,grid%dx
```

Example 1: Add a variable to the namelist

- **… and you also have access to the namelist variables from config_flags**

```
SUBROUTINE foo2 ( config_flags , ... )

  USE module_configure
  TYPE(grid_config_rec_type) :: config_flags

  print *,config_flags%dx
```

Examples

- **1) Add a variable to the namelist**
- **2) Add an array to solver, and IO stream**
- **3) Compute a diagnostic**
- **4) Add a physics package**

Example 2: Add an Array

- **Adding a state array to the solver, requires adding a single line in the Registry**
- **Use the previous Registry instructions for a state or I1 variable**
- **Select a variable similar to one that you would like to add**
  - **2d *vs* 3d**
  - **Staggered**
  - **Associated with a package**
  - **Part of a 4d array**
  - **Input (012), output, restart**
  - **Nesting, lateral forcing, feedback**

Example 2: Add an Array

- **Copy the "similar" field's line and make a few edits**
- **Remember, no Registry change takes effect until a clean and rebuild**

```
state  real  h_diabatic  ikj  misc  1  -      r                    \
       "h_diabatic"  "PREVIOUS TIMESTEP CONDENSATIONAL HEATING"

state  real  msft         ij   misc  1  -     i012rhdu=(copy_fcnm)  \
       "MAPFAC_M"    "Map scale factor on mass grid"

state  real  ht           ij   misc  1  -     i012rhdus             \
       "HGT"         "Terrain Height"

state  real  ht_input     ij   misc  1  -     -                     \
       "HGT_INPUT"   "Terrain Height from FG Input File"

state  real  TSK_SAVE     ij   misc  1  -     -                     \
       "TSK_SAVE"    "SURFACE SKIN TEMPERATURE"  "K"
```

Examples

- **1) Add a variable to the namelist**
- **2) Add an array**
- **3) Compute a diagnostic**
- **4) Add a physics package**

Example 3: Compute a Diagnostic

- **Problem: Output global average and global maximum and lat/lon location of maximum for 10 meter wind speed in WRF**
- **Steps:**
  - **Modify solve to compute wind-speed and then compute the local sum and maxima at the end of each time step**
  - **Use reduction operations built-in to WRF software to compute the global qualities**
  - **Output these on one process (process zero, the "monitor" process)**

Example 3: Compute a Diagnostic

- **Compute local sum and local max and the local indices of the local maximum**

```
   --- File: dyn_em/solve_em.F  (near the end) ---

! Compute local maximum and sum of 10m wind-speed
   sum_ws = 0.
   max_ws = 0.
   DO j = jps, jpe
     DO i = ips, ipe
       wind_vel = sqrt( grid%u10(i,j)**2+ grid%v10(i,j)**2 )
       IF ( wind_vel .GT. max_ws ) THEN
          max_ws = wind_vel
          idex = i
          jdex = j
       ENDIF
       sum_ws = sum_ws + wind_vel
     ENDDO
   ENDDO
```

Example 3: Compute a Diagnostic

- **Compute global sum, global max, and indices of the global max (WRF intrinsics)**

```
! Compute global sum
   sum_ws = wrf_dm_sum_real ( sum_ws )

! Compute global maximum and associated i,j point
   CALL wrf_dm_maxval_real ( max_ws, idex, jdex )
```

Example 3: Compute a Diagnostic

- **On the process that contains the maximum value, obtain the latitude and longitude of that point; on other processes set to an artificially low value.**
- The use parallel reduction to store that result on every process

```
   IF ( ips .LE. idex .AND. idex .LE. ipe .AND.  &
        jps .LE. jdex .AND. jdex .LE. jpe ) THEN
      glat = grid%xlat(idex,jdex)
      glon = grid%xlong(idex,jdex)
   ELSE
      glat = -99999.
      glon = -99999.
   ENDIF

! Compute global maximum to find glat and glon
   glat = wrf_dm_max_real ( glat )
   glon = wrf_dm_max_real ( glon )
```

Example 3: Compute a Diagnostic

- **Output the value on process zero, the “monitor”**

```
! Print out the result on the monitor process
   IF ( wrf_dm_on_monitor() ) THEN
      WRITE(outstring,*)'Avg. ',sum_ws/((ide-ids+1)*(jde-jds+1))
      CALL wrf_message ( TRIM(outstring) )
      WRITE(outstring,*)'Max. ',max_ws,' Lat. ',glat,&
                                       ' Lon. ',glon
      CALL wrf_message ( TRIM(outstring) )
   ENDIF
```

Example 3: Compute a Diagnostic

- **Output from process zero of an *n* process run**

```
   --- Output file: rsl.out.0000 ---
   . . .
  Avg.    5.159380
  Max.    15.09370     Lat.    37.25022     Lon.   -67.44571
Timing for main: time 2000-01-24_12:03:00 on domain   1:    8.96500 elapsed secs.
  Avg.    5.166167
  Max.    14.97418     Lat.    37.25022     Lon.   -67.44571
Timing for main: time 2000-01-24_12:06:00 on domain   1:    4.89460 elapsed secs.
  Avg.    5.205693
  Max.    14.92687     Lat.    37.25022     Lon.   -67.44571
Timing for main: time 2000-01-24_12:09:00 on domain   1:    4.83500 elapsed secs.
   . . .
```

Examples

- **1) Add a variable to the namelist**
- **2) Add an array**
- **3) Compute a diagnostic**
- **4) Add a physics package**

Example 4: Input periodic SSTs

- **Add a new physics package with time varying input source to the model**
- **This is how we could supply a time varying value to the model for a field that is traditionally fixed**
- **Example is sea surface temperature**

Example 4: Input periodic SSTs

- **Problem: adapt WRF to input a time-varying lower boundary condition, e.g. SSTs, from an input file for a new surface scheme**
- **Given: Input file in WRF I/O format containing 12-hourly SST's**
- **Modify WRF model to read these into a new state array and make available to WRF surface physics**

Example 4: Input periodic SSTs

- **Steps**
  - **Add a new state variable and definition of a new surface layer package (that will use the variable) to the Registry**
  - **Add to variable stream for an unused Auxiliary Input stream**
  - **Adapt physics interface to pass new state variable to physics**
  - **Setup namelist to input the file at desired interval**

Example 4: Input periodic SSTs

- **Add a new state variable to Registry/Registry.EM or Registry/Registry.NMM and put it in the variable set for input on AuxInput #4**

```
#     type  symbol dims use  tl stag  io    dname     description       units
state real  nsst   ij   misc  1  -    i4rh  "NEW_SST" "Time Varying SST" "K“
```

- **Also added to History and Restart**
- **Result:**
  - 2-D variable named **nsst** defined and available in solve_em
  - Dimensions: ims:ime, jms:jme
  - Input and output on the AuxInput #4 stream will include the variable under the name NEW_SST

Example 4: Input periodic SSTs

- **Pass new state variable to surface physics**

```
      --- File: dyn_em/solve_em.F ---

  CALL surface_driver(                                                   &
           . . .
! Optional
 &        ,QV_CURR=moist(ims,kms,jms,P_QV), F_QV=F_QV                     &
 &        ,QC_CURR=moist(ims,kms,jms,P_QC), F_QC=F_QC                     &
 &        ,QR_CURR=moist(ims,kms,jms,P_QR), F_QR=F_QR                     &
 &        ,QI_CURR=moist(ims,kms,jms,P_QI), F_QI=F_QI                     &
 &        ,QS_CURR=moist(ims,kms,jms,P_QS), F_QS=F_QS                     &
 &        ,QG_CURR=moist(ims,kms,jms,P_QG), F_QG=F_QG                     &
 &        ,NSST=grid%nsst                                                 & ! new
 &        ,CAPG=grid%capg, EMISS=grid%emiss, HOL=hol,MOL=grid%mol         &
 &        ,RAINBL=grid%rainbl,SR=grid%em_sr                               &
 &        ,RAINNCV=grid%rainncv,REGIME=regime,T2=grid%t2,THC=grid%thc     &
            . . .
```

Example 4: Input periodic SSTs

- **Add new variable nsst to Physics Driver in Mediation Layer**

```
     --- File: phys/module_surface_driver.F ---

SUBROUTINE surface_driver(                                                &
       . . .
          !  Other optionals (more or less em specific)
  &          ,nsst                                                        &
  &          ,capg,emiss,hol,mol                                          &
  &          ,rainncv,rainbl,regime,t2,thc                                &
  &          ,qsg,qvg,qcg,soilt1,tsnav                                    &
  &          ,smfr3d,keepfr3dflag                                         &
      . . .
                                                                          ))
      . . .
REAL, DIMENSION( ims:ime, jms:jme ), OPTIONAL, INTENT(INOUT)::   nsst
```

- **By making this an “Optional” argument, we preserve the driver’s compatibility with other cores and with versions of WRF where this variable hasn’t been added.**

Example 4: Input periodic SSTs

- **Add call to Model-Layer subroutine for new physics package to Surface Driver**

```
      --- File: phys/module_surface_driver ---

!$OMP PARALLEL DO   &
!$OMP PRIVATE ( ij, i, j, k )
   DO ij = 1 , num_tiles
     sfclay_select: SELECT CASE(sf_sfclay_physics)

       CASE (SFCLAYSCHEME)
          . . .
       CASE (NEWSFCSCHEME)  ! <- This is defined by the Registry “package” entry

         IF (PRESENT(nsst))  THEN
            CALL NEWSFCCHEME(                                    &
                nsst,                                            &
                ids,ide, jds,jde, kds,kde,                       &
                ims,ime, jms,jme, kms,kme,                       &
                i_start(ij),i_end(ij), j_start(ij),j_end(ij), kts,kte    )
         ELSE
           CALL wrf_error_fatal('Missing argument for NEWSCHEME in surface driver')
         ENDIF
          . . .
     END SELECT sfclay_select
   ENDDO
!$OMP END PARALLEL DO
```

- **Note the PRESENT test to make sure new optional variable nsst is available**

Example 4: Input periodic SSTs

- **Add definition for new physics package NEWSCHEME as setting 4 for namelist variable sf_sfclay_physics**

```
rconfig   integer  sf_sfclay_physics   namelist,physics   max_domains    0

package   sfclayscheme   sf_sfclay_physics==1       -            -
package   myjsfcscheme   sf_sfclay_physics==2       -            -
package   gfssfcscheme   sf_sfclay_physics==3       -            -
package   newsfcscheme   sf_sfclay_physics==4       -            -
```

- **This creates a defined constant NEWSFCSCHEME and represents selection of the new scheme when the namelist variable sf_sfclay_physics is set to ‘4’ in the namelist.input file**
- `clean –a` **and recompile so code and Registry changes take effect**

Example 4: Input periodic SSTs

- **Setup namelist to input SSTs from the file at desired interval**

```
     --- File: namelist.input ---

&time_control
   . . .
 auxinput4_inname      = "sst_input"
 auxinput4_interval_h  = 12
 . . .
/

   . . .
&physics
 sf_sfclay_physics  = 4, 4, 4
   . . .
/
```

- **Run code with sst_input file in run-directory**

Example 4: Input periodic SSTs

- A few notes…
  - The read times and the time-stamps in the input file must match exactly
  - We haven’t done anything about what happens if the file runs out of time periods (the last time period read will be used over and over again, though you’ll see some error messages in the output if you set debug_level to be 1 or greater in namelist.input)
  - We haven’t said anything about what generates sst_input

Nesting Setup

WRF Nesting: Set Up and Run

Wei Wang
NCAR/ESSL/MMM

Outline

- General comments
- Nest namelist options
- Running WRF with nests
  - NMM case: two-way nesting
  - ARW case: two-way nesting
  - ARW moving nest
  - ARW one-way nesting
- Summary

Before You Run ..

- Make sure you have selected nest compile options and appropriate executables are created in **`WRFV3/main/`** directory:

  For ARW:
  - **`ideal.exe`**
  - **`real.exe`**
  - **`wrf.exe`**
  - **`ndown.exe`**

  For NMM:
  - **`real_nmm.exe`**
  - **`wrf.exe`**

- If you are running a real-data case, be sure that files for *nest* domains from WPS are generated:
  - **`met_em.d01.`*<date>*`, met_em.d0*.`*<date>*** for ARW or
  - **`met_nmm.d01.`*<date>*`, geo_nmm_nest.l0*.nc`** for NMM

Steps to Run (same as before)

1. cd to *run/* or one of the *test case* directories
2. Link or copy WPS output files to the directory for real-data cases
3. Edit *namelist.input* file for the appropriate grid and times of the case
4. Run initialization program (*real.exe*, or *real_nmm.exe*) as in the single domain case
5. Run model executable, *wrf.exe*

All in the namelist…

- Nearly all controls for a nested run can be achieved by editing the namelist file.
- Look at nest specific namelist options

Important to note:

- Key variable: **`max_dom`** must be set to >= 2
- Need to pay attention to multi-column namelists

Nest namelist Options

&time_control

```
run_days         = 0,
run_hours        = 24,
run_minutes      = 0,
run_seconds      = 0,
start_year       = 2000, 2000, 2000,
start_month      = 01,   01,   01,
start_day        = 24,   24,   24,
start_hour       = 12,   12,   12,
start_minute     = 00,   00,   00,
start_second     = 00,   00,   00,
end_year         = 2000, 2000, 2000,
end_month        = 01,   01,   01,
end_day          = 25,   25,   25,
end_hour         = 12,   12,   12,
end_minute       = 00,   00,   00,
end_second       = 00,   00,   00,
interval_seconds = 21600
```

First column: domain 1 option

These controls the start and end times of the nests. They can be different from the parent domain, but must fit in the time window of the parent domain

&time_control

```
interval_seconds    = 21600
history_interval    = 180,  60,   60,
frame_per_outfile   = 1000, 1000, 1000,
restart_interval    = 360,
```

History files may be split into multiple pieces

- History files are written separately for each domains
- History intervals may be different for different domains
- restart files are also written one per domain

&time_control

Nest input option: ARW only

```
input_from_file  = .true., .true., .true.,
fine_input_stream   = 0, 2, 2,
```

Specify what fields to use in nest input: they can be all (0), or data specified in I/O stream 2 in Regsitry (2). Useful for nest start at a later time.

Whether to produce in *real* and use nest wrfinput files in *wrf*. This is usually the case for real-data runs. For idealized nest runs, set it to .false. .

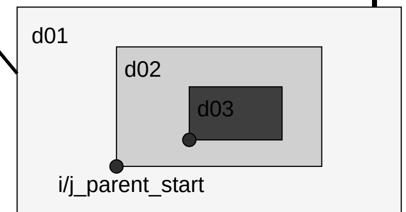
&domains

```
max_dom  = 3,
s_we     = 1,  1,  1,
e_we     = 74, 112,94,
s_sn     = 1,  1,  1,
e_sn     = 61, 97, 91,
s_vert   = 1,  1,  1,
e_vert   = 28, 28, 28,
grid_id        = 1,  2,  3,
parent_id      = 0,  1,  2,
i_parent_start = 0, 31, 30,
j_parent_start = 0, 17, 30,
```

Activate nests: no. of domains to run

Dimensions of all domains; same as in WPS.

Make sure the nest domain parameters match those defined in WPS



&domains

ARW

```
dx = 30000, 10000, 3333.33,
dy = 30000, 10000, 3333.33,
parent_grid_ratio = 1, 3, 3,
parent_time_step_ratio = 1,3,3,
```

All 4 variables must be specified. *Grid ratio* can be any integer, and *time step ratio* can be different from grid ratio. Grid distance is in meters, even for lat/lon map projection

NMM

```
dx = 0.096290,
dy = 0.096011,
parent_grid_ratio = 1,
parent_time_step_ratio = 1,
```

Values in nest columns are ignored. Everything is defined by 1:3 ratio in the model.

&domains

```
feedback      = 1,
smooth_option = 2,
```

When feedback is on, this option can be selected to smooth the area in the parent domain where nest is. Valid values are 0,1,2.

Whether nest will overwrite parent domain results. Setting *feedback*=0 ➔ 'one-way' nesting in a concurrent run.

&bdy_control

```
spec_bdy_width = 5, (1 for NMM)
spec_zone      = 1, (ARW only)
relax_zone     = 4, (ARW only)
specified      = .T.,.F.,.F.,
nested         = .F.,.T.,.T.,
```

Boundary condition option for domain 1.

Boundary condition option for nests.

May change *relax_zone* and *spec_bdy_width* for ARW

Other notes on namelists

- Use same physics options for all domains.
  - An exception is cumulus scheme. One may need to turn it off for a nest that has grid distance of a few kilometers.
- Also use same physics calling frequency (e.g. **`radt, cudt,`** etc.) in all domains.

Where do I start?

- Always start with a *namelist* template provided in a test case directory, whether it is a ideal case, ARW or NMM.
- Not all namelists are function of domains. If in doubt, check Registry.EM or Registry.NMM and registry.io_boilerplate (look for string 'namelist').
- Use document to guide the modification of the namelist values:
  - run/README.namelist
  - User's Guide, Chapter 5

Running NMM Nested Case

Running WRF NMM Nested Cases

- Files available from WPS:
  met_nmm.d01.<date>
  geo_nmm_nest.l0*.nc,.. (from **geogrid**)

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

```
cd test/nmm_real
ln -s ../../../WPS/met_nmm.*  .
ln -s ../../../WPS/geo_nmm.*  .
```

Mesoscale & Microscale Meteorological Division / NCAR 17

Running WRF NMM Nested Cases

- Edit **namelist.input** file for runtime options (set **max_dom >= 2** for a nest run)

- Run the real-data initialization program (MPI only):
  mpirun -np *N* ./real_nmm.exe

- Successfully running this program will create model initial and boundary files:
  wrfinput_d01
  wrfbdy_d01
  geo_nmm_nest.l01.nc ← from geogrid

Mesoscale & Microscale Meteorological Division / NCAR 18

Running WRF NMM Nested Cases

- Run the model executable by typing (MPI only):
  mpirun -np *N* ./wrf.exe

- Successfully running the model will create model *history* files, one for each domain:
  wrfout_d01_2005-08-28_00:00:00
  wrfout_d02_2005-08-28_00:00:00

  And *restart* file if selected:
  wrfrst_d01_<*date*>, wrfrst_d02_<date>

Mesoscale & Microscale Meteorological Division / NCAR 19

Running ARW Nested Case

Mesoscale & Microscale Meteorological Division / NCAR 20

Running WRF ARW Nested Cases

- Files available from WPS:

  met_em.d01.<*date*>

  met_em.d02.<*date*> (at least one time)

  …
- Link or copy WPS output files to the run directory:

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


Running WRF ARW Nested Cases

- Edit **namelist.input** file for runtime options (set **max_dom >= 2** in **&domains** for a nested run)
- Run the real-data initialization program:

  ./real.exe, if compiled serially / SMP, or

  mpirun -np *N* ./real.exe, for a MPI job

  where ***N*** is the number of processors requested

Running WRF ARW Nested Cases

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

wrfinput_d01 ← *Single time level data at model's start time for all domains*

wrfinput_d02

wrfbdy_d01 ← *Multiple time level data at the lateral boundary, and only for domain 1*

Running WRF ARW Nested Cases

- Run the model executable by typing:

  ./wrf.exe >& wrf.out &

  or

  mpirun -np *N* ./wrf.exe &
- Successfully running the model will create model *history* files, one for each domain:

  wrfout_d01_2005-08-28_00:00:00

  wrfout_d02_2005-08-28_00:00:00

  And *restart* file if selected:

  wrfrst_d01_<*date*>, wrfrst_d02_<*date*>

Moving Nest Case (ARW only)

- The main reason for using this option is to run the model economically.
- Must choose correct compile options when creating **`configure.wrf`** file
  - Choose preset move, or vortex following
- Other options are controlled by the namelists.
- Can do specified move, and automatic vortex tracking (for tropical cyclone application).
- All nest domains can move.

Specified Moving Case

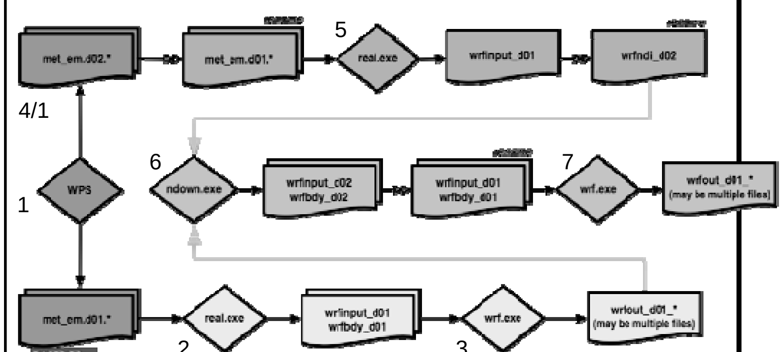
- Namelists in **`&domains`**:
  `num_moves, move_id, move_interval, move_cd_x, move_cd_y, corral_dist`
  ➔ only one-grid-cell move at a time
- Must specify initial nest location

Automatic Moving Case

- Tropical cyclone applications only.
- Works better for well developed storms.
- Namelists in **`&domains`**:
  `vortex_interval` (default 15 min)
  `max_vortex_speed` (default 40 m/s)
  `corral_dist` (default 8 coarse grid cells)
  `track_level` (default 50000 Pa)
- Must specify initial nest location

One-way Nesting: Two separate runs

ARW only



Summary

- NMM:
  - Two-way nesting, two inputs. (**`feedback=0`** ok)
- ARW:
  - Two-way, without nest input files (**`input_from_file=.f`**.)
  - Two-way, with nest input files (**`input_from_file = .t`**.)
  - Two-way, with static nest input only (**`input_from_file=.t. fine_input_stream = 2`**)
  - One-way, concurrent run (**`feedback = 0`**)
  - One-way, separate runs (treated like two single domain runs, with ***ndown***)
  - Two-way, specified moving nest run
  - Two-way, automatic vortex tracking run

References

- Information on compiling and running WRF with nests, and a more extensive list of namelist options and their definition / explanations can be found in the ARW and NMM User's Guide, Chapter 5
- Start with namelist templates in test/ directory
- Practice with online tutorial, and in the class.

ARW Nudging

WRF Four-Dimensional Data Assimilation (FDDA)

Jimy Dudhia

ARW only

FDDA

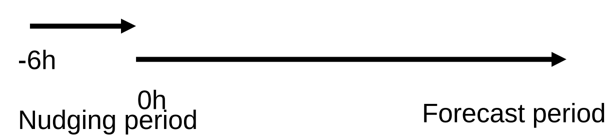
- Method of nudging model towards observations or analysis
- May be used for
  - Dynamical initialization (pre-forecast period)
  - Creating 4D meteorological datasets (e.g. for air quality model)
  - Boundary conditions (outer domain nudged towards analysis)

Method

- Model is run with extra nudging terms for horizontal winds, temperature and water vapor
- In analysis nudging, these terms nudge point-by-point to a 3d space- and time-interpolated analysis field
- In obs-nudging, points near observations are nudged based on model error at obs site
- The nudging is a relaxation term with a user-defined time scale around an hour or more
- Nudging will work with nesting and restarts

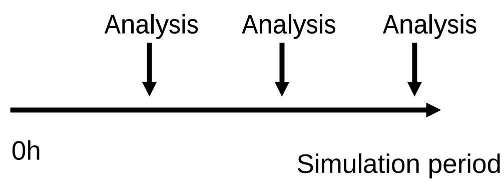
Dynamic Initialization

- Model domains are nudged towards analysis in a pre-forecast period of 6-12 hours
- This has benefit of smooth start up at forecast time zero



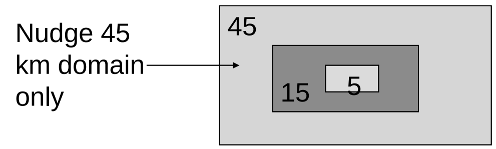
Four-Dimensional Met Analysis

- Produces analyses between normal analysis times
- High-resolution balanced and mass-continuity winds can be output to drive off-line air quality models



Boundary Conditions

- Nudge an outer domain towards analysis through forecast
- This has benefit of providing smoother boundary conditions to domain of interest than if 15 km domain is the outer domain with interpolated-analysis boundary conditions



FDDA Methods

- Two Methods
  - Grid or analysis nudging (suitable for coarse resolution)
  - Observation or station nudging (suitable for fine-scale or asynoptic obs)
- Nudging can be applied to winds, temperature, and water vapor

Note: nudging terms are fake sources, so avoid FDDA use in dynamics or budget studies

Analysis Nudging (grid_fdda=1)

- Each grid-point is nudged towards a value that is time-interpolated from analyses

$$\frac{\partial p^* \alpha}{\partial t} = F(\alpha, \mathbf{x}, t) + G_\alpha \cdot W_\alpha \cdot \epsilon_\alpha(\mathbf{x}) \cdot p^*(\hat{\alpha}_0 - \alpha)$$

In WRF p* is mu

Analysis Nudging

$$\frac{\partial p^* \alpha}{\partial t} = F(\alpha, \mathbf{x}, t) + G_\alpha \cdot W_\alpha \cdot \epsilon_\alpha(\mathbf{x}) \cdot p^*(\hat{\alpha}_0 - \alpha)$$

- G is nudging inverse time scale
- W is vertical weight (upper air and surface)
- ♏ is a horizontal weight for obs density (not implemented yet)

Analysis Nudging

- 3d analysis nudging uses the WRF input fields at multiple times that are put in wrffdda file by program real when run with grid_fdda=1
  - With low time-resolution analyses, it is recommended not to use 3d grid-nudging in the boundary layer, especially for temperature
- Surface (2d) analysis nudging not available yet (in Version 3.1)

Analysis-Nudging namelist options

Can choose

- Frequency of nudging calculations (fgdt in minutes)
- Nudging time scale for each variable (guv, gt, gq in inverse seconds)
- Which variables not to nudge in the PBL (if_no_pbl_nudging_uv, etc.)
- Model level for each variable below which nudging is turned off (if_zfac_uv, k_zfac_uv, etc.)
- Ramping period over which nudging is turned off gradually (if_ramping, dt_ramp_min)

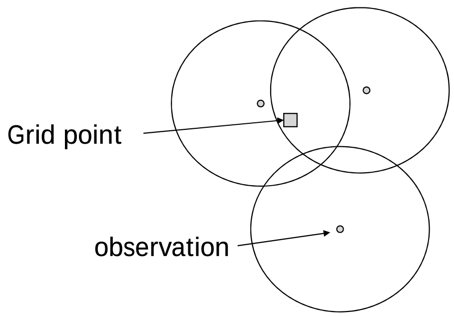
Obs Nudging (obs_nudge_opt=1)

- Each grid point is nudged using a weighted average of differences from observations within a radius of influence and time window

$$\frac{\partial p^* \alpha}{\partial t} = F(\alpha, \mathbf{x}, t) + G_\alpha \cdot p^* \frac{\sum_{i=1}^{N} W_i^2(\mathbf{x}, t) \cdot \gamma_i \cdot (\alpha_o - \hat{\alpha})_i}{\sum_{i=1}^{N} W_i(\mathbf{x}, t)}$$

$$W(\mathbf{x}, t) = w_{xy} \cdot w_\sigma \cdot w_t$$

Obs Nudging



Obs Nudging

$$w_{xy} = \frac{R^2 - D^2}{R^2 + D^2} \qquad 0 \le D \le R$$

$$w_{xy} = 0 \qquad D > R,$$

• R is radius of influence
• D is distance from ob modified by elevation difference

Obs Nudging

$$w_t = 1 \qquad |t - t_0| < \tau/2$$

$$w_t = \frac{\tau - |t - t_0|}{\tau/2} \qquad \tau/2 \le |t - t_0| \le \tau$$

• ♦ is the specified time window for the obs
• This is a function that ramps up and down

Obs Nudging

▪ w, is the vertical weighting – usually the vertical influence is set small (0.005 sigma) so that data is only assimilated on its own sigma level
▪ obs input file is a special ascii file with obs sorted in chronological order
  ▪ each record is the obs (u, v, T, Q) at a given model position and time
  ▪Utility programs exist to convert data to this format from other common formats (see www.mmm.ucar.edu/wrf/users/wrfv2/How_to_run_obs_fdda.html?)

Obs-Nudging namelist options

Can choose

- Frequency of nudging calculations (iobs_ionf)
- Nudging time scale for each variable (obs_coef_wind, etc.)
- Horizontal and vertical radius of influence (obs_rinxy, obs_rinsig)
- Time window (obs_twindo)
- Ramping period over which nudging is turned off gradually (obs_idynin, obs_dtramp)

FDDA Summary

- FDDA grid nudging is suitable for coarser grid sizes where analysis can be better than model-produced fields
- Obs nudging can be used to assimilate asynoptic or high-frequency observations
- Grid and obs nudging can be combined
- FDDA has fake sources and sinks and so should not be used on the domain of interest and in the time period of interest for scientific studies and simulations

Further plans

- Add 2d (surface) nudging and integrate with 3d nudging (in 3.1)
- Integrate with analyses produced by WRF-Var

WRF Physics

WRF Physics Options

Jimy Dudhia

WRF Physics

- Turbulence/Diffusion (diff_opt, km_opt)
- Radiation
  - Longwave (ra_lw_physics)
  - Shortwave (ra_sw_physics)
- Surface
  - Surface layer (sf_sfclay_physics)
  - Land/water surface (sf_surface_physics)
- PBL (bl_physics)
- Cumulus parameterization (cu_physics)
- Microphysics (mp_physics)

Turbulence/Diffusion

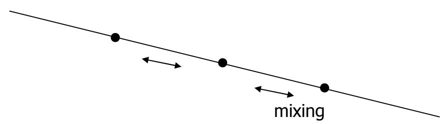
Sub-grid eddy mixing effects on all fields

ARW only

diff_opt=1

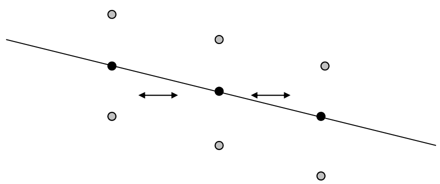
- 2^{nd} order diffusion on model levels
  - Constant vertical coefficient (kvdif) or use with PBL
  - For theta, only perturbation from base state is diffused
- km_opt
  - 1: constant (khdif and kvdif used)
  - 2: 1.5-order TKE prediction (not recommended with diff_opt=1)
  - 3: Smagorinsky (deformation/stability based K) (not recommended with diff_opt=1)
  - 4: 2D Smagorinsky (deformation based on horizontal wind for horizontal diffusion only)

Difference between diff_opt 1 and 2



diff_opt=1
Horizontal diffusion acts along model levels
Simpler numerical method with only neighboring points on the same model level

Difference between diff_opt 1 and 2



diff_opt=2
Horizontal diffusion acts along model levels
Numerical method includes vertical correction term using more grid points

ARW only

diff_opt=2

- 2nd order horizontal diffusion
- Allows for terrain-following coordinate
- km_opt
  - 1: constant (khdif and kvdif used)
  - 2: 1.5-order TKE prediction
  - 3: Smagorinsky (deformation/stability based K)
  - 4: 2D Smagorinsky (deformation based on horizontal wind for horizontal diffusion only)

ARW only

diff_opt=2 (continued)

- mix_full_fields=.true.: vertical diffusion acts on full (not perturbation) fields (recommended, but default = .false.)
- Idealized constant surface fluxes can be added in diff_opt=2 using namelist (dynamics section). Not available for diff_opt=1.
  - tke_drag_coefficient (C_D)
  - tke_heat_flux ($=H/\rho c_p$)
  - Must use isfflx=0 to use these switches

ARW only

diff_opt=2 (continued)

- Explicit large-eddy simulation (LES) PBL in real-data cases (V3) or idealized cases
  - sf_sfclay_physics (only option 1 currently) and sf_surface_physics (choose non-zero option)
  - bl_pbl_physics = 0
  - isfflx = 1 (drag and heat flux from physics) OR
  - isfflx = 2 (drag from physics, heat flux from tke_heat_flux)
  - km_opt = 2 or 3
- Not available for diff_opt=1.

Diffusion Option Choice

- Real-data case with PBL physics on
  - Best is diff_opt=1, km_opt=4
  - This complements vertical diffusion done by PBL scheme
- High-resolution real-data cases (~100 m grid)
  - No PBL
  - diff_opt=2; km_opt=2,3 (tke or Smagorinsky scheme)
- idealized cloud-resolving modeling (smooth or no topography)
  - diff_opt=2; km_opt=2,3
- Complex topography with no PBL scheme
  - diff_opt=2 is more accurate for sloped coordinate surfaces, and prevents diffusion up/down valley sides
- Note: WRF can run with no diffusion (diff_opt=0)

ARW only

diff_6th_opt

- 6th order optional added horizontal diffusion on model levels
  - Used as a numerical filter for 2*dx noise
  - Suitable for idealized and real-data cases
  - Affects all advected variables including scalars
- diff_6th_opt
  - 0: none (default)
  - 1: on (can produce negative water)
  - 2: on and prohibit up-gradient diffusion (better for water conservation)
- diff_6th_factor
  - Non-dimensional strength (typical value 0.12, 1.0 corresponds to complete removal of 2*dx wave in a time-step)

ARW only

damp_opt=1

- Upper level diffusive layer
- Enhanced horizontal diffusion at top
- Also enhanced vertical diffusion at top for diff_opt=2
- Cosine function of height
- Uses additional parameters
  - zdamp: depth of damping layer
  - dampcoef: nondimensional maximum magnitude of damping
- Works for idealized cases and real-data since 2.2 release

ARW only

damp_opt=2

- Upper level relaxation towards 1-d profile
- Rayleigh (relaxation) layer
- Cosine function of height
- Uses additional parameters
  - zdamp: depth of damping layer
  - dampcoef: inverse time scale (s^{-1})
- Works for idealized cases only

ARW only

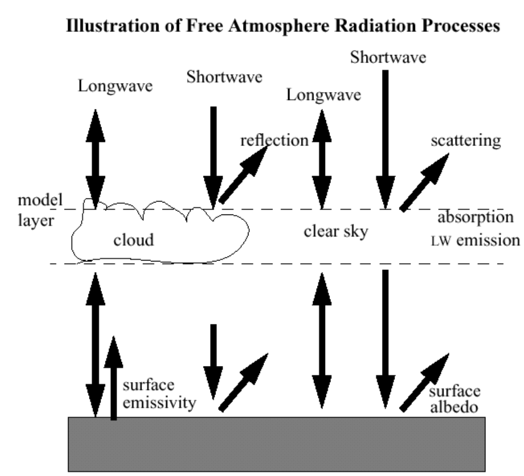
damp_opt=3

- "W-Rayleigh" (relaxation) layer
- Upper level relaxation towards zero vertical motion
- Cosine function of height
- Uses additional parameters
  - zdamp: depth of damping layer
  - dampcoef: inverse time scale (s^{-1})
- Works for idealized and real-data cases
- Applied in small time-steps (dampcoef=0.2 is stable)

Radiation

Atmospheric temperature tendency

Surface radiative fluxes



ra_lw_physics=1

RRTM scheme

- Spectral scheme
- K-distribution
- Look-up table fit to accurate calculations
- Interacts with clouds (1/0 fraction)
- Ozone profile specified
- CO2 constant (well-mixed)

ARW only

ra_lw_physics=3

CAM3 scheme

- Spectral scheme
- 8 longwave bands
- Look-up table fit to accurate calculations
- Interacts with clouds
- Can interact with trace gases and aerosols
- Ozone profile function of month, latitude
- CO2 fixed constant

ra_lw_physics=99

GFDL longwave scheme

- used in Eta/NMM
- Default code is used with Ferrier microphysics
  - Remove #define to compile for use without Ferrier
- Spectral scheme from global model
- Also uses tables
- Interacts with clouds (cloud fraction)
- Ozone profile based on season, latitude
- CO2 fixed

ra_sw_physics=1

MM5 shortwave (Dudhia)

- Simple downward calculation
- Clear-sky scattering
  - swrad_scat tuning parameter
    - 1.0 = 10% scattered, 0.5=5%, etc.
- Water vapor absorption
- Cloud albedo and absorption
- Version 3 has slope_rad and topo_shading switches (0,1) to turn on slope and shading effects in this radiation option only

ARW only

ra_sw_physics=2

Goddard shortwave

- Spectral method
- Interacts with clouds
- Ozone profile (tropical, summer/winter, mid-lat, polar)
- CO2 fixed

ARW only

ra_sw_physics=3

CAM3 shortwave

- Spectral method (19 bands)
- Interacts with clouds
- Ozone/CO2 profile as in CAM longwave
- Can interact with aerosols and trace gases
- Note: CAM schemes need some extra namelist items (see README.namelist)

ra_sw_physics=99

GFDL shortwave

- Used in Eta/NMM model
- Default code is used with Ferrier microphysics (see GFDL longwave)
- Ozone/CO2 profile as in GFDL longwave
- Interacts with clouds (and cloud fraction)

ARW only

radt

Radiation time-step recommendation

- Radiation is too expensive to call every step
- Frequency should resolve cloud-cover changes with time
- radt=1 minute per km grid size is about right (e.g. radt=10 for dx=10 km)
- Each domain can have its own value but recommend using same value on all 2-way nests

NMM only

nrads/nradl

Radiation time-step recommendation

- Number of fundamental steps per radiation call
- Operational setting should be 3600/dt
- Higher resolution could be used, e.g. 1800/dt
- Recommend same value for all nested domains

Surface schemes

Surface layer of atmosphere diagnostics (exchange/transfer coeffs)

Land Surface: Soil temperature /moisture /snow prediction /sea-ice temperature

Surface Fluxes

- Heat, moisture and momentum

$$H = \rho c_p u_* \theta_* \qquad E = \rho u_* q_* \qquad \tau = \rho u_* u_*$$

$$u_* = \frac{kV_r}{\ln(z_r / z_0) - \psi_m} \qquad \theta_* = \frac{k\Delta\theta}{\ln(z_r / z_{0h}) - \psi_h} \qquad q_* = \frac{k\Delta q}{\ln(z_r / z_{0q}) - \psi_h}$$

Subscript *r* is reference level (lowest model level, or 2 m or 10 m)
z_0 are the roughness lengths

Roughness Lengths

- Roughness lengths are a measure of the "initial" length scale of surface eddies, and generally differ for velocity and scalars
- Roughness length depends on land-use type
- Some schemes use smaller roughness length for heat than for momentum
- For water points roughness length is a function of surface wind speed

Exchange Coefficient

- C_{hs} is the exchange coefficient for heat, defined such that

$$H = \rho c_p C_{hs} \Delta\theta$$

It is related to the roughness length and u* by

$$C_{hs} = \frac{ku_*}{\ln\left(\frac{z}{z_0}\right) - \psi_h}$$

sf_sfclay_physics=1

Monin-Obukhov similarity theory

- Taken from standard relations used in MM5 MRF PBL
- Provides exchange coefficients to surface (land) scheme
- Should be used with bl_pbl_physics=1 or 99

sf_sfclay_physics=2

Monin-Obukhov similarity theory

- Modifications due to Janjic
- Taken from standard relations used in NMM model, including Zilitinkevich thermal roughness length
- Should be used with bl_pbl_physics=2

NMM only

sf_sfclay_physics=3

GFS Monin-Obukhov similarity theory

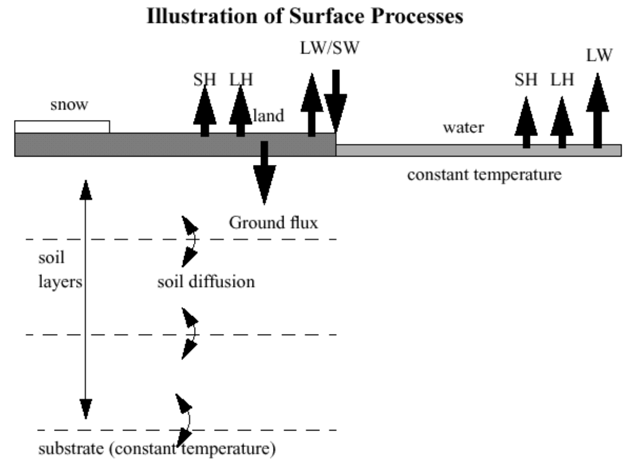
- For use with NMM-LSM
- Should be used with bl_pbl_physics=3

ARW only

sf_sfclay_physics=7

Pleim-Xiu surface layer (EPA)

- For use with PX LSM and ACM PBL
  - Should be used with sf_surface_physics=7 and bl_pbl_physics=7
- New in Version 3



ARW only

sf_surface_physics=1

5-layer thermal diffusion model from MM5

- Predict ground temp and soil temps
- Thermal properties depend on land use
- No effect for water (Version 3 has ocean mixed-layer model for hurricane applications)
- No soil moisture or snow-cover prediction
- Moisture availability based on land-use only
- Provides heat and moisture fluxes for PBL
- May be available for NMM in Version 3

sf_surface_physics=2

Noah Land Surface Model (Unified ARW/NMM version in Version 3)

- Vegetation effects included
- Predicts soil temperature and soil moisture in four layers
- Predicts snow cover and canopy moisture
- Handles fractional snow cover and frozen soil
- Diagnoses skin temp and uses emissivity
- Provides heat and moisture fluxes for PBL
- 2.2 has Urban Canopy Model option (ucmcall=1, ARW only)

sf_surface_physics=3

RUC Land Surface Model (Smirnova)

- Vegetation effects included
- Predicts soil temperature and soil moisture in six layers
- Multi-layer snow model
- Provides heat and moisture fluxes for PBL

ARW only

sf_surface_physics=7

Pleim-Xiu Land Surface Model (EPA)

- New in Version 3
- Vegetation effects included
- Predicts soil temperature and soil moisture in two layers
- Simple snow-cover model
- Provides heat and moisture fluxes for PBL

LANDUSE.TBL

Text (ASCII) file that has land-use properties (vegetation, urban, water, etc.)

- 24 USGS categories from 30" global dataset
- Each type is assigned summer/winter value
  - Albedo
  - Emissivity
  - Roughness length
- Other table properties (thermal inertia, moisture availability, snow albedo effect) are used by 5-layer model
- Also note
  - Other tables (VEGPARM.TBL, etc.) are used by Noah
  - RUC LSM uses same table files after Version 3

Initializing LSMs

- Noah and RUC LSM require additional fields for initialization
  - Soil temperature
  - Soil moisture
  - Snow liquid equivalent
- These are in the Grib files, but are not from observations
- They come from "offline" models driven by observations (rainfall, radiation, surface temperature, humidity wind)

Initializing LSMs

- There are consistent model-derived datasets for Noah and RUC LSMs
  - Eta/GFS/AGRMET/NNRP for Noah (although some have limited soil levels available)
  - RUC for RUC
- But, resolution of mesoscale land-use means there will be inconsistency in elevation, soil type and vegetation
- This leads to spin-up as adjustments occur in soil temperature and moisture
- This spin-up can only be avoided by running offline model on the same grid (e.g. HRLDAS for Noah)
- Cycling land state between forecasts also helps, but may propagate errors (e.g in rainfall effect on soil moisture)

ARW only

sst_update=1

Reads lower boundary file periodically to update the sea-surface temperature (otherwise it is fixed with time)

- For long-period simulations (a week or more)
- wrflowinp_d0*n* created by *real*
- Sea-ice not updated
  - Update available in Version 3
- Vegetation fraction update is included
  - Background albedo in Version 3

ARW only

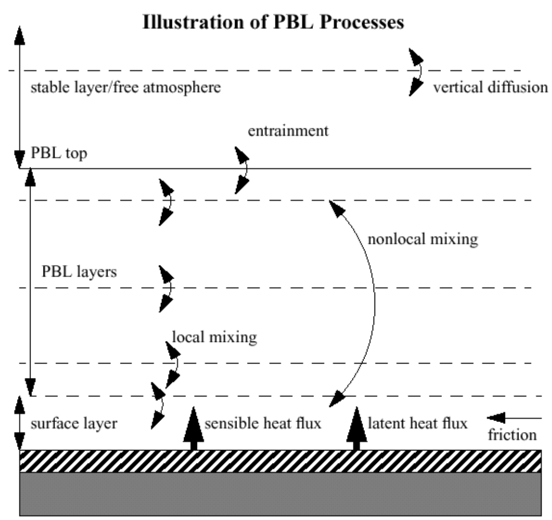
Hurricane Options

- Ocean Mixed Layer Model (omlcall=1)
  - Use with sf_surface_physics=1
  - 1-d slab ocean mixed layer (specified initial depth)
  - Includes wind-driven ocean mixing for SST cooling feedback
- Alternative surface-layer option for high-wind ocean surface (isftcflx=1)
  - Use with sf_sfclay_physics=1
  - Modifies Charnock relation to give less surface friction at high winds (lower Cd)
  - Modifies surface enthalpy (Ck, heat/moisture) formulation

Planetary Boundary Layer

Boundary layer fluxes (heat, moisture, momentum)

Vertical diffusion



bl_pbl_physics=1

YSU PBL scheme (Hong, Noh and Dudhia 2006)

- Parabolic non-local-K mixing in dry convective boundary layer
- Troen-Mahrt countergradient term (non-local flux)
- Depth of PBL determined from thermal profile
- Explicit treatment of entrainment
- Vertical diffusion depends on Ri in free atmosphere
- New stable surface BL mixing using bulk Ri
- Available for NMM in Version 3

bl_pbl_physics=2

Mellor-Yamada-Janjic (Eta/NMM) PBL

- 1.5-order, level 2.5, TKE prediction
- Local TKE-based vertical mixing in boundary layer and free atmosphere
- TKE_MYJ is advected by NMM, not by ARW (yet)

NMM only

bl_pbl_physics=3

GFS PBL

- 1st order Troen-Mahrt
- Closely related to MRF PBL
- Non-local-K vertical mixing in boundary layer and free atmosphere

ARW only

bl_pbl_physics=7

Asymmetrical Convective Model, Version 2 (ACM2) PBL (Pleim and Chang)

- Blackadar-type thermal mixing upwards from surface layer
- Local mixing downwards
- PBL height from critical bulk Richardson number

ARW only

bl_pbl_physics=99

MRF PBL scheme (Hong and Pan 1996)

- Non-local-K mixing in dry convective boundary layer
- Depth of PBL determined from critical Ri number
- Vertical diffusion depends on Ri in free atmosphere

ARW only

bldt

- Minutes between boundary layer/LSM calls
- Typical value is 0 (every step)

NMM only

nphs

- Time steps between PBL/turbulence/LSM calls
- Typical value is 10 for efficiency
- Also used for microphysics

PBL Scheme Options

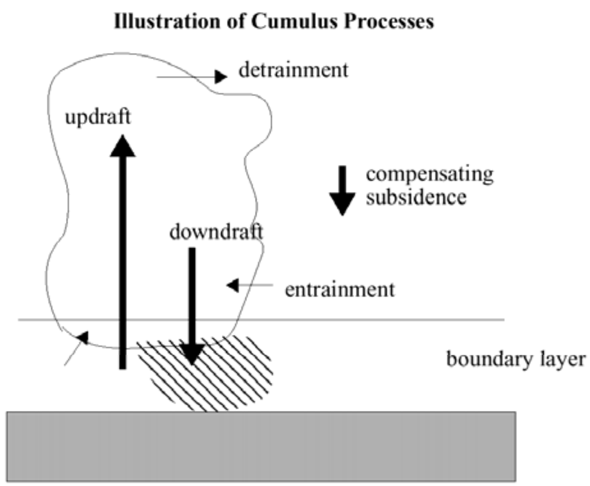
PBL schemes can be used for most grid sizes when surface fluxes are present

- With PBL scheme, lowest full level should be .99 or .995 (not too close to 1)
- Assumes that PBL eddies are not resolved
- At grid size dx << 1 km, this assumption breaks down
- Can use 3d diffusion instead of a PBL scheme in Version 3 (coupled to surface physics)
  - Works best when dx and dz are comparable

Cumulus Parameterization

Atmospheric heat and moisture/cloud tendencies

Surface rainfall



cu_physics=1

New Kain-Fritsch

- As in MM5 and Eta/NMM test version
- Includes shallow convection (no downdrafts)
- Low-level vertical motion in trigger function
- CAPE removal time scale closure
- Mass flux type with updrafts and downdrafts, entrainment and detrainment
- Includes clour, rain, ice, snow detrainment
- Clouds persist over convective time scale (recalculated every convective step in NMM)

cu_physics=2

Betts-Miller-Janjic

- As in NMM model (Janjic 1994)
- Adjustment type scheme
- Deep and shallow profiles
- BM saturated profile modified by cloud efficiency, so post-convective profile can be unsaturated in BMJ
- No explicit updraft or downdraft
- No cloud detrainment
- Scheme changed significantly since V2.1

cu_physics=3

Grell-Devenyi Ensemble

- Multiple-closure (e.g. CAPE removal, quasi-equilibrium) - 16 mass flux closures
- Multi-parameter (e.g maximum cap, precipitation efficiency) - e.g. 3 cap strengths, 3 mass flux profiles
- Explicit updrafts/downdrafts
- Includes cloud and ice detrainment
- Mean feedback of ensemble is applied
- Weights can be tuned (spatially, temporally) to optimize scheme (training)

NMM only

cu_physics=4

Simpified Arakawa-Schubert (SAS) GFS scheme

- Quasi-equilibrium scheme
- Related to Grell scheme in MM5
- Includes cloud and ice detrainment
- Downdrafts and single, simple cloud

cu_physics=5

Grell-3d

- Smaller ensemble than GD
- Explicit updrafts/downdrafts
- Includes cloud and ice detrainment
- Subsidence is spread to neighboring columns
  - This makes it more suitable for < 10 km grid size than other options
- Mean feedback of ensemble is applied
- Weights can be tuned (spatially, temporally) to optimize scheme (training)

ARW only

cudt

- Time steps between cumulus scheme calls
- Typical value is 5 minutes

NMM only

ncnvc

- Time steps between cumulus parameterization calls
- Typically 10 - same as NPHS

Cumulus scheme

Recommendations about use

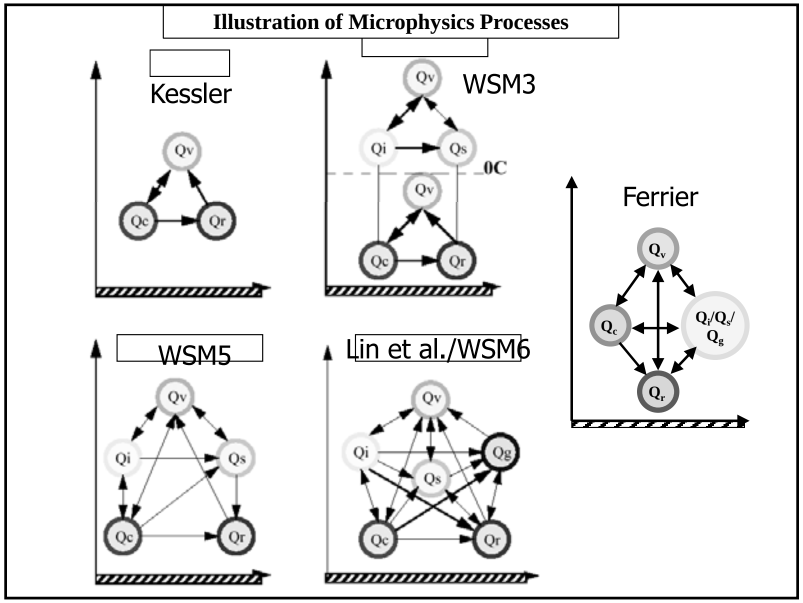
- For dx ≥ 10 km: probably need cumulus scheme
- For dx ≤ 3 km: probably do not need scheme
  - However, there are cases where the earlier triggering of convection by cumulus schemes help
- For dx=3-10 km, scale separation is a question
  - No schemes are specifically designed with this range of scales in mind
- Issues with 2-way nesting when physics differs across nest boundaries (seen in precip field on parent domain)
  - best to use same physics in both domains or 1-way nesting

Microphysics

Atmospheric heat and moisture tendencies

Microphysical rates

Surface rainfall



ARW only

mp_physics=1

Kessler scheme

- Warm rain – no ice
- Idealized microphysics
- Time-split rainfall

ARW only

mp_physics=2

Purdue Lin et al. scheme

- 5-class microphysics including graupel
- Includes ice sedimentation and time-split fall terms

ARW only

mp_physics=3

WSM 3-class scheme

- From Hong, Dudhia and Chen (2004)
- Replaces NCEP3 scheme
- 3-class microphysics with ice
- Ice processes below 0 deg C
- Ice number is function of ice content
- Ice sedimentation and time-split fall terms

mp_physics=4

WSM 5-class scheme

- Also from Hong, Dudhia and Chen (2004)
- Replaces NCEP5 scheme
- 5-class microphysics with ice
- Supercooled water and snow melt
- Ice sedimentation and time-split fall terms

mp_physics=5

Ferrier (current NAM) scheme

- Designed for efficiency
  - Advection only of total condensate and vapor
  - Diagnostic cloud water, rain, & ice (cloud ice, snow/graupel) from storage arrays – assumes fractions of water & ice within the column are fixed during advection
- Supercooled liquid water & ice melt
- Variable density for precipitation ice (snow/graupel/sleet) – "rime factor"

mp_physics=6

WSM 6-class scheme

- From Hong and Lim (2006, JKMS)
- 6-class microphysics with graupel
- Ice number concentration as in WSM3 and WSM5
- New combined snow/graupel fall speed
- Time-split fall terms with melting

ARW only

mp_physics=7

Goddard 6-class scheme

- From Tao et al.
- 6-class microphysics with graupel
- Based on Lin et al. with modifications for ice/water saturation
- gsfcgce_hail switch for hail/graupel properties
- gsfcgce_2ice switch for removing graupel or snow processes
- Time-split fall terms with melting

mp_physics=8

Thompson et al. graupel scheme

- From Thompson et al. (2006, WRF workshop)
- Newer version of Thompson et al. (2004) scheme
- Updated significantly for 2.2
- 6-class microphysics with graupel
- Ice number concentration also predicted (double-moment ice)
- Time-split fall terms

ARW only

mp_physics=10

Morrison 2-moment scheme

- New in Version 3.0
- 6-class microphysics with graupel
- Number concentrations also predicted for ice, snow, rain, and graupel (double-moment)
- Time-split fall terms

ARW only

no_mp_heating=1

- Turn off heating effect of microphysics
  - Zeroes out the temperature tendency
  - Equivalent to no latent heat
  - Other microphysics processes not affected
  - New in Version 3

ARW only

mp_zero_out

Microphysics switch (also mp_zero_out_thresh)

- 1: all values less than threshold set to zero (except vapor)
- 2: as 1 but vapor also limited ≥ 0
- Note: this option will not conserve total water
- Not needed when using positive definite advection
- NMM: Recommend mp_zero_out=0

NMM only

nphs

- Time steps between microphysics calls
- Same as parameter for turbulence/PBL/LSM
- Typical value is 10 for efficiency

Microphysics Options

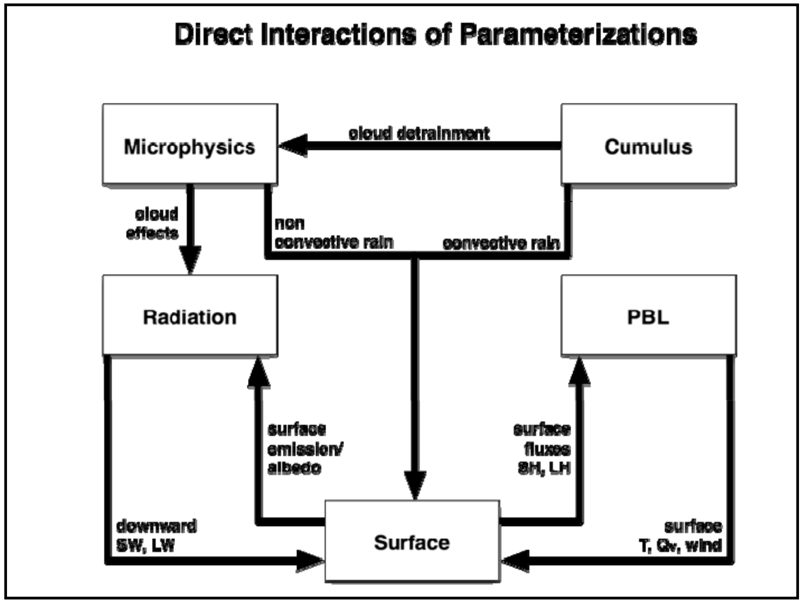
Recommendations about choice

- Probably not necessary to use a graupel scheme for dx > 10 km
  - Updrafts producing graupel not resolved
  - Cheaper scheme may give similar results
- When resolving individual updrafts, graupel scheme should be used
- All domains use same option

Rainfall Output

- Cumulus and microphysics can be run at the same time
- WRF outputs rainfall accumulations since simulation start time (0 hr) in mm
- RAINC comes from cumulus scheme
- RAINNC comes from microphysics scheme
- Total is RAINC+RAINNC
  - RAINNCV is time-step value
  - SNOWNC/SNOWNCV are snow sub-set of RAINC/RAINNCV (also GRAUPELNC, etc.)

Physics Interactions



&physics

Seven major physics categories:

- **mp_physics: 0,1,2,3,4,5,6,8,10**
- **ra_lw_physics: 0,1,3,99**
- **ra_sw_physics: 0,1,2,3,99**
- **sf_sfclay_physics: 0,1,2**
- **sf_surface_physics: 0,1,2,3,99** (set before running **real** or **ideal**, need to match with **num_soil_layers** variable)
  - **ucm_call = 0,1**
- **bl_pbl_physics: 0,1,2,99**
- **cu_physics: 0,1,2,3,99**

Post-processing Tools (3): RIP

Post-processing Tools: RIP4
(WRF-ARW & WRF-NMM)

Cindy Bruyère

RIP4

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

Mesoscale & Mircoscale Meteorology Division / NCAR

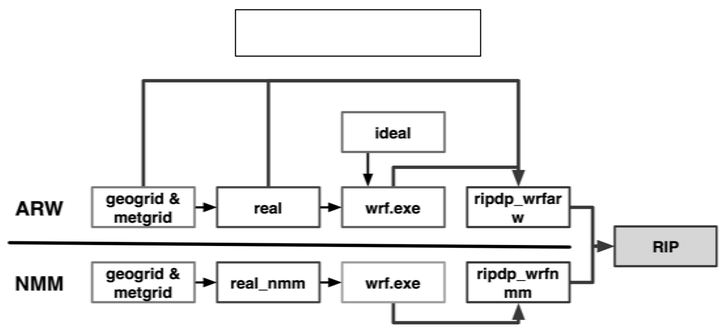
RIP4

Important versions

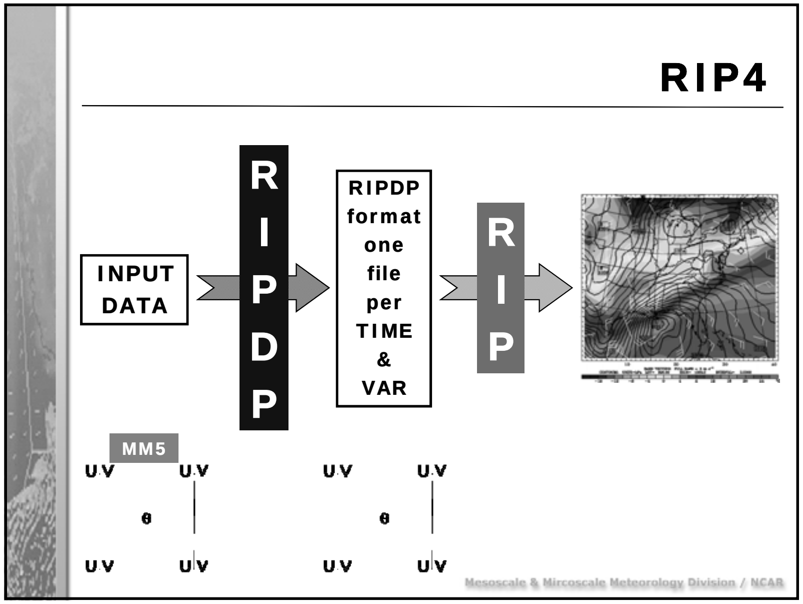
ARW WRF output:	**version 4.0**
ARW idealized data:	**version 4.1**
WPS (ARW WRF):	**version 4.2**
NMM WRF output:	**version 4.3**
Current version:	**version 4.4**

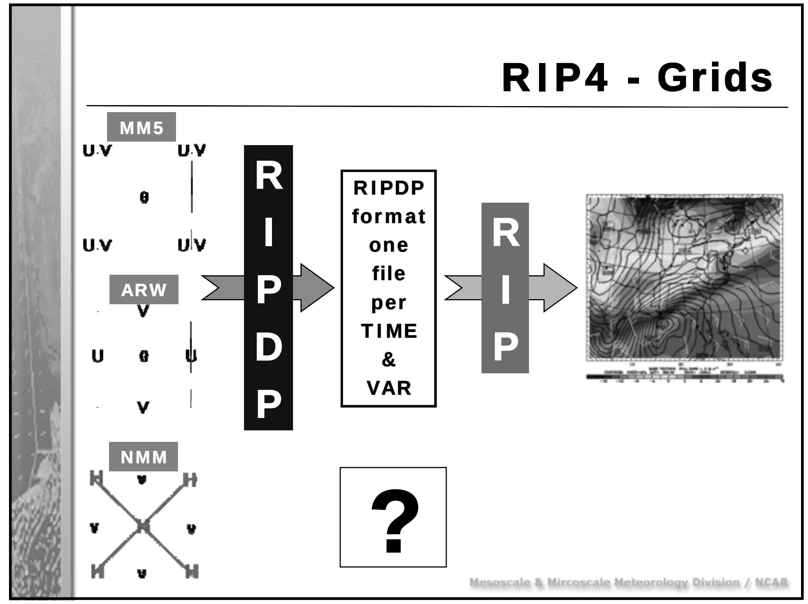
Mesoscale & Mircoscale Meteorology Division / NCAR

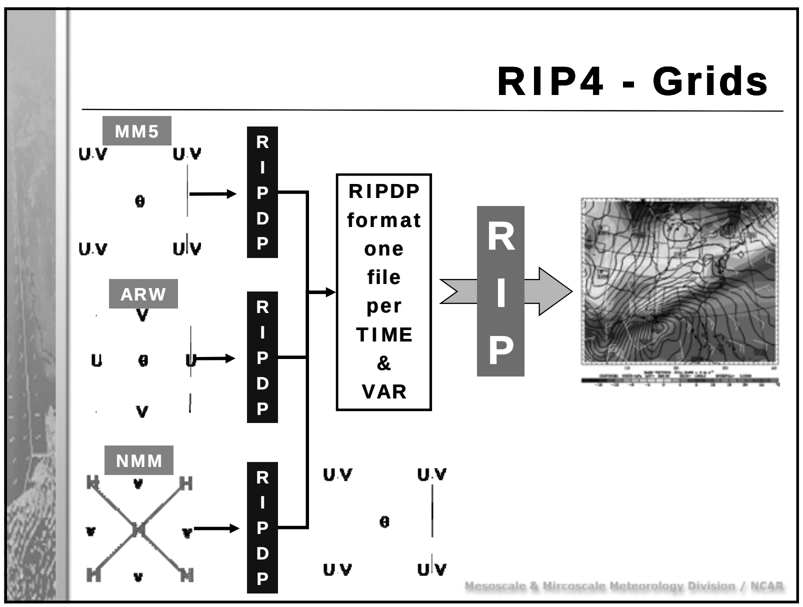
RIP4 Input Data

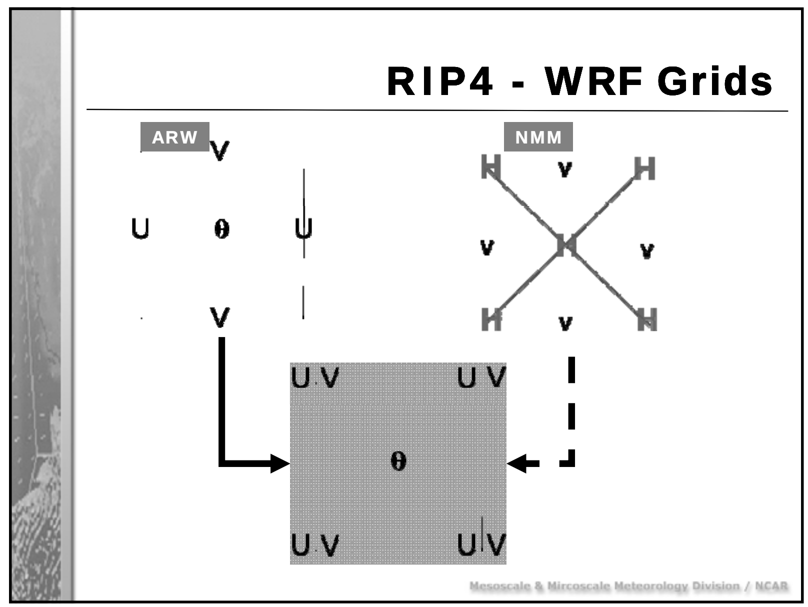


Mesoscale & Mircoscale Meteorology Division / NCAR

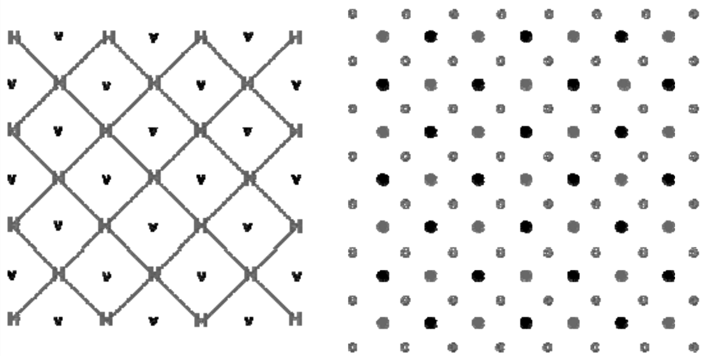

RIP4
INPUT DATA
RIPDP
RIPDP format one file per TIME & VAR
RIP
MM5
U.V U.V
θ
U.V U.V
U.V U.V
θ
U.V U.V
Mesoscale & Mircoscale Meteorology Division / NCAR


RIP4 - Grids
MM5
U.V U.V
θ
U.V U.V
ARW
V
U θ U
V
NMM
H v H
v H v
H v H
RIPDP
RIPDP format one file per TIME & VAR
RIP
?
Mesoscale & Mircoscale Meteorology Division / NCAR


RIP4 - Grids
MM5
U.V U.V
θ
U.V U.V
ARW
V
U θ U
V
NMM
H v H
v H v
H v H
RIPDP
RIPDP
RIPDP
RIPDP format one file per TIME & VAR
RIP
U.V U.V
θ
U.V U.V
Mesoscale & Mircoscale Meteorology Division / NCAR

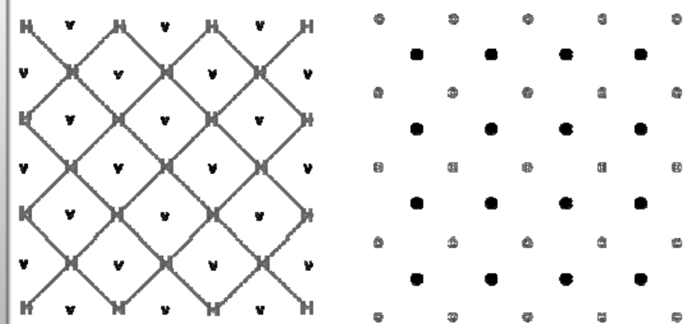

RIP4 - WRF Grids
ARW
V
U θ U
V
NMM
H v H
v H v
H v H
U.V U.V
θ
U.V U.V
Mesoscale & Mircoscale Meteorology Division / NCAR

RIP4 - NMM Grid *(iinterp 0)*



Mesoscale & Mircoscale Meteorology Division / NCAR

RIP4 - NMM Grid *(iinterp 1)*



new projection ; no direct relationship

Mesoscale & Mircoscale Meteorology Division / NCAR

General

- **Requires NCAR Graphics low-level routines**
  - ***http://ngwww.ucar.edu***
- **NCL Version 5:**
  - ***http://www.ncl.ucar.edu***
  - Released November 2007
  - Combine NCL and NCAR Graphics
  - Open Source
  - *Recommended*

Mesoscale & Mircoscale Meteorology Division / NCAR

General

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

Mesoscale & Mircoscale Meteorology Division / NCAR

General

- **Download Code:**
  - ***http://www.mmm.ucar.edu/wrf/users/download/get_source.html***
  - ***http://www.dtcenter.org/wrf-nmm/users/downloads/index.php***
- **OnLine Tutorial:**
  - ***http://www.mmm.ucar.edu/wrf/users/graphics/RIP4/RIP4.htm***
  - ***http://www.dtcenter.org/wrf-nmm/users/OnLineTutorial/NMM/RIP/index.php***

RIP4 on your computer

- **set environment variables**

  setenv RIP_ROOT /usr/$USER/RIP4 (rip_root)
  setenv NCARG_ROOT /usr/local/ncarg (/usr/local/ncl)

- **Edit *Makefile* to define paths to netCDF library and include file on your computer:**

  NETCDFLIB and *NETCDFINC*

- **make <machine type>** (*it'll make suggestions*)

  make linux (*example*)

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

  ripdp_mm5
  ripdp_wrfarw
  ripdp_wrfnmm

ripdp

- **ripdp_wrf*xxx***
  RIP Data Preparation for WRF (ARW / NMM)
- **RIPDP** converts different input file formats *(WRF - netCDF)* into RIP input format *(B - grid)*
- **RIPDP** puts each **Variable** at each **Time** into a separate file – **LOTS** of files

  → **mkdir RIPDP**

Running ripdp

Optional

**ripdp_wrf*xxx* [–n namelist-file] **
**<model_data_name> [basic/all] **
<input_file1 input_file2>

Example

ripdp_wrfarw RIPDP/arw all wrfout*
ripdp_wrfnmm RIPDP/nmm all wrfout*

use directory as part of the model_data_name

ripdp namelist

- **Use namelist to add control**
  - **ptimes** *(times for ripdp to process)*
    - *0,1,2,3,4,5,6* *(0,1,2,3,4,5,6)*
    - *0,-6,1* *(0,1,2,3,4,5,6)*
    - *0, 2,-4,1, 6* *(0, 2,3,4, 6)*
  - ***tacc:*** *input files not on exact times*
    history_interval=10 ; time_step=180 (*3 min*)
    Output times **uneven** (29_00:00, 29_00:09, 29_00:21, 29_00:30)

    history_ interval=12 ;time_step=180 *(3 min)* Output times **even** (29_00:00, 29_00:12, 29_00:24, 29_00:36:00)
  - **discard:** fields if '*all*' is selected on the command line
  - **retain:** fields if 'basic' is selected on the command line

•NMM only

ripdp namelist

- **iinterp = 1:** interpolate to a new B-grid
- **dskmcib:** grid spacing, in km, of the coarse domain on which the new B-grid will be based
- **miycorsib, mjxcorsib:** number of grid points in the y and x directions of new B-grid
- **nprojib:** map projection number (0: none/ideal, 1: LC, 2: PS, 3: ME, 4: SRCE) of new B-grid
- **xlatcib, xloncib:** central latitude and longitude of new B-grid
- **truelat1ib, truelat2ib:** two true latitudes of new B-grid
- **miyib, mjxib:** number of grid points in the y and x directions, of the fine domain
- **yicornib, xjcornib:** coarse domain y and x locations of the lower left corner point of the fine domain
- **dskmib:** grid spacing, in km, of the fine domain

rip

- **read the output generated by *ripdp***
- **read User Input File (UIF) *(rip_sample.in)***
  - **First** section is a list of general parameters *(namelist format)*
  - **Second** section is a series of plots in the Plot Specification Table (PST)
- **generate meta file**

Running rip

- **Edit the User Input File (UIF)**
- **setenv NCARG_ROOT /usr/local/ncarg**
  setenv NCARG_ROOT /usr/local/ncl
  (if you installed NCL version 5)
- **setenv RIP_ROOT *your-rip-directory***
  - *Can overwrite this with rip_root in input namelist*

Running rip

created by ripdp

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

User Input File (UIF)

Example

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

use directory as part of the model_data_set_name

<u>output</u>

[rip_sample.out]
rip_sample.TYPE

rip UIF

&userin
……. } **Namelist controlling general parameters**
&end
&trajcalc
……. } **Namelist for trajectory calculations**
Only used if itrajcalc=1, in userin namelist
&end

```
=================================
---------- Plot Specification Table ----------
=================================
feld= ……… } Frame specification
feld= ……… } group (FSG)
=================================
feld= ………} Plot specification line (PSL)
feld= ………
=================================
```

Plot Specification Table (PST)

rip namelist - *&userin*

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

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

rip namelist - *&userin*

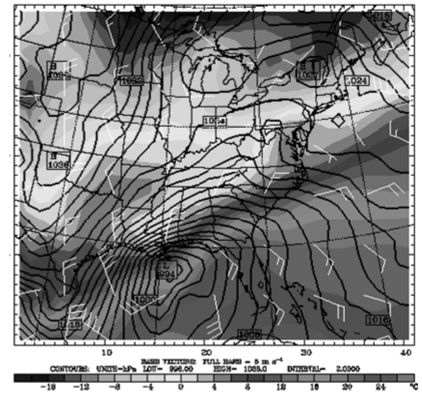
- **idotitle – first part of first title line**
- **titlecolor – color of title lines**
- **ptimes, ptimeunits – times to process**
- **tacc – tolerance for processing data**
- **timezone –display of local time**
- **iusedaylightrule – 1 applied, 0 not applied**
- **iinittime – plotting of initial time**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

rip namelist - *&userin*

- **ntextcd – text font**
- **fcoffset – 12 means hour 12 of the MM5 forecast is considered hour 0 by you**
- **idotser – generate time series output**
- **idescriptive – more descriptive titles**
- **icgmsplit – split metacode into several files**
- **maxfld – reserve memory for RIP (10-15)**
- **itrajcalc – 0, 1 ONLY when doing trajectory calculations *(use also namelist trajcalc)***
- **imakev5d – 0, 1 generate Vis5D data**
- **rip_root - override RIP_ROOT**
- **ncarg_root - output type: X11, cgm, pdf, ps**

Creating a Plot

Temperature @ lowest sigma level
Sea Level Pressure
Winds @ lowest sigma level



Mesoscale & Mircoscale Meteorology Division / NCAR

Creating a Plot

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

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

Mesoscale & Mircoscale Meteorology Division / NCAR

Summary: How to run RIP4?

- **Compile the code**
  make <*machine type*>
- **Run ripdp_wrf*xxx***
  Create a new directory for the output
- **Set environment variables**
  setenv NCARG_ROOT /usr/local/ncarg *(/usr/local/ncl)*
  setenv RIP_ROOT *your-rip-directory (or in namelist)*
- **Edit the User Input File (UIF)**
- **Run rip**

Mesoscale & Mircoscale Meteorology Division / NCAR

Post-processing Tools (4): ARWpost

Post-processing Tools: ARWpost & VAPOR *(WRF-ARW)*

Cindy Bruyère

ARWpost

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

General

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

General

- **MUST have WRF compiled *(similar to WPS)***
- **For GrADS output**
  - GrADS libraries only needed to display data
  - ***http://grads.iges.org/grads/grads.html***
  - GrADS libraries are free
- **For vis5d output**
  - vis5d libraries needed for compilation
  - ***http://www.ssec.wisc.edu/~billh/vis5d.html***
  - vis5d libraries are free

Configure & Compile

./configure

Make sure this is correct. If not, set environment variable NETCDF

```
Will use NETCDF in dir: /usr/local/netcdf-pgi
------------------------------------------------
Please select from among the following supported
platforms.
1. PC Linux i486 i586 i686, PGI compiler
   (no vis5d)
2. PC Linux i486 i586 i686, PGI compiler
   (vis5d)
3. PC Linux i486 i586 i686, Intel compiler
   (no vis5d)
4. PC Linux i486 i586 i686, Intel compiler
   (vis5d)

Enter selection [1-4] :
```

Configure & Compile

- **configure.arwp,** will be created
- If your WRF code is not compiled under **../WRFV3,** edit **configure.arwp,** and set **"WRF_DIR"** to the correct location of your WRFV3 code
- **./compile**
  - This will create **ARWpost.exe**

namelist.ARWpost

&datetime	
start_date ***end_date***	Start & end date Format: *YYYY-MM-DD_HH:mm:ss*
interval_seconds	Seconds between times to process. *Code will skip times not required.* *Data can be in multiple files.*
tacc	If model output is not at regular intervals, use next time if within *tacc* seconds of time requested. *2008-04-10_12:00:00* *2008-04-10_13:00:10* *tacc=10*
debug_level	Set high for extra information

namelist.ARWpost

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

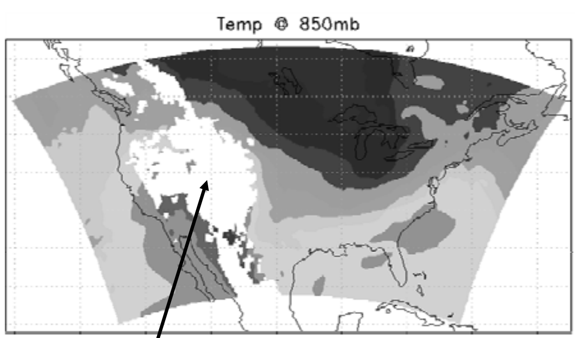
namelist.ARWpost

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

namelist.ARWpost

&interp	
interp_method	0 = sigma levels, -1 = code defined "nice" height levels, 1 = user defined height or pressure levels
interp_levels	Only used if interp_method=1 Supply levels to interpolate to, in hPa (*pressure*) or km (*height above sea level*) Supply levels bottom to top **NOTE**: *NO extrapolation below ground*

Interpolation



NO extrapolation below ground

Run

- **./ARWpost.exe**
- **Will create either,**

***output_root_name*.dat & *output_root_name*.ctl**

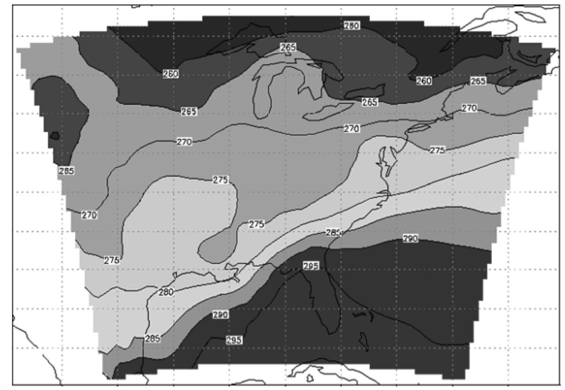
OR

***output_root_name*.v5d**

GrADS specific notes

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

GrADS - projections



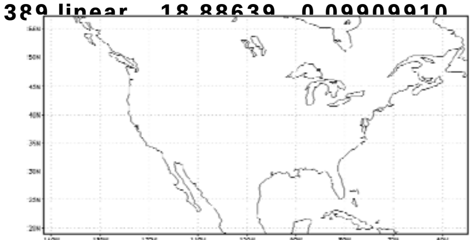
GrADS - .ctl file

dset ^test.dat
options byteswapped
undef 1.e37
title OUTPUT FROM WRF V2.2 MODEL
pdef 259 163 lcc 40.000 -98.000 130.000 82.000
60.00000 30.00000 -98.00000 22000.000
22000.000
xdef 877 linear -141.49254 0.09909910
ydef 389 linear 18.88639 0.09909910

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

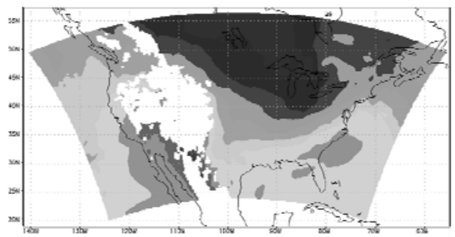
GrADS - .ctl file

dset ^test.dat
options byteswapped
undef 1.e37
title OUTPUT FROM WRF V2.2 MODEL
pdef 259 163 lcc 40.000 -98.000 130.000 82.000
60.00000 30.00000 -98.00000 22000.000
22000.000
xdef 877 linear -141.49254 0.09909910
ydef 389 linear 18.88639 0.09909910



GrADS - .ctl file

```
dset ^test.dat
options byteswapped
undef 1.e37
title OUTPUT FROM WRF V2.2 MODEL
pdef 259 163 lcc 40.000 -98.000 130.000 82.000
60.00000 30.00000 -98.00000 22000.000 22000.000
xdef 877 linear -141.49254 0.09909910
ydef 389 linear 18.88639 0.09909910
```

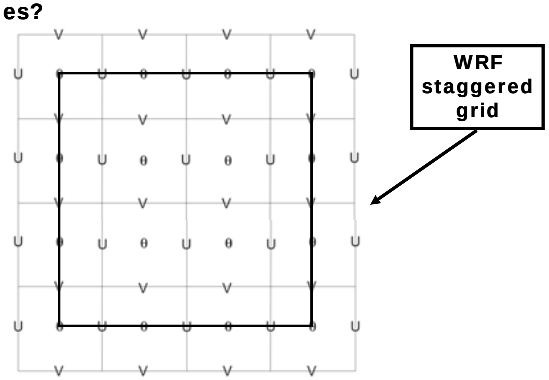


GrADS conversion - question

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

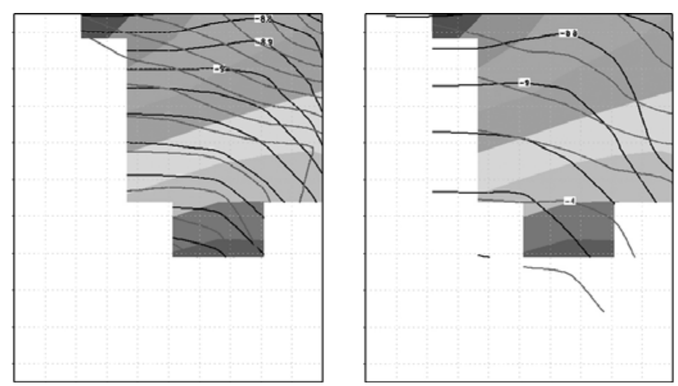
GrADS conversion - question

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



Staggering

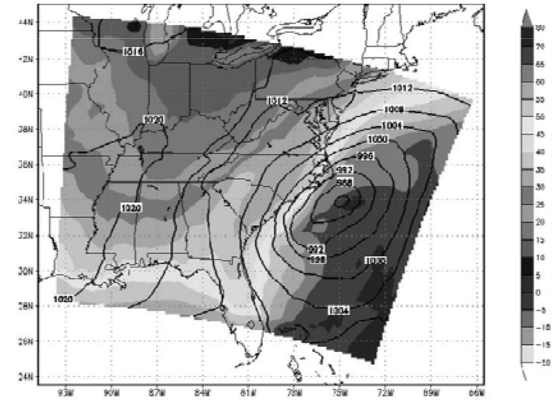
shaded=T ; black=U ; red=V



Staggering

- **Since GrADS version 1.9**
  - a new **gradsnc** interface is available *created by GrADS developers for WRF*
- **To USE**
  - must create 4 .ctl files (M ; U ; V ; W)
  - must open the all at once
- **Utility**
  - ARWpost/util/WRFnc2ctl.f

Creating a Plot



Creating a Plot

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

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

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

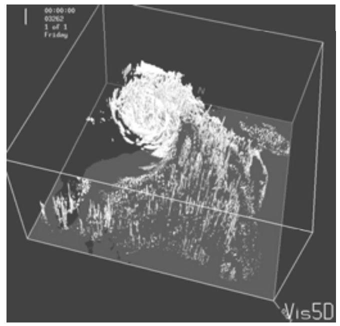
vis5d specific notes

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

  vis5d *output_root_name*.v5d

- **Graphical Interface**

vis5d graphical interface



VAPOR

Visualization and Analysis Platform for Oceanic, atmospheric and solar Research

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

WRF in VAPOR

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

CISL Computational and Information Systems Laboratory
National Center for Atmospheric Research

WRF in VAPOR

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

CISL Computational and Information Systems Laboratory
National Center for Atmospheric Research

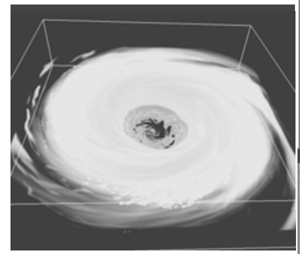
WRF in VAPOR

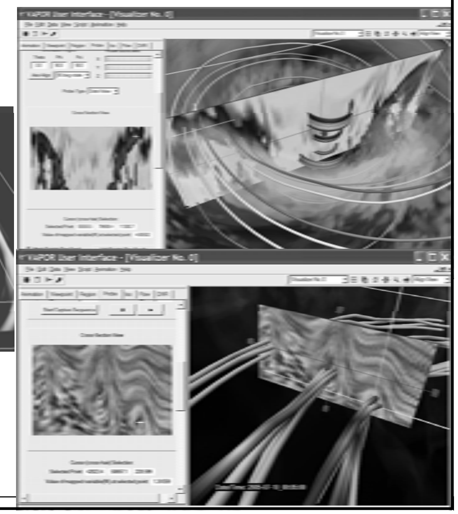
- **Steady and unsteady flow integration**
- **Interactive seed placement**
- **Data probing**
- **Contour plotting**
- **Terrain surface image render**
- **Interactive performance on terabyte datasets**

- **Downloads, documentation, examples at: http://www.vapor.ucar.edu**

- **http://www.vapor.ucar.edu/doc/WRFsupport.pdf**

WRF in VAPOR





CISL Computational and Information Systems Laboratory
National Center for Atmospheric Research

WPS Advanced Usage

Advanced Features of the WRF Preprocessing System

Michael Duda

Outline

- The GEOGRID.TBL file
  - What is the GEOGRID.TBL file?
  - Ingesting new static fields
  - Examples: Using high-resolution land use and topography data
- The METGRID.TBL file
  - What is the METGRID.TBL file?
  - Example: Defining interpolation options for a new field
  - Example: Using the METGRID.TBL file for a real-time system

The GEOGRID.TBL File

- GEOGRID.TBL is the file that determines which fields are interpolated by geogrid *at runtime*
  - Each entry in GEOGRID.TBL corresponds to one data source
  - When new data sources are involved, or when the default treatment of fields is inadequate, user may want/need to edit GEOGRID.TBL
  - However, default GEOGRID.TBL is sufficient to initialize a WRF simulation

The GEOGRID.TBL File

- Format of GEOGRID.TBL file is simple text, with specifications of the form *keyword*=*value*
- Example entry for a 30” landuse data set:

```
===============================
name=LANDUSEF   # Houston, TX urban data
       priority   = 1
       dest_type  = categorical
       z_dim_name = land_cat
       interp_option = 30s:nearest_neighbor
       abs_path = 30s:/users/duda/Houston/
===============================
```

For a complete list of possible keywords: See p. 3-37

The GEOGRID.TBL File

- Using the GEOGRID.TBL, we can
  - Change the method(s) used to interpolate a field
  - Apply smoothing filters to continuous fields
  - Derive fields from others
    - E.g., dominant category or slope fields
  - Add new data for geogrid to interpolate

New Fields in GEOGRID.TBL

There are three basic types of new data to be added through the GEOGRID.TBL file:

1) Completely new fields
   - fields that were previously not processed by geogrid
2) Different resolution data sets for an existing field
   - Such sources *do not need to be supplemented* by existing data
   - E.g., Adding a 90-meter resolution topography data set
3) Alternative sources for a field that *must be used in addition to an existing source*
   - E.g., A new soil category data set exists, but covers only part of South Korea

1) Completely new fields

Completely new fields:

For a new field, simply add an entry in GEOGRID.TBL for that field.

```
============================
name = MY_NEW_FIELD_NAME
   priority = 1
   dest_type  = continuous
   interp_option = four_pt
   abs_path      = /data/duda/mydata/
============================
```

Name of field that this entry is for

Priority of this data source compared with other sources for same field

How to interpolate this field

Where on disk to find the data for this field

See p. 3-37

2) Different resolution data set

Different resolution data sets for an existing field :

Specify the path to the new data set and which interpolation methods should be used for the new resolution in the existing entry for that field.

```
===============================
name = HGT_M
     priority = 1
     dest_type = continuous
     smooth_option = smth-desmth
     interp_option =        30s:special(4.0)+four_pt
     interp_option =      my_res:four_pt
     interp_option =     default:four_pt
     rel_path=      30s:topo_30s/
     rel_path=   my_res:new_topo_directory/
     rel_path=  default:topo_2m/
===============================
```

3) Alternative data sources

Alternative sources for a field that must be used in addition to an existing source :

Add a new entry for the field that has the same name as the field's existing entry, but make priority of new entry higher.

```
===============================
name = HGT_M
      priority = 2
      dest_type = continuous
      interp_option = default:four_pt
      rel_path      = default:some_path/
===============================
name = HGT_M
     priority = 1
     dest_type = continuous
     interp_option = default:four_pt
     rel_path      = default:topo_2m/
===============================
```


Preparing new geogrid data sets

To add a new data source, we need to

1) Write the data in the proper binary format

– See Chapter 3: "Writing Static Data to the Geogrid Binary Format"

– Can make use of read_geogrid.c and write_geogrid.c

2) Create an "index" metadata file for the data set

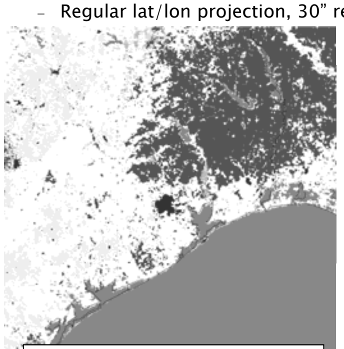
– This tells geogrid about the projection, coverage, resolution, type, and storage representation of the data set

3) Add/edit entry for the data in the GEOGRID.TBL file

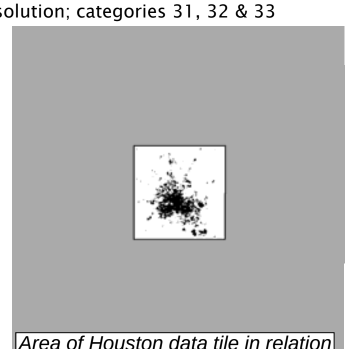
– The change to GEOGRID.TBL will follow one of the three cases mentioned before

Example: Houston LU Data Set

- Given dataset for new Houston urban land use categories
  - Regular lat/lon projection, 30" resolution; categories 31, 32 & 33



Urban areas (black) using USGS 24-category data set



Area of Houston data tile in relation to model domain – white=missing data and blue=valid data

Example: Houston LU Data Set

To make use of the new data, we do the following:

1) Write the data to the binary format used by geogrid

2) Create an index file for the data

```
type=categorical
category_min=31; category_max=33
projection=regular_ll
dx=0.00833333; dy=0.00833333
known_x=1.0;   known_y=1.0
known_lat=29.3375
known_lon=-95.9958333
wordsize=1
tile_x=157; tile_y=143; tile_z=1
missing_value = 0.
units="category"
description="3-category urban LU"
```

Data set has categories 31 through 33 (category_min / category_max)

30 arc second resolution (dx / dy)

Geographic location of data set (known_x, known_y, known_lat, known_lon)

Treat 0 as "no data" (missing_value)

See p. 3-40

Example: Houston LU Data Set

3) Define an entry for the data in GEOGRID.TBL

```
===============================
name=LANDUSEF
        priority   = 2
        dest_type  = categorical
        z_dim_name = land_cat
        interp_option = default:nearest_neighbor
        abs_path  = default:/users/duda/Houston/
===============================
```

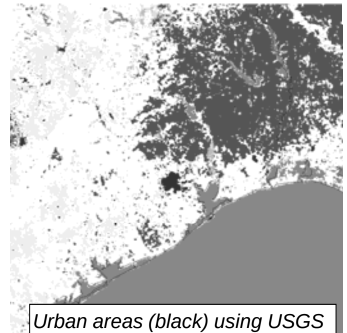
Give this data source priority over default data sources

How to interpolate this data source, and where to find it on disk

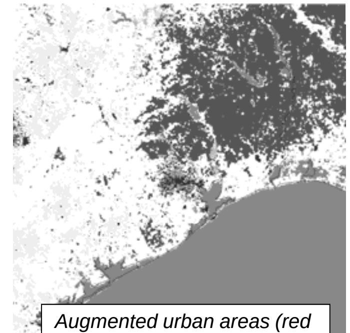
Example: Houston LU Data Set

4) Run geogrid.exe

Any gridpoints covered by Houston data will use it; otherwise default USGS data will be used



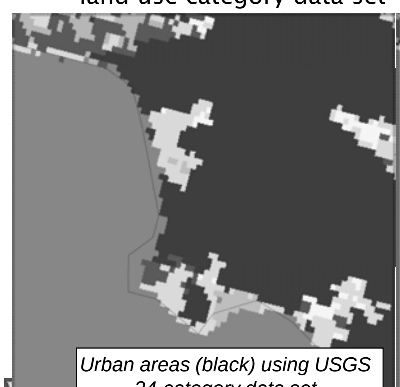
Urban areas (black) using USGS 24-category data set



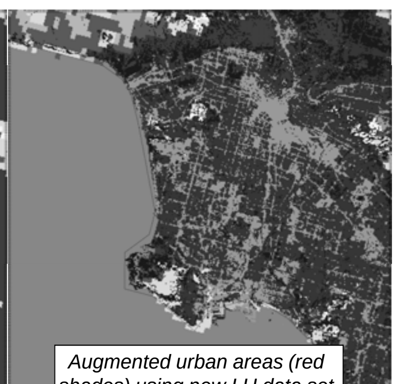
Augmented urban areas (red shades) using new LU data set

Another Example: Los Angeles

For Los Angeles, we have a 30-meter resolution, 3 urban land use category data set



Urban areas (black) using USGS 24-category data set



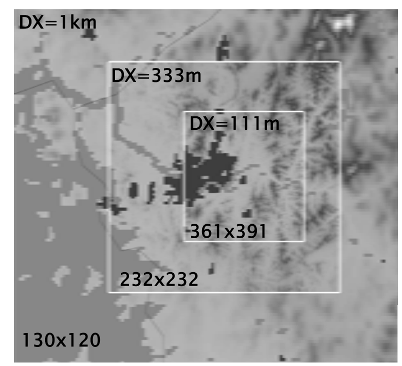
Augmented urban areas (red shades) using new LU data set

Example: South Korea

Shuttle Radar Topography Mission (SRTM) 3 arc second topography data

We would like to use the SRTM data, especially for domains 2 and 3.

Follow steps for adding a new resolution for an existing data set (case 2)



Example: Seoul

To use the SRTM topography data, we

1) Write data to geogrid binary format

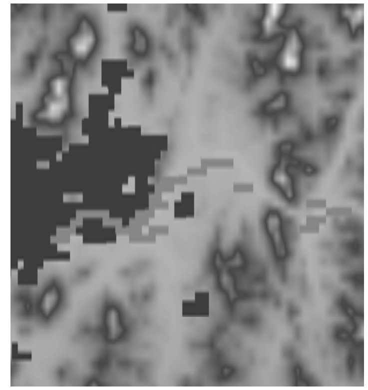
2) Create an index file for the data set

3) Modify the GEOGRID.TBL entries for HGT_M, HGT_U, and HGT_V

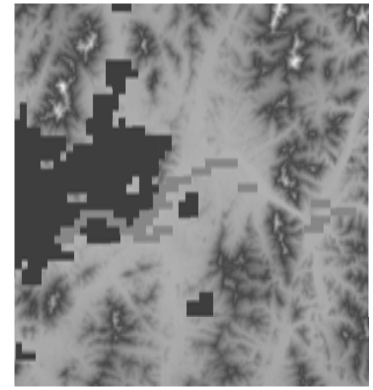
```
===============================
name = HGT_M
     priority = 1
     dest_type = continuous
     interp_option  =  30s:special(4.0)+four_pt
     interp_option  =  SRTM:four_pt
     rel_path    =  30s:topo_30s/
     rel_path    =  SRTM:SRTM/
===============================
```

4) Specify that we should interpolate from SRTM in namelist by setting
`geog_data_res = '30s','SRTM+30s','SRTM+30s'`

Example: Seoul



Domain 3 (DX=111m) using default 30" USGS topography



Domain 3 (DX=111m) using 3" SRTM topography

Outline

- The GEOGRID.TBL file
  - What is the GEOGRID.TBL file?
  - Ingesting new static fields
  - Example: Houston urban data
- The METGRID.TBL file
  - What is the METGRID.TBL file?
  - Example: Building a METGRID.TBL entry for a new field
  - Example: Using the METGRID.TBL file for real-time runs

The METGRID.TBL File

The METGRID.TBL file controls how meteorological fields are interpolated

- Unlike GEOGRID.TBL, METGRID.TBL *does not determine which fields will be processed*, only *how to process them* if they are encountered
- Every field in intermediate files will be interpolated
  - If no entry in METGRID.TBL for a field, a default interpolation scheme (nearest neighbor) will be used
  - It is possible to specify in METGRID.TBL that a field should be discarded

The METGRID.TBL File

- Suitable entries in METGRID.TBL are provided for common fields
  - *Thus, many users will rarely need to edit METGRID.TBL*
- When necessary, different interpolation methods (and other options) can be set in METGRID.TBL
  - Interpolation options can depend on the source of a field

The METGRID.TBL File

- Example METGRID.TBL entry (for "soil moisture 0-10 cm")

```
========================================
name=SM000010
interp_option=sixteen_pt+four_pt+average_4pt
masked=water
interp_mask=LANDSEA(0)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```


Example: A new METGRID.TBL entry

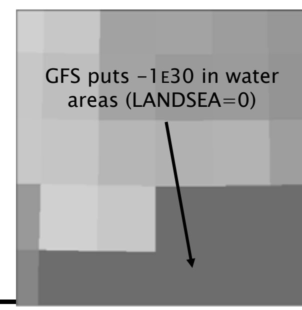
- Suppose we have a 1000x1000 domain over Houston (dx=500 m)
  - This is the same domain as in the urban land use example
- Meteorological data come from 1-degree GFS
  - *Note that we will be interpolating 1-degree data onto a 500-m grid!*
- We want to create an entry for a new soil moisture field, SM000010

Example: A new METGRID.TBL entry

- Initially, we run metgrid.exe and get the message:

```
INFORM: Entry in METGRID.TBL not found for field SM000010.
  Default options will be used for this field!
```

- The resulting SM000010 field looks very coarse
- We need to create a METGRID.TBL entry so metgrid will know how to interpolate this field!



Example: A new METGRID.TBL entry

- We add an initial entry in METGRID.TBL for SM000010:

```
===========================
name = SM000010
masked = water
interp_mask = LANDSEA(0)
interp_option = sixteen_pt + nearest_neighbor
fill_missing = 0.
===========================
```

Specify that the field should *not* be interpolated to model water points (masked = water)

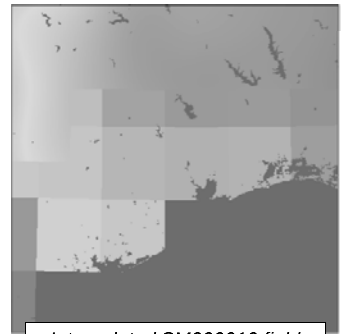
Specify that metgrid should not use points in source where LANDSEA field equals 0 (interp_mask = LANDSEA(0))

Fill model points that don't receive an interpolated value (like water) to 0 (fill_missing = 0.)

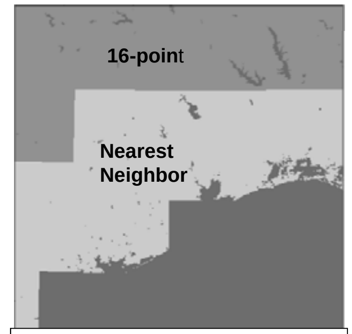
For a complete list of possible keywords: See p. 3-42

Example: A new METGRID.TBL entry

- Now, after running metgrid.exe again, the SM000010 field looks like



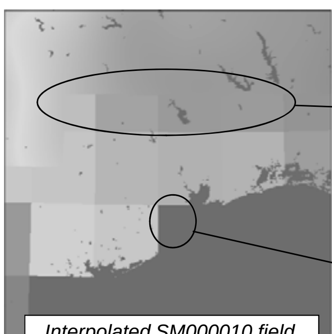
Interpolated SM000010 field (sixteen_pt + nearest_neighbor)



Which interpolator was used at each model grid point

Example: A new METGRID.TBL entry

- But, the interpolated field still looks bad near the coastline



Interpolated SM000010 field (sixteen_pt + nearest_neighbor)

Should be sufficient data to use 4-point interpolation in these areas

Model grid points here should be adjacent to at least one valid GFS point (though not nearest)

Example: A new METGRID.TBL entry

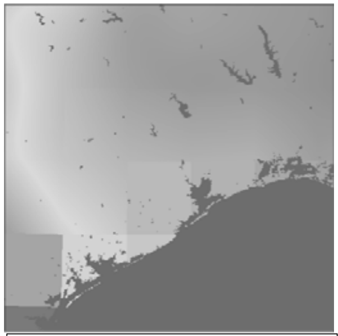
- Update the METGRID.TBL entry for SM000010

```
===========================
name = SM000010
masked = water
interp_mask = LANDSEA(0)
interp_option = sixteen_pt + four_pt + average_4pt
fill_missing = 0.
===========================
```

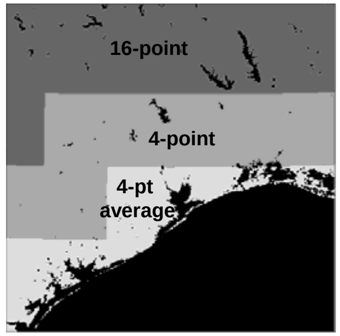
- If 16-pt doesn't work, then try 4-pt before reverting to a 4-point average
  - Note that 4-point average will work anywhere nearest_neighbor would (missing/masked values not counted in the average)

Example: A new METGRID.TBL entry

- The resulting field, below-left:



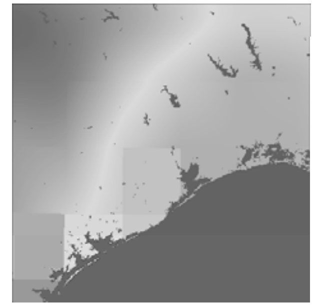
Interpolated SM000010 field (sixteen_pt + four_pt + average_4pt)



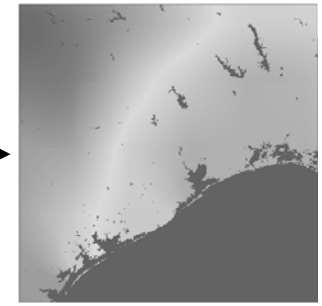
Which interpolator was used at each model grid point

Example: A new METGRID.TBL entry

- By using **`wt_average_4pt`** instead of **`average_4pt`**:



sixteen_pt + four_pt + average_4pt



sixteen_pt + four_pt + wt_average_4pt

METGRID.TBL: Real-time System Example

- Suppose we have a real-time system that:
  - Uses GFS for initial and boundary conditions
  - When possible (i.e., if the files are available soon enough) uses *soil moisture* and *soil temperature* fields from AGRMET
- In our system, it may occasionally happen that the AGRMET files are not ready when we want to start our WRF run
  - Because system is real-time, we want to proceed using just the GFS land surface fields!

METGRID.TBL: Real-time System Example

- We already know how to run ungrib on multiple sources of data to get

  GFS:YYYY-MM-DD_HH

  and

  AGRMET:YYYY-MM-DD_HH

  intermediate files, and specify

  fg_name = 'GFS', 'AGRMET',

  in the `&metgrid` namelist record to use both sources

See p. 3-17

METGRID.TBL: Real-time System Example

Without further changes, what happens if:

Only GFS data are available when we run metgrid

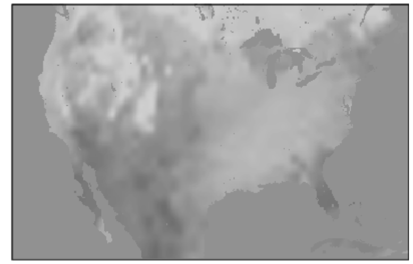
Metgrid runs and warns that no AGRMET data files were found:

```
Processing 2006-04-01_00
     GFS
     AGRMET
 WARNING: Couldn't open file AGRMET:2006-04-01_00 for
 input.
```

Metgrid will finish, but will only use GFS data!

METGRID.TBL: Real-time System Example

And the 0-10 cm soil moisture field (SM000010) looks like:

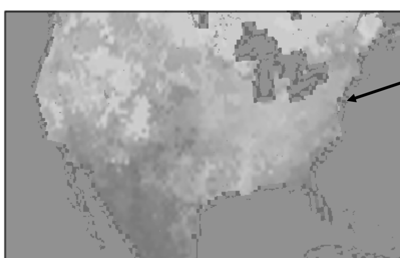


METGRID.TBL: Real-time System Example

However, what happens if:

Both GFS and AGRMET files are available when we run metgrid?

Our SM000010 field looks like:



We get unreasonable values with magnitude ~1E30 near land-water boundaries!

METGRID.TBL: Real-time System Example

Why are there bad values near coastlines? What went wrong?

In both Vtable.GFS and Vtable.AGRMET, the land-sea mask field is named LANDSEA

– In METGRID.TBL, our entry for SM000010 says:

```
========================================
name=SM000010
interp_option=sixteen_pt+four_pt+wt_average_4pt+search
masked=water
interp_mask=LANDSEA(0)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

METGRID.TBL: Real-time System Example

```
========================================
name=SM000010
interp_option=sixteen_pt+four_pt+wt_average_4pt+search
masked=water
interp_mask=LANDSEA(0)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

After metgrid reads in LANDSEA from GFS file *to use as an interpolation mask*, it ignored the LANDSEA field from AGRMET *for use as a mask*.

– So, metgrid used the GFS LANDSEA mask even when interpolating AGRMET data!

METGRID.TBL: Real-time System Example

When metgrid interpolated SM000010, it used the GFS landmask for a field masked by the AGRMET landmask!

GFS LANDSEA field

AGRMET LANDSEA field

Note the disagreement between the two data sources near coastlines.

METGRID.TBL: Real-time System Example

Solution:

- Rename LANDSEA to *AGR_LAND* in Vtable.AGRMET
- Rename LANDSEA to *GFS_LAND* in Vtable.GFS
- Create separate entries in METGRID.TBL
  - one for GFS SM000010 field
  - another for AGRMET SM000010 field

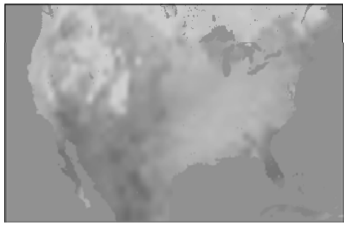
METGRID.TBL: Real-time System Example

```
========================================
name=SM000010; from_input=GFS
interp_option=sixteen_pt+four_pt+wt_average_4pt+search
masked=water
interp_mask=GFS_LAND(0)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

```
========================================
name=SM000010; from_input=AGRMET
interp_option=sixteen_pt+four_pt+wt_average_4pt+search
masked=water
interp_mask=AGR_LAND(-1.E30)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

METGRID.TBL: Real-time System Example

With modified Vtables and METGRID.TBL:



The SM000010 field when only GFS files are available

The SM000010 field when both GFS and AGRMET files are available

Summary

- In this lecture, we've seen
  - What the GEOGRID.TBL and METGRID.TBL files do
  - How to use new geographical data sources in the WPS
    - High-resolution land use and topography data
  - How to use the METGRID.TBL file to correct two types of interpolation-related problems
- For more other features of the WPS, see Chapter 3 of the User's Guide

ARW Idealized Cases

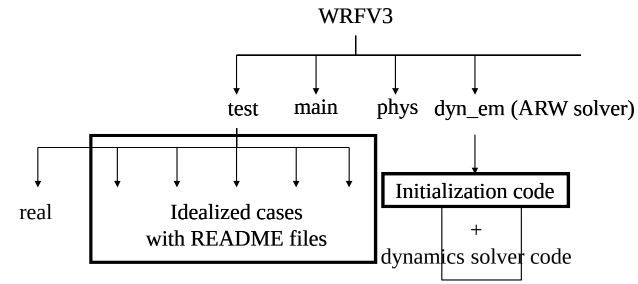
Initialization for Idealized Cases

Bill Skamarock and Wei Wang

Why do we provide idealized cases?

1. The cases provide simple tests of the dynamics solver for a broad
   range of space and time scale:
   LES - Δx meters, Δt < second;
   Baroclinic waves - Δx 100 km, Δt = 30 minutes.
2. The test cases reproduce known solutions
   (analytic, converged, or otherwise).
3. The cases provide a starting point for other idealized experiments.
4. Could be used to test physics development.
5. Easy starting point to test the code on your computer.

WRF ARW code



Test Cases for the WRF ARW Model

- 2D flow over a bell-shaped mountain
  WRFV3/test/em_hill2d_x
- 2D squall line (x, z ; y, z)
  WRFV3/test/em_squall2d_x, em_squall2d_y
- 3D quarter-circle shear supercell thunderstorm
  WRFV3/test/em_quarter_ss
- 3D baroclinic wave
  WRFV3/test/em_b_wave
- 2D gravity current
  WRFV3/test/em_grav2d_x
- 3D large-eddy simulation case
  WRFV3/test/em_les
- 3D global: Held-Suarez case
  WRFV3/test/em_heldsuarez
- 2D sea-breeze case
  WRFV3/test/em_seabreeze2d_x

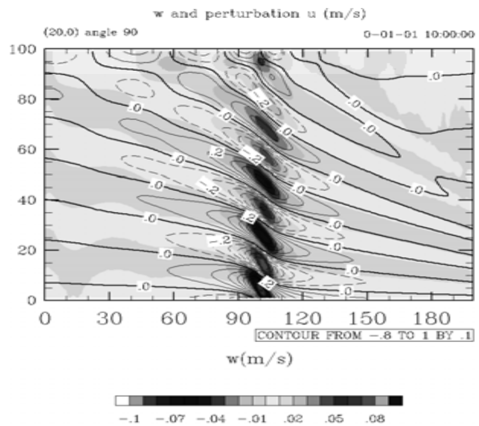
(last three cases: new in V3)

Initialization Tasks

- Read in a single sounding (h, θ, Q_v, u and v) or pre-computed, balanced 2D profile (in the case of b_wave)
- Compute full pressure and (inverse) air density from input sounding
- Compute thermodynamic reference state, based on the sounding without moisture (dry pressure, dry inverse air density)
- Compute dry column pressure μ_d, and then model η levels
- Interpolating θ to η levels, compute inverse air density, and then geopotential - μ_d, θ, and geopotential are in exact hydrostatic balance
- Interpolating other fieds to model η levels
- Model levels are set automatically; they can be stretched in η (close to equally spaced z), or equally spaced in η

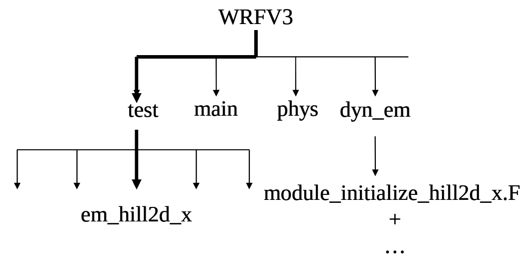
2D Flow Over a Bell-Shaped Mountain

(dx = 2 km, dt = 20 s, T=10 hr)



2D Flow Over a Bell-Shaped Mountain

Initialization module: dyn_em/module_initialize_hill2d_x.F
Case directory: test/em_hill2d_x



2D Flow Over a Bell-Shaped Mountain

Initialization code is in
WRFV3/dyn_em/module_initialize_hill2d_x.F
The terrain profile is set in the initialization code.

The thermodynamic sounding and the initial wind field is read from the ascii file
WRFV3/test/em_hill2d_x/input_sounding

The 2D solution is computed by integrating the 3D model with 3 points in periodic direction y; without an initial perturbation in y the solution remains y-independent.

Setting the terrain heights

In *WRFV3/dyn_em/module_initialize_hill2d_x.F*

```
 SUBROUTINE init_domain_rk ( grid, &

...
 hm = 100.          <---- mountain height and half-width
 xa = 5.0

 icm = ide/2        <---- mountain position in domain
...                       (center gridpoint in x)

Set height
field          DO j=jts,jte
               DO i=its,ite  ! flat surface
          ----->  grid%ht(i,j) = hm/(1.+(float(i-icm)/xa)**2)
                  grid%phb(i,1,j) = g*grid%ht(i,j)
                  grid%php(i,1,j) = 0.  <---- lower boundary condition
                  grid%ph0(i,1,j) = grid%phb(i,1,j)
               ENDDO
               ENDDO
```

Setting the Initial Condition

In *WRFV3/dyn_em/module_initialize_hill2d_x.F*

```
   SUBROUTINE init_domain_rk ( grid, &

...
!  get the sounding from the ascii sounding file, first get dry sounding and
!  calculate base state

  write(6,*) ' getting dry sounding for base state '
  dry_sounding = .true.
  CALL get_sounding( zk, p_in, pd_in, theta, rho, u, v, qv, dry_sounding, &
                     nl_max, nl_in, .true.)
...

!  calculate full state for each column - this includes moisture.

  write(6,*) ' getting moist sounding for full state '
  dry_sounding = .false.
  CALL get_sounding( zk, p_in, pd_in, theta, rho, u, v, qv, dry_sounding, &
                     nl_max, nl_in, .false. )
...
```

Base state
Dry sounding (points to: dry_sounding = .true.)

Full state
Moist sounding (points to: dry_sounding = .false.)

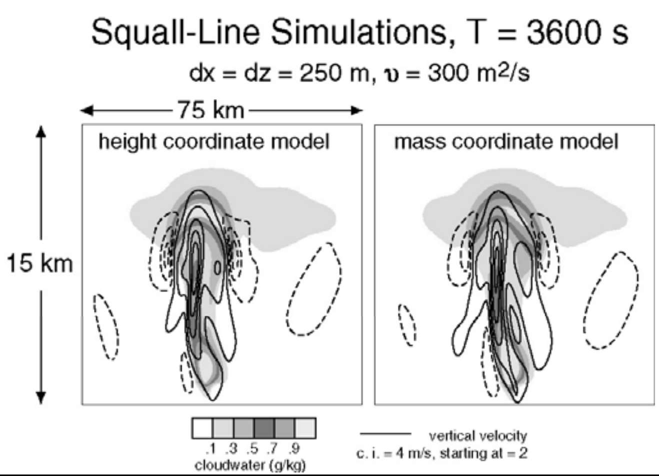
Sounding File Format

File: *WRFV3/test/em_quarter_ss/input_sounding*

Line 1: surface Pressure (mb), surface potential Temperature (K), Surface vapor mixing ratio (g/kg)

line 1	1000.00	300.00	14.00		
each successive line is a point in the sounding	250.00	300.45	14.00	-7.88	-3.58
	750.00	301.25	14.00	-6.94	-0.89
	1250.00	302.47	13.50	-5.17	1.33
	1750.00	303.93	11.10	-2.76	2.84
	2250.00	305.31	9.06	0.01	3.47
	2750.00	306.81	7.36	2.87	3.49
	3250.00	308.46	5.95	5.73	3.49
	3750.00	310.03	4.78	8.58	3.49
	4250.00	311.74	3.82	11.44	3.49
	4750.00	313.48	3.01	14.30	3.49
	height (m)	potential temperature (K)	vapor mixing ratio (g/kg)	U (west-east) velocity (m/s)	V (south-north) velocity (m/s)

2D squall line simulation



2D squall line simulation

squall2d_x is (x,z), *squall2d_y* is (y,z); both produce the same solution.

Initialization codes are in
WRFV3/dyn_em/module_initialize_squall2d_x.F
WRFV3/dyn_em/module_initialize_squall2d_y.F
This code also introduces the initial perturbation.

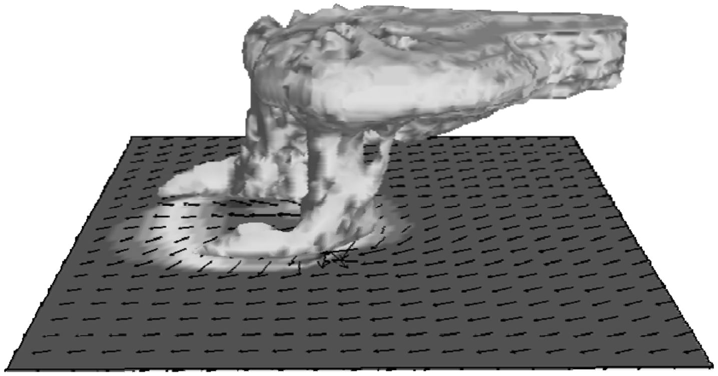
The thermodynamic soundings and hodographs are in the ascii input files
WRFV3/test/em_squall2d_x/input_sounding
WRFV3/test/em_squall2d_y/input_sounding

3D supercell simulation

Height coordinate model

(dx = dy = 2 km, dz = 500 m, dt = 12 s, 160 x 160 x 20 km domain)

Surface temperature, surface winds and cloud field at 2 hours



3D supercell simulation

Initialization code is in
WRFV3/dyn_em/module_initialize_quarter_ss.F

The thermodynamic sounding and hodograph is read from the ascii input file
WRFV3/test/em_quarter_ss/input_sounding

The initial perturbation (warm bubble) is hardwired in the initialization code.

Setting the initial perturbation

In *WRFV3/dyn_em/module_initialize_quarter_ss.F*

```
SUBROUTINE init_domain_rk ( grid, &
...
!  thermal perturbation to kick off convection
...
  DO J = jts, min(jde-1,jte)
    yrad = dy*float(j-nyc)/10000.
!   yrad = 0.
    DO I = its, min(ide-1,ite)
      xrad = dx*float(i-nxc)/10000.
!     xrad = 0.
      DO K = 1, kte-1

!  put in preturbation theta (bubble) and recalc density.  note,
!  the mass in the column is not changing, so when theta changes,
!  we recompute density and geopotential

        zrad = 0.5*(ph_1(i,k,j)+ph_1(i,k+1,j)  &
                   +phb(i,k,j)+phb(i,k+1,j))/g
        zrad = (zrad-1500.)/1500.
        RAD=SQRT(xrad*xrad+yrad*yrad+zrad*zrad)
        IF(RAD <= 1.) THEN
           grid%t_1(i,k,j)=T_1(i,k,j)+delt*COS(.5*PI*RAD)**2
           grid%t_2(i,k,j)=T_1(i,k,j)
           qvf = 1. + 1.61*moist_1(i,k,j,P_QV)
           grid%alt(i,k,j) = (r_d/p1000mb)*(t_1(i,k,j)+t0)*qvf* &
                        (((p(i,k,j)+pb(i,k,j))/p1000mb)**cvpm)
           grid%al(i,k,j) = alt(i,k,j) - alb(i,k,j)
        ENDIF
      ENDDO
```

horizontal radius of the perturbation is 10 km, centered at (x,y) gridpoints (nxc, nyc)

vertical radius of the perturbation is 1500 m

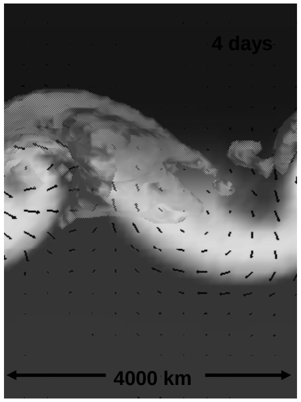
perturbation added to initial theta field

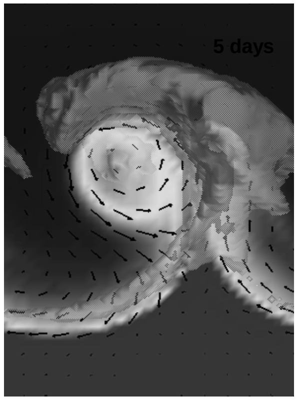
maximum amplitude of the perturbation

Moist Baroclinic Wave Simulation

Height coordinate model (dx = 100 km, dz = 250 m, dt = 600 s)

Surface temperature, surface winds, cloud and rain water





Moist Baroclinic Wave Simulation

Initialization code is in
WRFV3/dyn_em/module_initialize_b_wave.F

The initial jet (y,z) is read from the binary input file
WRFV3/test/em_b_wave/input_jet

The initial perturbation is hardwired in the initialization code.

Moist Baroclinic Wave Simulation

Default configuration in
WRFV3/test/em_b_wave/namelist.input
runs the dry jet in a periodic channel with dimension (4000 x 8000 x 16 km) (x,y,z).

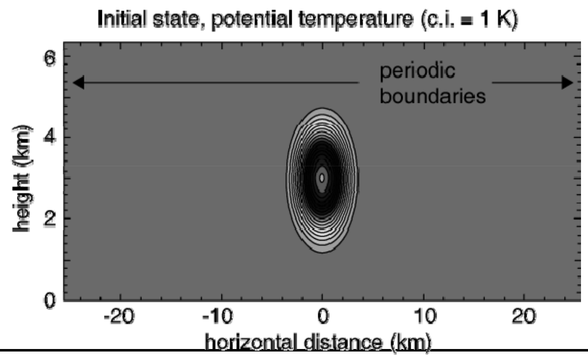
Turning on any microphysics
(mp_physics > 0 in namelist.input) puts moisture into the model state.

The initial jet only works for dy = 100 km and
81 grid points in the y (south-north) direction.

Gravity Current Simulation

(Straka et al, IJNMF, 1993)

2D channel (x , z ; 51.2 x 6.4 km)
Initial state: theta = 300 K (neutral) + perturbation (max = 16.2 K)
Eddy viscosity = 75 m**2/s**2 (constant)

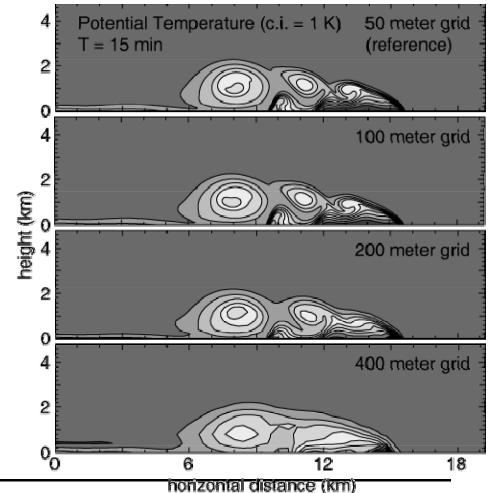


Gravity Current Simulation

Default case, dx = 100 m,
5th order upwind advection,
uses namelist.input.100m

dx = 200 m,
5th order upwind advection,
use namelist.input.200m

dx = 400 m,
5th order upwind advection,
use namelist.input.400m

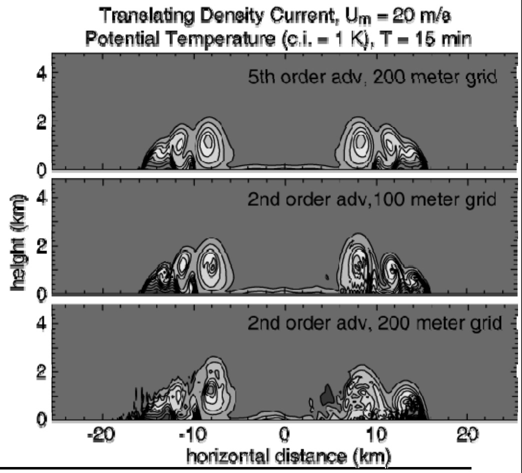


Gravity Current Simulation

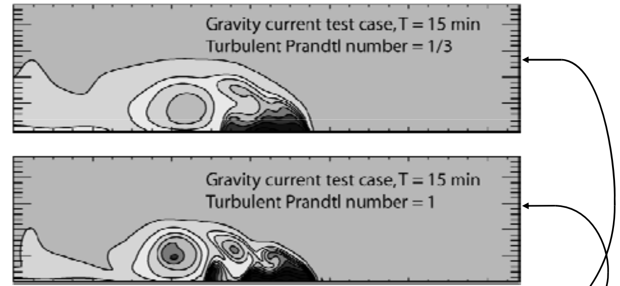
5th order upwind advection, use namelist.input.200m and input_sounding.um=20

use namelist.input.100m with 2nd order advection and input_sounding.um=20

use namelist.input.200m with 2nd order advection and input_sounding.um=20



Gravity Current Simulation



The turbulent Prandtl number in WRF 1/3,
So the default WRF test case will give this solution.

To recover the Straka et al (1993) solution, change the parameter *Prandtl* to 1 (from 1/3) in
WRFV3/share/module_model_constants.F
WRFV3/share/module_diffusion_em.F, module_big_step_utilities.F

Gravity Current Simulation

Initialization code is in
WRFV3/dyn_em/module_initialize_grav2d_x.F

The initial cold bubble is hardwired in the initialization code.

Held-Suarez Case

Initialization code is in
WRFV3/dyn_em/module_initialize_heldsuarez.F

The initial model state is an isothermal atmosphere on flat earth with no winds, and random temperature perturbation

Test case directory is in
WRFV3/test/em_heldsuarez

Large-Eddy Simulation Case

Initialization code is in
WRFV3/dyn_em/module_initialize_les.F

Test case directory is in
WRFV3/test/em_les

The default case is a large-eddy simulation of free convective boundary layer with no winds. The turbulence of the free CBL is driven and maintained by namelist-specified surface heat flux.

An initial sounding with mean winds is also provided.

Reference: Moeng et al. 2007 MWR

Large-Eddy Simulation Case

QuickTime™ and a decompressor are needed to see this picture.

QuickTime™ and a decompressor are needed to see this picture.

QuickTime™ and a decompressor are needed to see this picture.

January 2009 Mesoscale & Microscale Meteorology Division, NCAR

2D Sea-Breeze Simulation Case

Initialization code is in
WRFV3/dyn_em/module_initialize_seabreeze2d_x.F

Test case directory is in
WRFV3/test/em_seabreeze2d_x

The initial state has no wind, and is perturbed by small random temperature changes

An example to show how to set surface parameters so that one may use full surface physics

January 2009 Mesoscale & Microscale Meteorology Division, NCAR

More on Idealized Cases ..

Descriptions:
WRFV3/README_test_cases
WRFV3/test/em_*/README

ARW Tech Note

Publications

January 2009 Mesoscale & Microscale Meteorology Division, NCAR

Objective Analysis:

OBSGRID

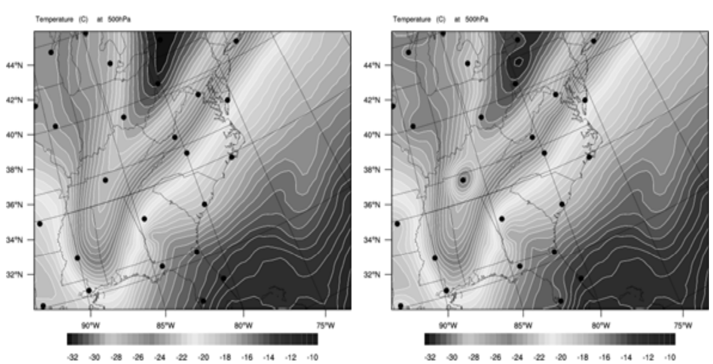
Objective Analysis (OBSGRID)

Cindy Bruyère

Objective Analysis

- **To improve a *first-guess* gridded analysis by incorporating additional observational information**
  - Traditionally, this first-guess analysis comes from low-resolution global analysis and forecast grids
  - These days, higher-resolution, regional scale analyses are more readily available
    - *These high-resolution analyses mean that in many cases, the objective analysis step is not essential when running WRF*

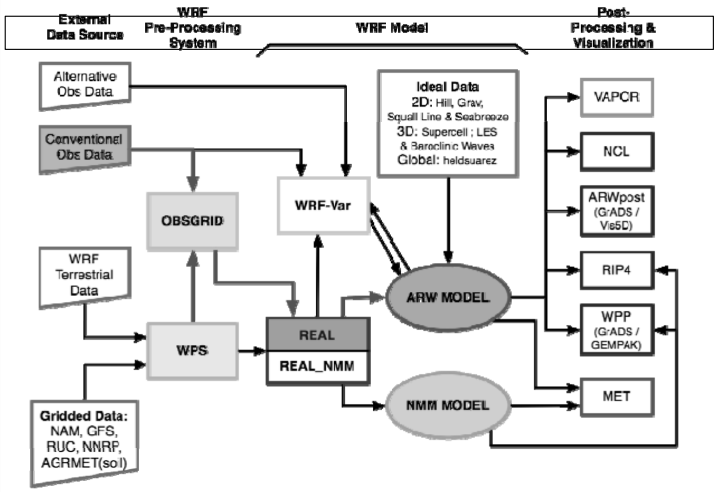
Objective Analysis



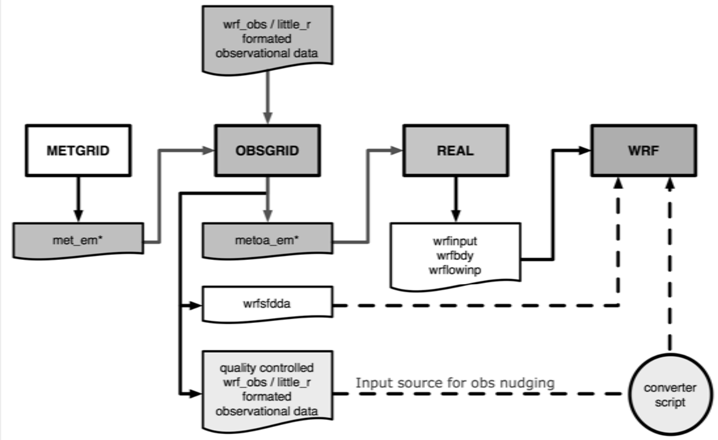
Objective Analysis in WRF

- **Traditional methods**
  - Direct observations of T, U, V, RH at surface and on pressure levels *(conventional observations)*
- **Variational Analysis**
  - Direct and indirect observations on model levels *(conventional + alternative observations)*

WRF Modeling System Flow Chart



OBSGRID Flow Chart



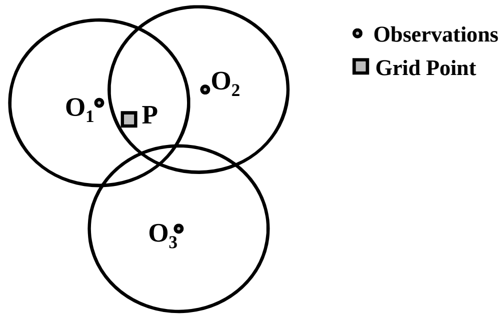
Surface FDDA Option

- **Creates a separate surface analysis file for later use by the WRF Surface FDDA Grid Nudging option**
- **Surface analyses usually created more frequently than upper-air analyses**
- ***WRF Surface FDDA not yet available - planned release March 2009***

OA Techniques in OBSGRID

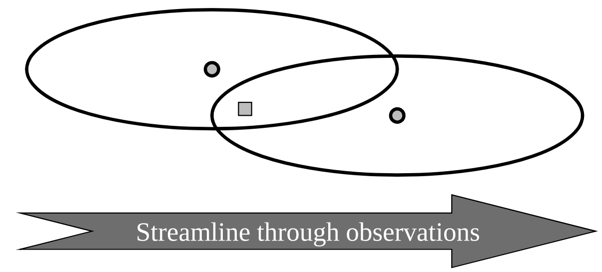
- **Cressman-based analysis**
  - Impact of observation within radius of influence only
  - Multiple scans
  - With ellipse and banana extensions
- **Multi-Quadric analysis**
  - Impact of observations are over the entire model domain
  - Scheme is sensitive to the data density distribution

Cressman Scheme

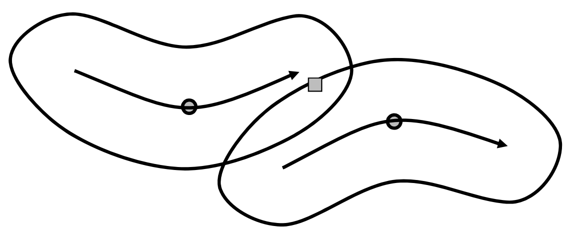


Observations O_1 and O_2 influences grid point P, O_3 does not

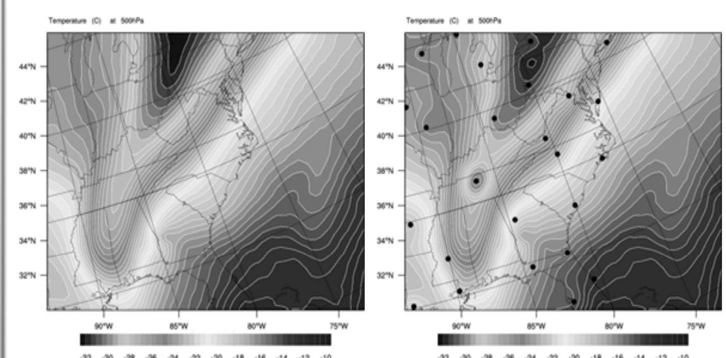
Ellipse Scheme



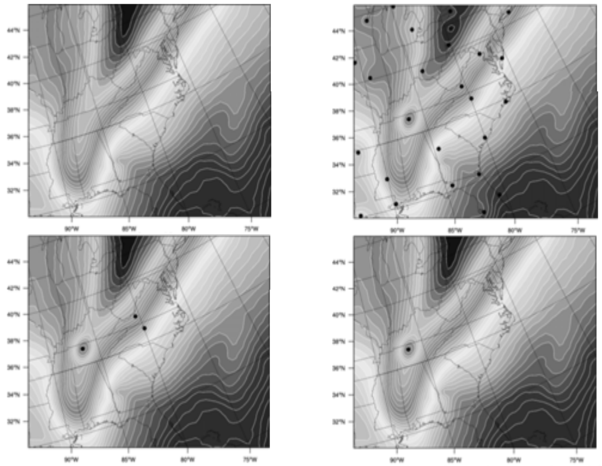
Banana Scheme



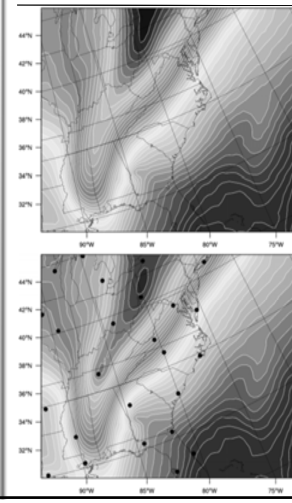
Cressman Scheme



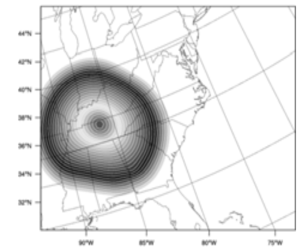
Cressman Scheme



Multi-Quadric Scheme

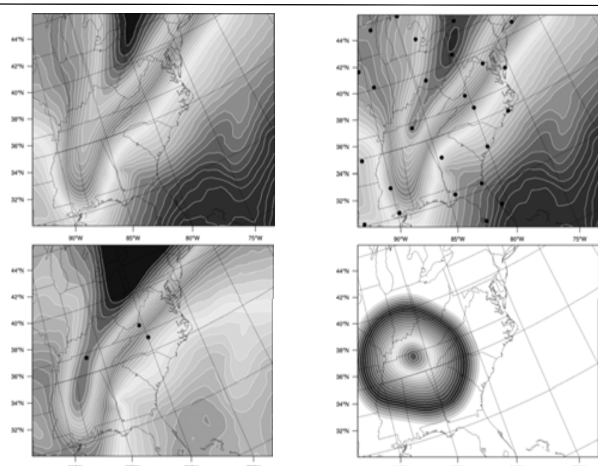


Makes use of hyperboloid radial basis functions to interpolate observation corrections onto model grid



Mesoscale & Mircoscale Meteorology Division / NCAR

Multi-Quadric Scheme



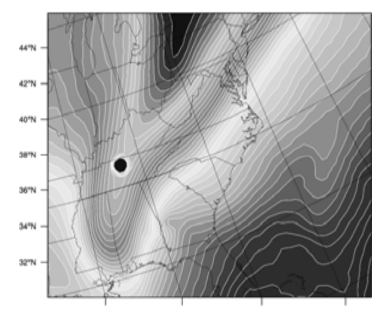
Mesoscale & Mircoscale Meteorology Division / NCAR

Quality Control for Observations

- **A critically important step**

  Bad observations = Bad objective analysis

- **Even a single bad observation can ruin initial conditions**



Mesoscale & Mircoscale Meteorology Division / NCAR

Quality Control for Observations

- Tests on individual reports
- ERRMAX test
- Buddy test

Tests on individual reports

- Gross error checks
- Remove spikes from temperature and wind profiles *(optional, not recommended)*
- Adjust temperature profiles to remove superadiabatic layers *(optional, not recommended)*
- No comparisons to other soundings or to first-guess field

Mesoscale & Mircoscale Meteorology Division / NCAR

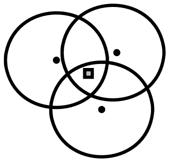
ERRMAX test

- Limited user control over data removal
- Observations are compared to first-guess field
- If the difference between the observation and the first-guess exceeds a threshold, the observation is discarded
- Threshold varies depending on field, level, time of day
- Works well with good first-guess field

Mesoscale & Mircoscale Meteorology Division / NCAR

Buddy test

- Limited user control over data removal
- Observations are compared to the first guess and to nearby observations
- If an observation deviates from the first guess in a manner inconsistent with the deviations of surrounding stations from the first guess, then that observation is discarded
- Works well in regions of good data density



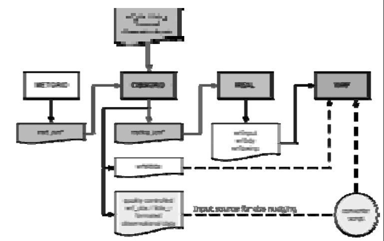
Mesoscale & Mircoscale Meteorology Division / NCAR

How to run OBSGRID

- **Get the source code**
- **Compile *(./configure & ./compile)***
- **Prepare observations files**
  - Users need to generate this file *(some sample programs are available)*
- **Edit the namelist**
- **Run the program**
- **Check your output**

Observations

- **ASCII text files *(wrf_obs / little_r format)***
  - One entry per observation *(sfc or upper-air)*
    - Header ; Data ; End
- **Each time period is stored in a separate file**
- **OBSGRID combines reports, removes duplicates, interpolates to analysis levels**



Observations Format: *Header*

latitude	F20.5	Station latitude
longitude	F20.5	Station longitude
id	A40	Station ID
name	A40	Station name
platform	A40	Measurement device
source	A40	Source of observations
elevation	F20.5	Station elevation (m)
num_vld_fld	I10	Number of valid fields
num_error	I10	Number of errors in decoding
num_warning	I10	Number of warnings in decoding

Observations Format: *Header*

seq_num	I10	Sequence number of this report
num_dups	I10	Number of duplicates found for this report
is_sound	L10	Multiple or single levels
bogus	L10	Bogus or normal report
discard	L10	Duplicate and Discarded report
sut	I10	Time of report (s since 1970-01-01)
julian	I10	Day of the year of the report
date_char	A20	Report time (YYYYMMDDHHmmss)
slp, qc	F13.5, I7	SLP Value and QC flag
ref_pres, qc	F13.5, I7	Reference pressure and QC flag

Observations Format: *Header*

ground_t, qc	F13.5, I7	Ground T and QC flag
sst, qc	F13.5, I7	SST and QC flag
psfc, qc	F13.5, I7	Surface P and QC flag
precip, qc	F13.5, I7	Accumulated Precip and QC flag
t_max, qc	F13.5, I7	Daily maximum T and QC flag
t_min, qqc	F13.5, I7	Daily minimum T and QC flag
t_min_night, qc	F13.5, I7	Overnight min T and QC flag
p_tend03, qc	F13.5, I7	3-hr pressure tendency and QC
p_tend24, qc	F13.5, I7	24-r pressure tendency and QC
cloud_cvr, qc	F13.5, I7	Cloud cover (oktas) and QC flag
ceiling, qc	F13.5, I7	Height of cloud base and QC flag

Observations Format: *Data*

pressure, qc	F13.5, I7	Pressure
height, qc	F13.5, I7	Height
temperature, qc	F13.5, I7	Temperature
dew_point, qc	F13.5, I7	Dewpoint
speed, qc	F13.5, I7	Wind speed
direction, qc	F13.5, I7	Wind direction
u, qc	F13.5, I7	U-component of wind
v, qc	F13.5, I7	V-component of wind
rh, qc	F13.5, I7	Relative Humidity
thickness, qc	F13.5, I7	Thickness

Observations Format: *End*

num_vld_fld	I7	Number of valid fields
num_error	I7	Errors encountered in decoding
num_warning	I7	Warnings encountered in decoding

Quality-Control Flags

- Binary flags indicating which warning and error conditions have been met

Pressure interpolated from first-guess height	2**1	2
Temperature and dewpoint both 0	2**4	16
Wind speed and direction both 0	2**5	32
Wind speed negative	2**6	64
Wind direction < 0 or > 360	2**7	128
Level vertically interpolated	2**8	256
Value vertically extrapolated from a single level	2**9	512
Sign of temperature reversed	2**10	1012
Superadiabatic level detected	2**11	2048
Vertical spike in wind speed or direction	2**12	4096
Convective adjustment applied to temperature field	2**13	8192
No neighboring observations for buddy check	2**14	16384
Error maximum test failed	2**15	32768
Buddy test failed	2**16	65536
Observation outside domain	2**17	131072

Namelist: **&record1**

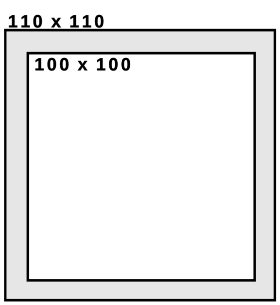
start_year	Four-digit starting year
start_month	Two-digit starting month (01-12)
start_day	Two-digit starting day (01-31)
start_hour	Two-digit starting hour (00-23)
end_year	Ending year
end_month	Ending month
end_day	Ending day
end_hour	Ending hour
interval	Time interval (s) to process

Namelist: **&record2**

domain_id	ID of domain to process
obs_filename	One or more file names of the observation files; one file required for each time period
sfc_obs_filename	One or more file names of the surface fdda observation files; one file required for each surface analysis time period. Used only if F4D=.TRUE.
trim_domain	Set to .TRUE. if this domain must be cut down on output
trim_value	Value by which the domain will be cut down in each direction

Mesoscale & Mircoscale Meteorology Division / NCAR

Trim output domain



- **Why do this:**
This allows observations just outside the desired grid box of 100x100 grid points to be included in the OA

- ***geogrid* and *metgrid* run with a domain size of 110x110**

- **trim_domain = .TRUE.**
- **trim_value = 5**

Mesoscale & Mircoscale Meteorology Division / NCAR

Namelist: **&record3**

max_number_of_obs	Maximum number of observations to be processed in OBSGRID
fatal_if_exceed_max_obs	T/F flag to stop the program if more observations are found

Mesoscale & Mircoscale Meteorology Division / NCAR

Namelist: **&record4**

qc_test_error_max	Turn on error-max test (T/F)
qc_test_buddy	Turn on buddy test (T/F)
qc_test_vert_consistency	Turn on vertical tests (T/F)
qc_test_convective_adj	Remove superadiabatic (T/F)
max_error_t	Max T difference (K)
max_error_uv	Max u or v difference (m/s)
max_error_rh	Max RH difference (%)
max_error_p	Max SLP difference (Pa)
max_buddy_t	Threshold for T buddy check
max_buddy_uv	Threshold for u/v buddy check
max_buddy_rh	Threshold for RH buddy check
max_buddy_p	Threshold for SLP buddy check
buddy_weight	Scaling for buddy thresholds
max_p_extend_t	Pressure range (Pa) through which a single T report may be extended
max_p_extend_w	Pressure range (Pa) through which a single wind report may be extended

Namelist: **records 5 & 6**

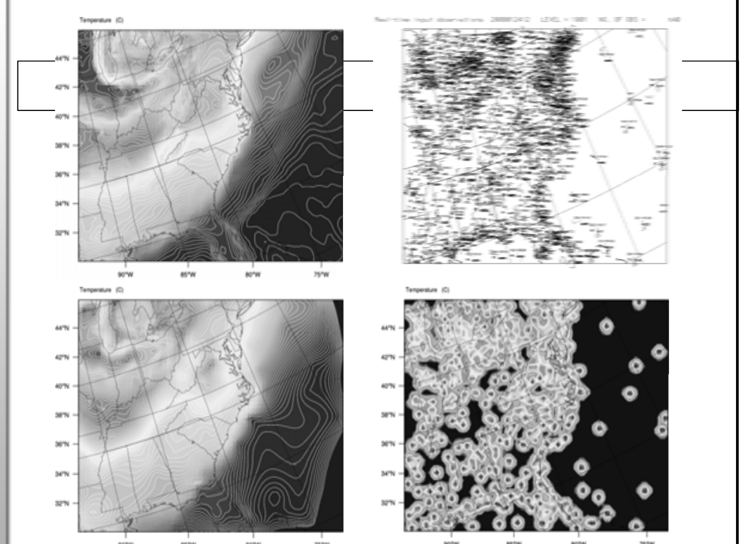
- **&record5**
  - A Bunch of print flags for various categories of printout
    - ".TRUE." will turn on a lot of printout
    - ".FALSE." will turn off printout
- **&record6**
  - No namelist record6

Mesoscale & Mircoscale Meteorology Division / NCAR

Namelist: **&record7**

use_first_guess	.TRUE.
f4d	Turn on (.TRUE.) or off (.FALSE.) the creation of surface analysis files *(wrf_sfdda_d0x)*
intf4d	Time interval (s) for surface analyses
lagtem	Use a lag-time (.TRUE.) or temporal interpolation (.FALSE.) for surface analysis first guess.

Mesoscale & Mircoscale Meteorology Division / NCAR



Mesoscale & Mircoscale Meteorology Division / NCAR

Namelist: **&record8**

smooth_type	1-2-1 or smoother/desmoother
smooth_sfc_wind	No. of passes for sfc wind
smooth_sfc_temp	No. of passes for sfc T
smooth_sfc_rh	No. of passes for sfc RH
smooth_sfc_slp	No. of passes for SLP
smooth_upper_wind	No. of passes for upper-air wind
smooth_upper_temp	No. of passes for upper-air T
smooth_upper_rh	No. of passes for upper-air RH

Namelist: **&record9**

OA_type	"MQD" or "Cressman"
MQD_minimum_num_obs	Minimum number of obs for MQD
MQD_maximum_num_obs	Maximum number of obs for MQD
radius_influence	Radius of influence for Cressman
OA_min_switch (T/F)	Switch to Cressman if too few obs for MQD
OA_max_switch (T/F)	Switch to Cressman if too many obs for MQD

Mesoscale & Mircoscale Meteorology Division / NCAR

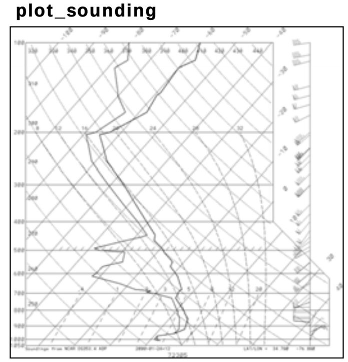
Utilities

- **plot_sound**
  - Plot sounding from the 'useful' or 'qc' wrf_obs / little_r output files
  - Namelist control: **&plot_sounding**
- **plot_level**
  - Plot data used on all levels
  - Can plot data from '3D' or 'sfc' wrf_obs / little_r output files
  - Namelist control: **&plot_level**

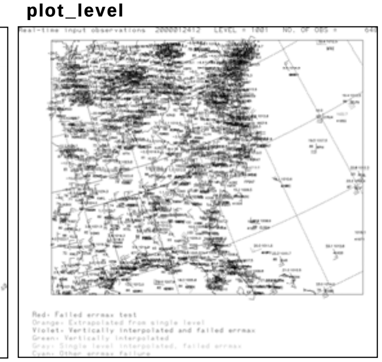
Mesoscale & Mircoscale Meteorology Division / NCAR

Utilities

plot_sounding



plot_level



Mesoscale & Mircoscale Meteorology Division / NCAR

WRF Software

WRF Software Architecture

John Michalakes, Head WRF Software Architecture

Michael Duda

Dave Gill

Outline

- Introduction
- Computing Overview
- WRF Software Overview

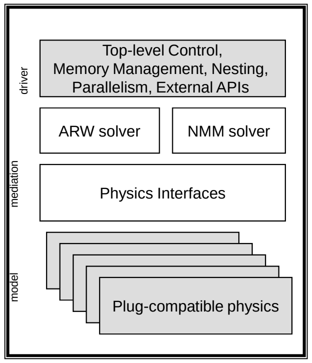
Introduction – WRF Software Characteristics

- Developed from scratch beginning around 1998, primarily Fortran and C
- Requirements emphasize flexibility over a range of platforms, applications, users, performance
- WRF develops rapidly. First released Dec 2000; current release WRF 3.0 (April 2008); next release WRF v3.1 (March 2009)
- Supported by flexible efficient architecture and implementation called the WRF Software Framework

Introduction - WRF Software Framework Overview

- Implementation of WRF Architecture
  - Hierarchical organization
  - Multiple dynamical cores
  - Plug compatible physics
  - Abstract interfaces (APIs) to external packages
  - Performance-portable
- Designed from beginning to be adaptable to today's computing environment for NWP

http://box.mmm.ucar.edu/wrf/WG2/bench/

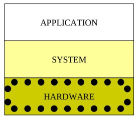


Introduction - WRF Supported Platforms

Vendor	Hardware	OS	Compiler
Apple	G4/G5 + Intel	MacOS	IBM, g95, PGI, Intel
Cray Inc.	X1, X1e	UNICOS	Cray
	Opteron	Linux	PGI, PathScale
HP/Compaq	Alpha	Tru64	Compaq
	Itanium-2	Linux	Intel
		HPUX	HP
IBM	Power-3/4/5/6	AIX	IBM
SGI	Opteron, Itanium-2	Linux	Intel
	MIPS	IRIX	SGI
Sun	UltraSPARC	Solaris	Sun
	Xeon and Athlon	Linux	PGI, Intel, g95, Pathscale
	Itanium-2 and Opteron		
ENIAC	Bunch of tubes	Mostly people	Not invented

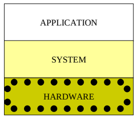
Outline

- Introduction
- Computing Overview
- WRF Software Overview



Hardware: The Computer

- The 'N' in NWP
- Components
  - Processor
    - A program counter
    - Arithmetic unit(s)
    - Some scratch space (registers)
    - Circuitry to store/retrieve from memory device
    - Cache
  - Memory
  - Secondary storage
  - Peripherals
- The implementation has been continually refined, but the basic idea hasn't changed much



Hardware has not changed much…

A computer in 1960

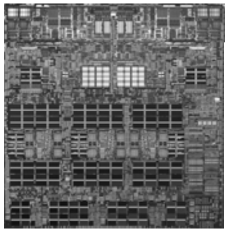
IBM 7090

6-way superscalar
36-bit floating point precision
~144 Kbytes

~50,000 flop/s
48hr 12km WRF CONUS in 600 years

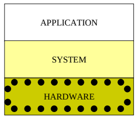
A computer in 2008

IBM P6



Dual core, 4.7 GHz chip
64-bit floating point precision
1.9 MB L2, 36 MB L3
Upto 16 GB per processor

~5,000,000,000 flop/s
48 12km WRF CONUS in 52 Hours



…how we use it has

- Fundamentally, processors haven't changed much since 1960
- Quantitatively, they haven't improved nearly enough
  - 100,000x increase in peak speed
  - 100,000x increase in memory size
  - These are too slow and too small for even a moderately large NWP run today
- We make up the difference with parallelism
  - Ganging multiple processors together to achieve $10^{11\text{-}12}$ flop/second
  - Aggregate available memories of $10^{11\text{-}12}$ bytes

~1,000,000,000,000 flop/s ~250 procs
48-h,12-km WRF CONUS in under 15 minutes

Examples

- If the machine consists of 4 nodes, each with 4 processors, how many different ways can you run a job to use all 16 processors?
  - 4 MPI processes, each with 4 threads

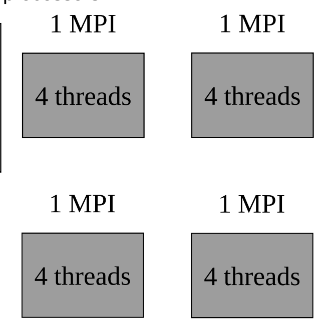
```
setenv OMP_NUM_THREADS 4
mpirun –np 4 wrf.exe
```

  - 8 MPI processes, each with 2 threads

```
setenv OMP_NUM_THREADS 2
mpirun –np 8 wrf.exe
```

  - 16 MPI processes, each with 1 thread

```
setenv OMP_NUM_THREADS 1
mpirun –np 16 wrf.exe
```



Examples

- If the machine consists of 4 nodes, each with 4 processors, how many different ways can you run a job to use all 16 processors?
  - 4 MPI processes, each with 4 threads

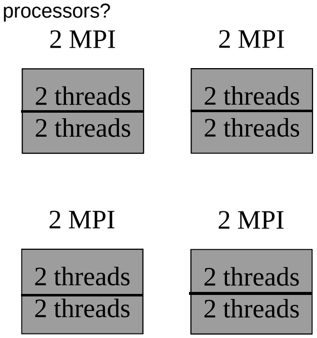
```
setenv OMP_NUM_THREADS 4
mpirun –np 4 wrf.exe
```

  - 8 MPI processes, each with 2 threads

```
setenv OMP_NUM_THREADS 2
mpirun –np 8 wrf.exe
```

  - 16 MPI processes, each with 1 thread

```
setenv OMP_NUM_THREADS 1
mpirun –np 16 wrf.exe
```



Examples

- If the machine consists of 4 nodes, each with 4 processors, how many different ways can you run a job to use all 16 processors?
  - 4 MPI processes, each with 4 threads

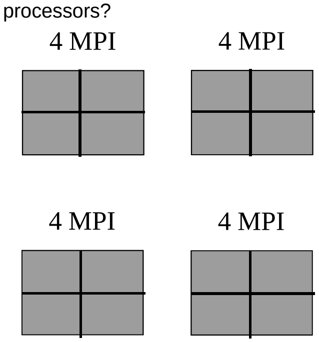
```
setenv OMP_NUM_THREADS 4
mpirun –np 4 wrf.exe
```

  - 8 MPI processes, each with 2 threads

```
setenv OMP_NUM_THREADS 2
mpirun –np 8 wrf.exe
```

  - 16 MPI processes, each with 1 thread

```
setenv OMP_NUM_THREADS 1
mpirun –np 16 wrf.exe
```



Examples (cont.)

- Note, since there are 4 nodes, we can never have fewer than 4 MPI processes because nodes do not share memory
- What happens on this same machine for the following?

```
setenv OMP_NUM_THREADS 8
mpirun –np 32
```

APPLICATION
SYSTEM
HARDWARE

Application: WRF

- WRF can be run serially or as a parallel job
- WRF uses ***domain decomposition*** to divide total amount of work over parallel processes

APPLICATION
SYSTEM
HARDWARE

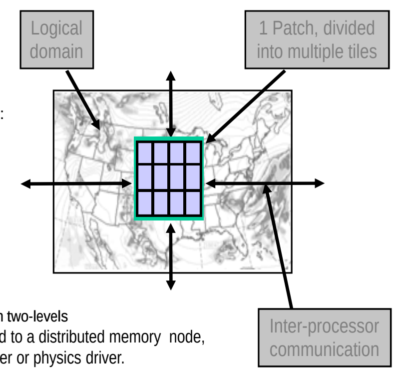
Application: WRF

- Since the process model has two levels (heavy-weight and light-weight = MPI and OpenMP), the decomposition of the application over processes has two levels:
  - The ***domain*** is first broken up into rectangular pieces that are assigned to heavy-weight processes. These pieces are called ***patches***
  - The ***patches*** may be further subdivided into smaller rectangular pieces that are called ***tiles***, and these are assigned to ***threads*** within the process.

APPLICATION
SYSTEM
HARDWARE

Parallelism in WRF: Multi-level Decomposition

- Single version of code for efficient execution on:
  - Distributed-memory
  - Shared-memory (SMP)
  - Clusters of SMPs
  - Vector and microprocessors



Model domains are decomposed for parallelism on two-levels

Patch: section of model domain allocated to a distributed memory node, this is the scope of a mediation layer solver or physics driver.

Tile: section of a patch allocated to a shared-memory processor within a node; this is also the scope of a model layer subroutine.

Distributed memory parallelism is over patches; shared memory parallelism is over tiles within patches

Distributed Memory Communications

When Needed? Communication is required between patches when a horizontal index is incremented or decremented on the right-hand-side of an assignment.

Why? On a patch boundary, the index may refer to a value that is on a different patch.

Following is an example code fragment that requires communication between patches

Note the tell-tale +1 and –1 expressions in indices for **rr**, **H1,** and **H2** arrays on right-hand side of assignment.

These are ***horizontal data dependencies*** because the indexed operands may lie in the patch of a neighboring processor. That neighbor's updates to that element of the array won't be seen on this processor.

Dr Phil We have to communicate.

Distributed Memory Communications

```
                      (module_diffusion.F )

SUBROUTINE horizontal_diffusion_s (tendency, rr, var, . . .
. . .
   DO j = jts,jte
   DO k = kts,ktf
   DO i = its,ite
      mrdx=msft(i,j)*rdx
      mrdy=msft(i,j)*rdy
      tendency(i,k,j)=tendency(i,k,j)-                             &
           (mrdx*0.5*((rr(i+1,k,j)+rr(i,k,j))*H1(i+1,k,j)-         &
                      (rr(i-1,k,j)+rr(i,k,j))*H1(i  ,k,j))+        &
            mrdy*0.5*((rr(i,k,j+1)+rr(i,k,j))*H2(i,k,j+1)-         &
                      (rr(i,k,j-1)+rr(i,k,j))*H2(i,k,j  ))-        &
            msft(i,j)*(H1avg(i,k+1,j)-H1avg(i,k,j)+                &
                       H2avg(i,k+1,j)-H2avg(i,k,j)                 &
                                )/dzetaw(k)                        &
           )
   ENDDO
   ENDDO
   ENDDO
 . . .
```

Distributed Memory Communications

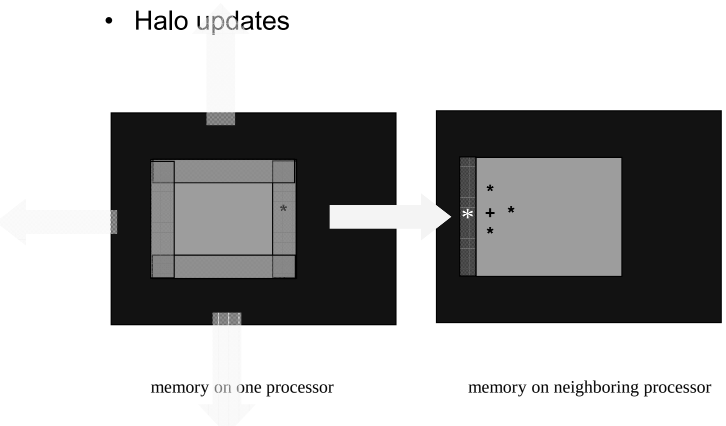
```
                      (module_diffusion.F )

SUBROUTINE horizontal_diffusion_s (tendency, rr, var, . . .
. . .
   DO j = jts,jte
   DO k = kts,ktf
   DO i = its,ite
      mrdx=msft(i,j)*rdx
      mrdy=msft(i,j)*rdy
      tendency(i,k,j)=tendency(i,k,j)-                             &
           (mrdx*0.5*((rr(i+1,k,j)+rr(i,k,j))*H1(i+1,k,j)-         &
                      (rr(i-1,k,j)+rr(i,k,j))*H1(i  ,k,j))+        &
            mrdy*0.5*((rr(i,k,j+1)+rr(i,k,j))*H2(i,k,j+1)-         &
                      (rr(i,k,j-1)+rr(i,k,j))*H2(i,k,j  ))-        &
            msft(i,j)*(H1avg(i,k+1,j)-H1avg(i,k,j)+                &
                       H2avg(i,k+1,j)-H2avg(i,k,j)                 &
                                )/dzetaw(k)                        &
           )
   ENDDO
   ENDDO
   ENDDO
 . . .
```

Distributed Memory Communications

```
                      (module_diffusion.F )

SUBROUTINE horizontal_diffusion_s (tendency, rr, var, . . .
. . .
   DO j = jts,jte
   DO k = kts,ktf
   DO i = its,ite
      mrdx=msft(i,j)*rdx
      mrdy=msft(i,j)*rdy
      tendency(i,k,j)=tendency(i,k,j)-                             &
           (mrdx*0.5*((rr(i+1,k,j)+rr(i,k,j))*H1(i+1,k,j)-         &
                      (rr(i-1,k,j)+rr(i,k,j))*H1(i  ,k,j))+        &
            mrdy*0.5*((rr(i,k,j+1)+rr(i,k,j))*H2(i,k,j+1)-         &
                      (rr(i,k,j-1)+rr(i,k,j))*H2(i,k,j  ))-        &
            msft(i,j)*(H1avg(i,k+1,j)-H1avg(i,k,j)+                &
                       H2avg(i,k+1,j)-H2avg(i,k,j)                 &
                                )/dzetaw(k)                        &
           )
   ENDDO
   ENDDO
   ENDDO
 . . .
```

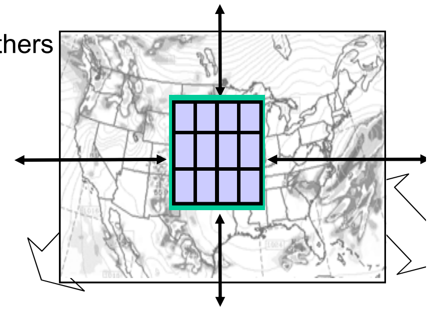
Distributed Memory MPI Communications

- Halo updates



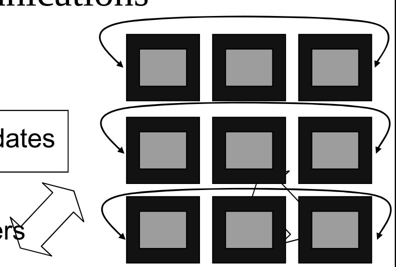
Distributed Memory (MPI) Communications

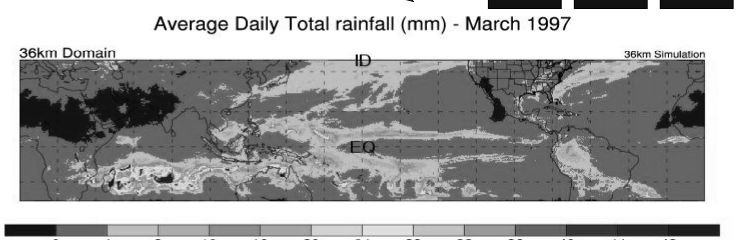
- Halo updates
- Periodic boundary updates
- Parallel transposes
- Nesting scatters/gathers



Distributed Memory (MPI) Communications

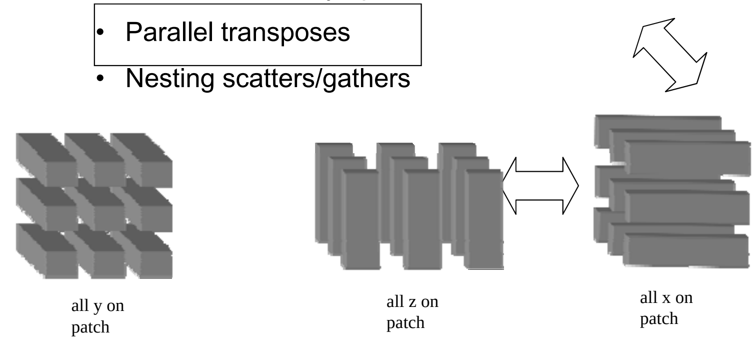
- Halo updates
- Periodic boundary updates
- Parallel transposes
- Nesting scatters/gathers



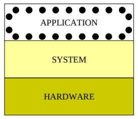


Distributed Memory (MPI) Communications

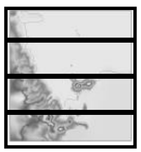
- Halo updates
- Periodic boundary updates
- Parallel transposes
- Nesting scatters/gathers



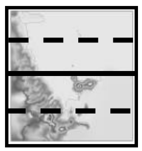
Distributed Memory (MPI) Communications



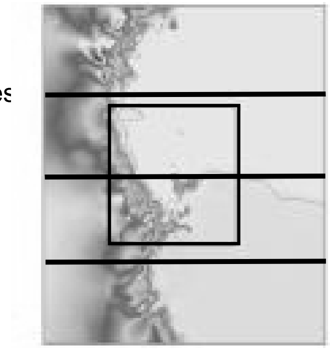
- Halo updates
- Periodic boundary updates
- Parallel transposes
- Nesting scatters/gathers



NEST:2.22 km



INTERMEDIATE: 6.66 km



COARSE
Ross Island
6.66 km

Review – Computing Overview

		Distributed Memory Parallel		Shared Memory Parallel
APPLICATION (WRF)				
Domain	*contains*	Patches	*contain*	Tiles
SYSTEM (UNIX, MPI, OpenMP)				
Job	*contains*	Processes	*contain*	Threads
HARDWARE (Processors, Memories, Wires)				
Cluster	*contains*	Nodes	*contain*	Processors

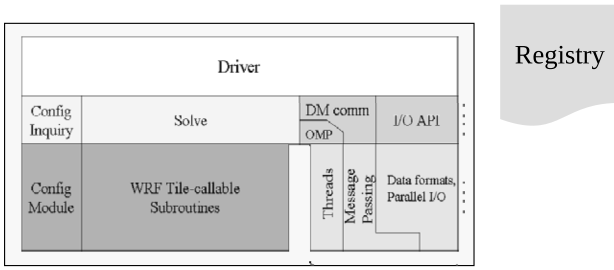
Outline

- Introduction
- Computing Overview
- WRF Software Overview

WRF Software Overview

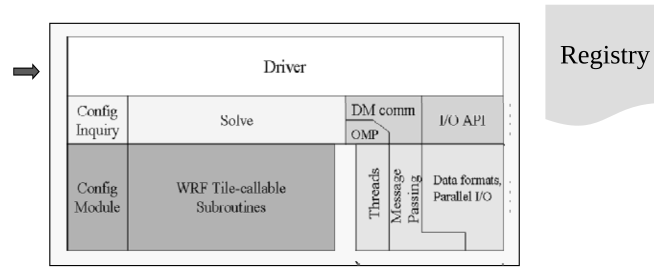
- Architecture
- Directory structure
- Model Layer Interface
- Data Structures
- I/O

WRF Software Architecture



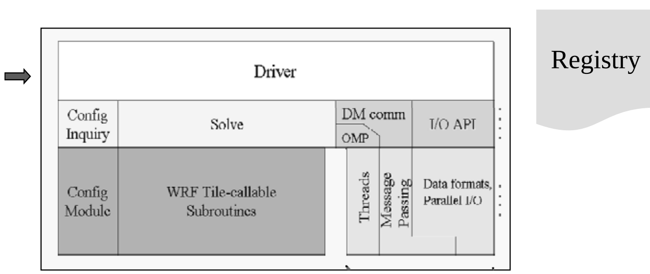
- Hierarchical software architecture
  - Insulate scientists' code from parallelism and other architecture/implementation-specific details
  - Well-defined interfaces between layers, and external packages for communications, I/O, and model coupling facilitates code reuse and exploiting of community infrastructure, e.g. ESMF.

WRF Software Architecture



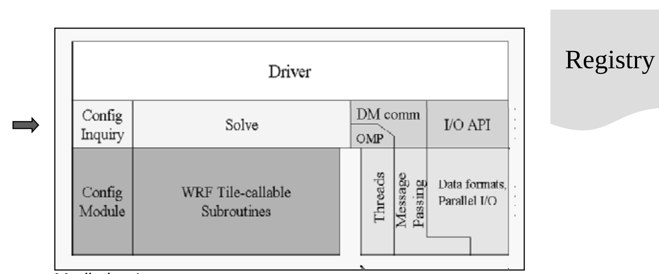
- Driver Layer
  - **Domains**: Allocates, stores, decomposes, represents abstractly as single data objects
  - **Time loop**: top level, algorithms for integration over nest hierarchy
  - **Mediation Layer calls**: nest forcing and feedback

WRF Software Architecture



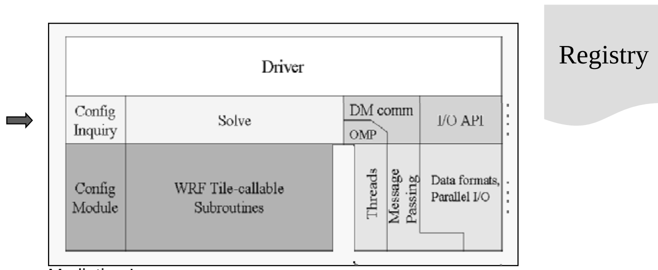
- Driver Layer
  - **Non package-specific access**: communications and I/O
  - **Utilities**: for example module_wrf_error, which is used for diagnostic prints and error stops, accessibility to run-time options

WRF Software Architecture



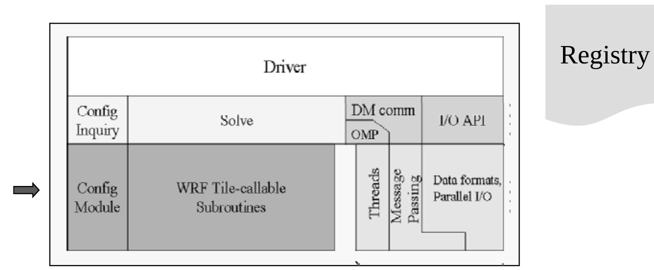
- Mediation Layer
  - Provides to the Driver Layer
    - Solve routine, which takes a domain object and advances it one time step
    - I/O routines that Driver calls when it is time to do some input or output operation on a domain
    - Nest forcing, interpolation, and feedback routines

WRF Software Architecture



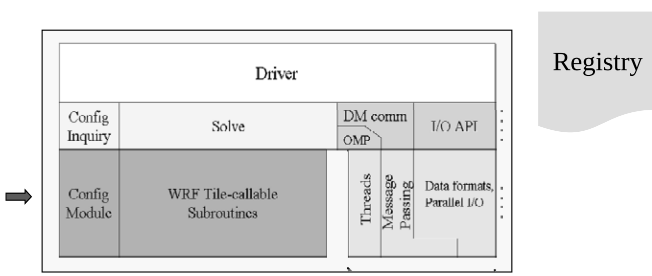
- Mediation Layer
  - Provides to Model Layer
    - The sequence of calls for doing a time-step for one domain is known in Solve routine
    - Dereferences fields in calls to physics drivers and dynamics code
    - Calls to message-passing are contained here as part of Solve routine

WRF Software Architecture



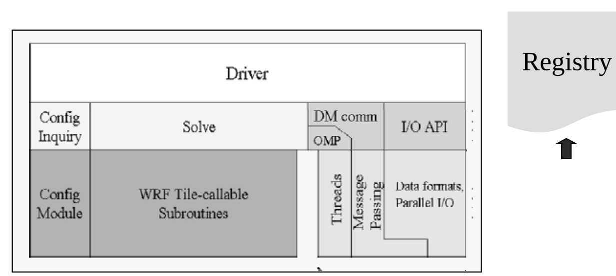
- Model Layer
  - **Information about the model itself**: machine architecture and implementation aspects abstracted out and moved into layers above
  - **Physics and Dynamics**: contains the actual WRF model routines are written to perform some computation over an arbitrarily sized/shaped subdomain

WRF Software Architecture



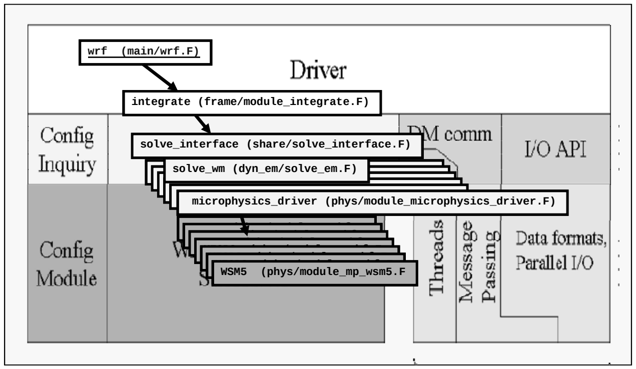
- Model Layer
  - **F77-esque**: all state data objects are simple types, passed in through argument list from physics drivers
  - **No I/O, comms, control**: Model Layer routines don't know anything about communication or I/O, executed on **one thread** – they never contain a **PRINT**, **WRITE**, or **STOP** statement
  - **Model Layer Subroutine Interface**: "tile-callable", no external COMMON, no decomposed heap data

WRF Software Architecture



- Registry: an "Active" data dictionary
  - Tabular listing of model state and attributes
  - Large sections of interface code generated automatically
  - Scientists manipulate model state simply by modifying Registry, without further knowledge of code mechanics
  - Special "cases" exist: chemistry, SST coupling

Call Structure Superimposed on Architecture



WRF Software Overview

- Architecture
- Directory structure
- Model Layer Interface
- Data Structures
- I/O

WRF Model Top-Level Directory Structure

WRF Design and Implementation Doc, p 5

DRIVER ● (dark grey)
MEDIATION ● (black)
MODEL ○ (white)

```
Makefile
README
README_test_cases
clean           } build
compile         } scripts
configure       }
Registry/         CASE input files
arch/             machine build rules
● dyn_em/       }
● dyn_nnm/      }
external/       }
● frame/        } source
inc/            } code
● main/         } directories
○ phys/         }
● share/        }
tools/          }
run/            } execution
test/           } directories
```

WRF Software Overview

- Architecture
- Directory structure
- Model Layer Interface
- Data Structures
- I/O

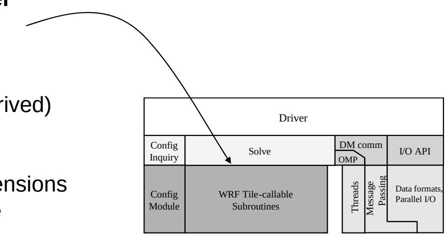
WRF Model Layer Interface

Mediation Layer / Model Layer Interface

All state arrays passed through argument list as simple (not derived) data types

Domain, memory, and run dimensions passed unambiguously in three physical dimensions

Model layer routines are called from mediation layer (physics drivers) in loops over tiles, which are multi-threaded



WRF Model Layer Interface

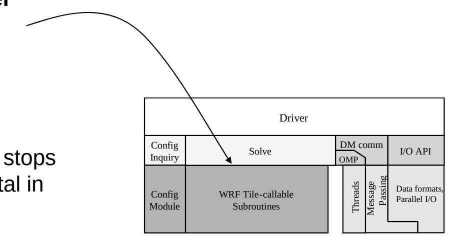
Mediation Layer / Model Layer Interface

Restrictions on Model Layer subroutines:

No I/O, communication, no stops or aborts (use wrf_error_fatal in frame/module_wrf_error.F)

No common/module storage of decomposed data (exception allowed for set-once/read-only tables)

Spatial scope of a Model Layer call is one "tile"



WRF Model Layer Interface

- Mediation layer / Model Layer Interface
- Model layer routines are called from mediation layer in loops over tiles, which are multi-threaded
- All state arrays passed through argument list as simple data types

WRF Model Layer Interface

- Domain, memory, and run dimensions passed unambiguously in three physical dimensions
- Restrictions on model layer subroutines
  - No I/O, communication, no stops or aborts (use wrf_error_fatal in frame/module_wrf_error.F)
  - No common/module storage of decomposed data (exception allowed for set-once/read-only tables)
  - Spatial scope of a Model Layer call is one "tile"
  - Temporal scope of a call is limited by coherency

WRF Model Layer Interface

```
SUBROUTINE driver_for_some_physics_suite (
    . . .
!$OMP DO PARALLEL
   DO ij = 1, numtiles
      its = i_start(ij) ; ite = i_end(ij)
      jts = j_start(ij) ; jte = j_end(ij)
      CALL model_subroutine( arg1, arg2, . . .
           ids , ide , jds , jde , kds , kde ,
           ims , ime , jms , jme , kms , kme ,
           its , ite , jts , jte , kts , kte )
   END DO
    . . .

 END SUBROUTINE
```

WRF Model Layer Interface

template for model layer subroutine

```
SUBROUTINE model_subroutine ( &
  arg1, arg2, arg3, … , argn,   &
  ids, ide, jds, jde, kds, kde, &  ! Domain dims
  ims, ime, jms, jme, kms, kme, &  ! Memory dims
  its, ite, jts, jte, kts, kte  )  ! Tile dims

IMPLICIT NONE

! Define Arguments (State and I1) data
REAL, DIMENSION (ims:ime,kms:kme,jms:jme) :: arg1, . . .
REAL, DIMENSION (ims:ime,jms:jme)         :: arg7, . . .
 . . .
! Define Local Data (I2)
REAL, DIMENSION (its:ite,kts:kte,jts:jte) :: loc1, . . .
 . . .
```

WRF Model Layer Interface

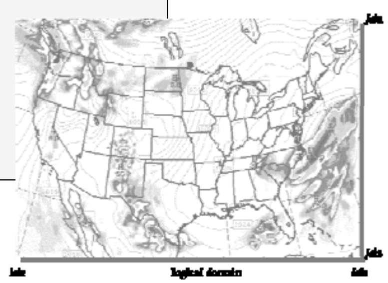
template for model layer subroutine

```
. . .
 ! Executable code; loops run over tile
 ! dimensions
 DO j = MAX(jts,jds), MIN(jte,jde-1)
   DO k = kts, kte
     DO i = MAX(its,ids), MIN(ite,ide-1)
       loc1(i,k,j) = arg1(i,k,j) + …
     END DO
   END DO
 END DO
```

template for model layer subroutine

```
SUBROUTINE model ( &
  arg1, arg2, arg3, … , argn,   &
  ids, ide, jds, jde, kds, kde, &  ! Domain dims
  ims, ime, jms, jme, kms, kme, &  ! Memory dims
  its, ite, jts, jte, kts, kte  )  ! Tile dims

IMPLICIT NONE

! Define Arguments (S and I1) data
REAL, DIMENSION (ims:ime,kms:kme,jms:jme) :: arg1, . . .
REAL, DIMENSION (ims:ime,jms:jme)         :: arg7, . . .
 . . .
! Define Local Data (I2)
REAL, DIMENSION (its:ite,kts:kte,jts:jte) :: loc1, . . .
 . . .
! Executable code; loops run over tile
! dimensions
DO j = MAX(jts,jds), MIN(jte,jde-1)
  DO k = kts, kte
   DO i = MAX(its,ids), MIN(ite,ide-1)
     loc1(i,k,j) = arg1(i,k,j) + …
    END DO
  END DO
END DO
```

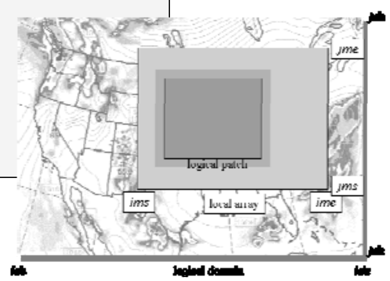
- Domain dimensions
  - Size of logical domain
  - Used for bdy tests, etc.



template for model layer subroutine

```
SUBROUTINE model ( &
  arg1, arg2, arg3, … , argn,   &
  ids, ide, jds, jde, kds, kde, &  ! Domain dims
  ims, ime, jms, jme, kms, kme, &  ! Memory dims
  its, ite, jts, jte, kts, kte  )  ! Tile dims

IMPLICIT NONE

! Define Arguments (S and I1) data
REAL, DIMENSION (ims:ime,kms:kme,jms:jme) :: arg1, . . .
REAL, DIMENSION (ims:ime,jms:jme)         :: arg7, . . .
 . . .
! Define Local Data (I2)
REAL, DIMENSION (its:ite,kts:kte,jts:jte) :: loc1, . . .
 . . .
! Executable code; loops run over tile
! dimensions
DO j = MAX(jts,jds), MIN(jte,jde-1)
  DO k = kts, kte
    DO i = MAX(its,ids), MIN(ite,ide-1)
      loc1(i,k,j) = arg1(i,k,j) + …
    END DO
  END DO
END DO
```

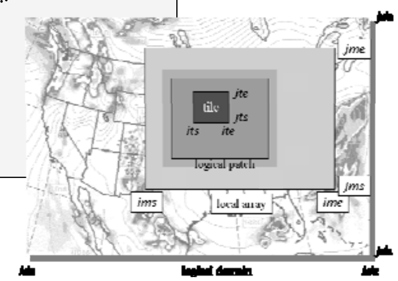
- Domain dimensions
  - Size of logical domain
  - Used for bdy tests, etc.
- Memory dimensions
  - Used to dimension dummy arguments
  - Do not use for local arrays



template for model layer subroutine

```
SUBROUTINE model ( &
  arg1, arg2, arg3, … , argn,   &
  ids, ide, jds, jde, kds, kde, &  ! Domain dims
  ims, ime, jms, jme, kms, kme, &  ! Memory dims
  its, ite, jts, jte, kts, kte  )  ! Tile dims

IMPLICIT NONE

! Define Arguments (S and I1) data
REAL, DIMENSION (ims:ime,kms:kme,jms:jme) :: arg1, . . .
REAL, DIMENSION (ims:ime,jms:jme)         :: arg7, . . .
 . . .
! Define Local Data (I2)
REAL, DIMENSION (its:ite,kts:kte,jts:jte) :: loc1, . . .
 . . .
! Executable code; loops run over tile
! dimensions
DO j = MAX(jts,jds), MIN(jte,jde-1)
  DO k = kts, kte
    DO i = MAX(its,ids), MIN(ite,ide-1)
      loc1(i,k,j) = arg1(i,k,j) + …
    END DO
  END DO
END DO
```

- Domain dimensions
  - Size of logical domain
  - Used for bdy tests, etc.
- Memory dimensions
  - Used to dimension dummy arguments
  - Do not use for local arrays
- Tile dimensions
  - Local loop ranges
  - Local array dimensions

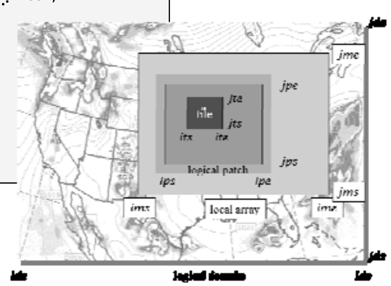


template for model layer subroutine

```
SUBROUTINE model ( &
  arg1, arg2, arg3, … , argn,   &
  ids, ide, jds, jde, kds, kde, &  ! Domain dims
  ims, ime, jms, jme, kms, kme, &  ! Memory dims
  its, ite, jts, jte, kts, kte  )  ! Tile dims

IMPLICIT NONE

! Define Arguments (S and I1) data
REAL, DIMENSION (ims:ime,kms:kme,jms:jme) :: arg1, . . .
REAL, DIMENSION (ims:ime,jms:jme)         :: arg7, . . .
 . . .
! Define Local Data (I2)
REAL, DIMENSION (its:ite,kts:kte,jts:jte) :: loc1, . . .
 . . .
! Executable code; loops run over tile
! dimensions
DO j = MAX(jt,jds), MIN(jte,jde-1)
  DO k = kts, kte
    DO i = MAX(its,ids), MIN(ite,ide-1)
      loc1(i,k,j) = arg1(i,k,j) + …
    END DO
  END DO
END DO
```

- Domain dimensions
  - Size of logical domain
  - Used for bdy tests, etc.
- Memory dimensions
  - Used to dimension dummy arguments
  - Do not use for local arrays
- Tile dimensions
  - Local loop ranges
  - Local array dimensions

- Patch dimensions
  - Start and end indices of local distributed memory subdomain
  - Available from mediation layer (solve) and driver layer; not usually needed or used at model layer

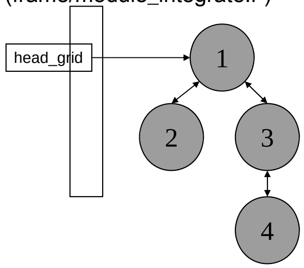


WRF Software Overview

- Architecture
- Directory structure
- Model Layer Interface
- Data Structures
- I/O

Driver Layer Data Structures: Domain Objects

- Driver layer
  - All data for a domain is an object, a domain **derived data type** (DDT)
  - The domain DDTs are dynamically allocated/deallocated
  - Linked together in a tree to represent nest hierarchy; root pointer is **head_grid**, defined in frame/module_domain.F
  - Supports recursive depth-first traversal algorithm (frame/module_integrate.F)



Model Layer Data Structures: F77

- Model layer
  - All data objects are scalars and arrays of simple types only
  - Virtually all passed in through subroutine argument lists
  - Non-decomposed arrays and “local to a module” storage are permitted with an initialization at the model start

Mediation Layer Data Structures: Objects + F77

- Mediation layer
  - One task of mediation layer is to dereference fields from DDTs
  - Therefore, sees domain data in both forms, as DDT and as individual fields which are components of the DDTs
- The name of a data type and how it is referenced differs depending on the level of the architecture

Data Structures

- WRF Data Taxonomy
  - State data
  - Intermediate data type 1 (I1)
  - Intermediate data type 2 (I2)
  - Heap storage (COMMON or Module data)

Data Structures

- WRF Data Taxonomy
  - State data
  - Intermediate data type 1 (I1)
  - Intermediate data type 2 (I2)
  - Heap storage (COMMON or Module data)

Defined in the Registry

Data Structures

- WRF Data Taxonomy
  - State data
  - Intermediate data type 1 (I1)
  - Intermediate data type 2 (I2)
  - Heap storage (COMMON or Module data)

Defined in the subroutine

Data Structures

- WRF Data Taxonomy
  - State data
  - Intermediate data type 1 (I1)
  - Intermediate data type 2 (I2)
  - Heap storage (COMMON or Module)

Defined in the module top, typically look-up tables and routine constants, NO HORIZ DECOMPOSED DATA!

Mediation/Model Layer Data Structures: State Data

- Duration: Persist between start and stop of a domain
- Represented as fields in domain data structure
  - Memory for state arrays are dynamically allocated, only big enough to hold the local subdomain's (ie. patch's) set of array elements
  - Always **memory** dimensioned
  - Declared in Registry using **state** keyword
- Only state arrays can be subject to I/O and Interprocessor communication

Mediation/Model Layer Data Structures: I1 Data

- Persist for the duration of a single time step in solve
- Represented as fields in domain data structure
  - Memory for I1 arrays are dynamically allocated, only big enough to hold the local subdomain's (ie. patch's) set of array elements
  - Always **memory** dimensioned
  - Declared in Registry using **I1** keyword
  - Typically tendency fields computed, used, and discarded in a single time step

Model Layer Data Structures: I2 Data

- Persist for the duration of a call of the physics routine
- NOT contained within the DDT structure
  - Memory for I2 arrays are dynamically allocated on subroutine entry, and automatically deallocated on exit
  - Always **tile** dimensioned
  - Not declared in the Registry, not communicated, no IO, not passed back to the solver

Grid Representation in Arrays

- Increasing indices in WRF arrays run
  - West to East (X, or I-dimension)
  - South to North (Y, or J-dimension)
  - Bottom to Top (Z, or K-dimension)
- Storage order in WRF is IKJ but this is a WRF Model convention, not a restriction of the WRF Software Framework (provides cache coherency, but long vectors possible)
- Output data has grid ordering independent of the ordering inside the WRF model

Grid Representation in Arrays

- The extent of the logical or *domain* dimensions is always the "staggered" grid dimension. That is, from the point of view of a non-staggered dimension, there is always an extra cell on the end of the domain dimension

WRF Software Overview

- Architecture
- Directory structure
- Model Layer Interface
- Data Structures
- I/O

WRF I/O

- Streams: pathways into and out of model
  - History + 11 auxiliary output streams (10 and 11 are reserved for nudging)
  - Input + 11 auxiliary input streams (10 and 11 are reserved for nudging)
  - Restart, boundary, and a special Var stream

WRF I/O

- Attributes of streams
  - Variable set
    - The set of WRF state variables that comprise one read or write on a stream
    - Defined for a stream at compile time in Registry
  - Format
    - The format of the data outside the program (e.g. NetCDF), split
    - Specified for a stream at run time in the namelist

WRF I/O

- Attributes of streams
  - Additional namelist-controlled attributes of streams
    - Dataset name
    - Time interval between I/O operations on stream
    - Starting, ending times for I/O (**specified as intervals from start of run**)

Outline - Review

- Introduction
  - WRF started 1998, clean slate, Fortran + C
  - Targeted for research and operations
- WRF Software Overview
  - Hierarchical software layers
  - Patches (MPI) and Tiles (OpenMP)
  - Strict interfaces between layers
  - Contract with developers
  - Data Structures
  - I/O

Other Run-time Options

(digital filter, global, adaptive time step, etc.)

WRF:
More Runtime Options

Wei Wang

NCAR/ESSL/MMM

More options

- Have covered basic, nesting runtime options, physics / diffusion options, nudging options..
- More are introduced here:
  - IO options (applies to ARW and NMM)
  - Vertical interpolation options (ARW)
  - SST update (ARW)
  - Adaptive-time step (ARW)
  - Digital filter (ARW)
  - Global runs (ARW)
  - IO quilting (ARW and NMM)
- Time series output (ARW and NMM)

IO Control (1)

History output control in **`&time_control`**

`history_interval_h:` history output interval in hours
`history_interval_s:` history output interval in seconds
`history_begin_h:` history output beginning time in hours
`history_begin_d:` history output beginning time in days

Complete listing in
`Registry/registry.io_boilerplate`

IO Control (2)

Optional history output in **`&time_control`**

1. change Registry.EM and recompile:

```
state integer rainc ij misc 1 - h03 ”RAINC" ""
  ”ACCUMULATED TOTAL CUMULUS PRECIPITATION"
state integer rainnc ij misc 1 - h03 ”RAINC" ""
  ”ACCUMULATED TOTAL GRID SCALE PRECIPITATION”
```

2. Edit namelist.input to output these variables:

```
auxhist3_outname = “rainfall_d<domain>”
auxhist3_interval = 10
```

Vertical interpolation options (1)

Program **`real`** for ARW only, optional, &domains:

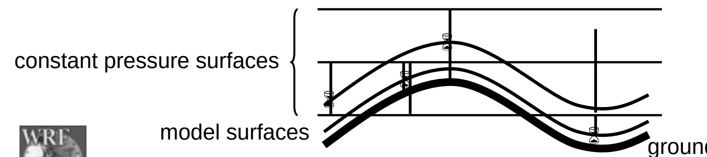
- **`use_surface:`** whether to use surface observations
- **`use_leves_below_ground:`** whether to use data below the ground
- **`lowest_lev_from_sfc:`** logical, whether surface data is used to fill the lowest model level values
- **`force_sfc_in_vinterp:`** number of levels to use surface data, default is 1
- **`extrap_type:`** how to do extrapolation: 1 - use 2 lowest levels; 2 - constant
- **`t_extrap_type :`** extrapolation option for temperature: 1 - isothermal; 2 - 6.5 K/km; 3 - adiabatic

Vertical interpolation options (2)

Program **`real`** for ARW only, optional:

- **`interp_type:`** in pressure or log pressure
- **`lagrange_order:`** linear or quadratic
- **`zap_close_levels:`** delta p where a non-surface pressure level is removed in vertical interpolation

related namelists: **`examples.namelist`**



SST update for long simulations (1)

Lower boundary update control: allow SST, seaice, monthly vegetation fraction and albedo to be updated during a model run (ARW only):

`sst_update:` 0 – no SST update
1 – update SST

Set before running **`real`**, and this will create additional output files: wrflowinp_d01, wrflowinp_d02, ..

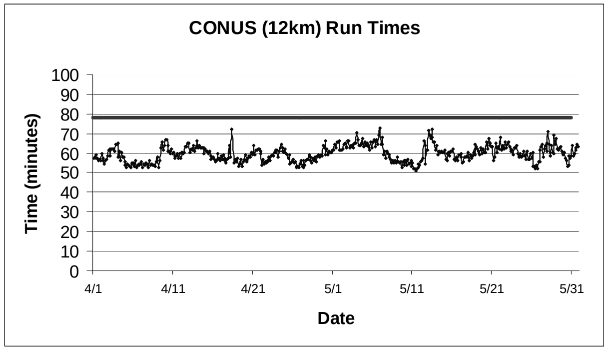
To use the files in **`wrf`**, in **`&time_control`**, add

`auxinput4_inname` = “wrflowinp_d<domain>”
`auxinput4_interval` = 360

Adaptive time steps (1)

- Adaptive-time-step is a way to maximize the model time step while keeping the model numerically stable
- New in V3. Works well for single domain. Good to use for real-time run

Adaptive time steps (2): an example



On average, forecasts finish in 60 min (50-73min) as compared to 79 min standard runtime

Adaptive time steps (3)

Namelist control: **`&domains`**

- **`use_adaptive_time_step :`** logical switch
- **`step_to_output_time:`** whether to write at exact history output times
- **`target_cfl:`** maximum cfl allowed (1.2)
- **`max_step_increase_pct:`** percentage of time step increase each time
- **`starting_time_step:`** in seconds; -1: 6*DX;
- **`max_time_step:`** in seconds; -1: 3*starting step
- **`min_time_step:`** in seconds; -1: 0.5*starting step

* USE WITH GREAT CARE

Digital filter initialization (1)

Digital filter initialization is a simple way to remove initial model imbalance:

- May be introduced by simple interpolation, or by objective analysis, or data assimilation
- It may generate spurious gravity waves in the early simulation hours, which could cause erroneous precipitation, numerical instability and degrade subsequent data assimilation

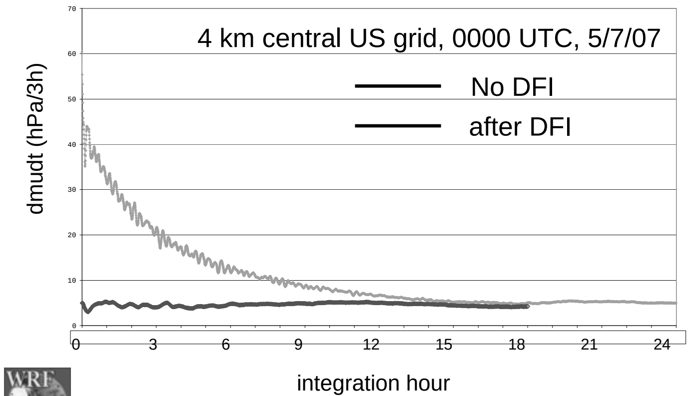
Digital filter initialization (2)

Using DFI

- can construct consistent model fields which do not exist in the initial conditions, e.g. vertical motion, cloud variables
- may reduce the spin-up problem in early simulation hours

DFI is done after program **`real`**, or data-assimilation step, just before model integration

Digital filter initialization (3)

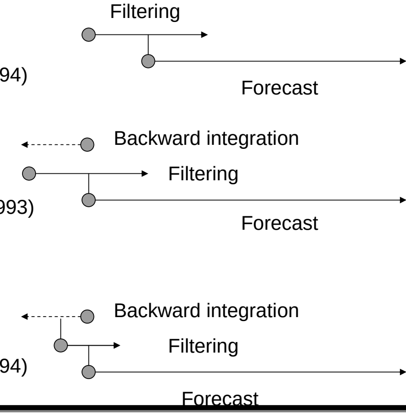


Digital filter initialization (4)

DFL:
(Lynch and Huang, 1994)

DDFI:
(Huang and Lynch, 1993)

TDFI:
(Lynch and Huang, 1994)



Digital filter inilialization (5)

Namelist control: **&dfi**

- **dfi_opt:** dfi options: 0: no DFI; 1: DFL; 2: DDFI; 3: TDFI
- **dfi_nfilter:** filter options 0 - 8, recommended 7
- **dfi_cutoff_seconds :** cutoff period
- **dfi_write_filtered_input :** logical
- **dfi_bckstop_* :** stop time for backward integration
- **dfi_fwdstop_* :** stop time for forward integration

related namelists: **examples.namelist**

Global application

Setup mostly done in WPS:

- **map_proj = 'lat-lon'**
- **e_we, e_sn: geogrid** will compute dx, dy
- See template **'namelist.wps.global'**

In the model stage:

- **fft_filter_lat:** default value is 45 degrees
- Caution: some options do not work, or have been tested with global domain. Start with template **'namelist.input.global'**

IO quilting: &namelist_quilt

Parallel I/O control:

nio_tasks_per_group (>0): allow IO to be done on separate processors. Performance improvement for large domain runs. A value of 2 to 4 works well.

io_groups (>1): number of I/O streams that the quilting applies.

Time Series Output (1)

- It is a special output in text format with file name like

  ***prefix*.d<domain>.TS**
- It outputs 14 surface variables at every time step:

  e.g. 10 m u/v, 2 m T/qv, precipitation, radiation, surface fluxes
- One file per location/weather station

Time Series Output (2)

- Not a namelist option
- Depends the presence of a file called '**tslist**' (a sample of the file is available in **WRFV3/run/**) in the run directory

```
#-----------------------------------------------#
# 24 characters for name | pfx |  LAT  |   LON  |
#-----------------------------------------------#
Cape Hallett             hallt -72.330  170.250
McMurdo Station          mcm   -77.851  166.713
```

- This file provides a list of locations where you would like to output time series
- More information in run/README.tslist and User's Guide, Chapter 5

Recommended

Start with the namelist template in a particular test directory, and the options specified in them, and make modifications.

For special applications, look for related namelists in the file examples.namelist in test/em_real/ directory.

For more information on global extension, DFI and adaptive time step, read ARW Tech Note, and User's Guide.

WRF Utilities

WRF Utilities

Cindy Bruyère

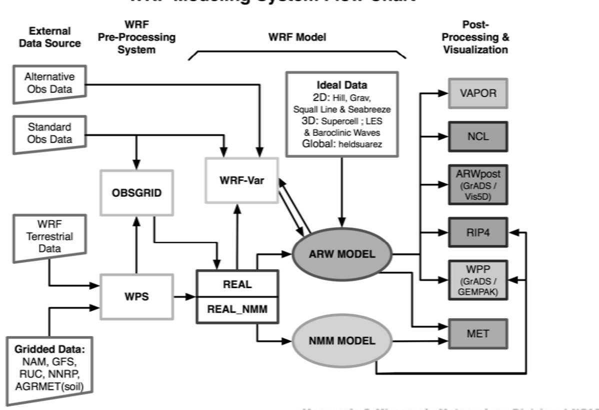
Overview

- Graphical Tools
- WRF Model Domain Design
- Intermediate Files
- netCDF
- GRIB1 / GRIB2
- Verification Tools
- Domain Wizard

Mesoscale & Mircoscale Meteorology Division / NCAR

Graphics

WRF Modeling System Flow Chart



Mesoscale & Mircoscale Meteorology Division / NCAR

Graphics

- **NCL**
  - Graphical package
  - *WRF-ARW Only*
- **ARWpost**
  - Converter (GrADS & vis5d)
  - *WRF-ARW Only*
- **RIP4**
  - Converter and interface to graphical package NCAR Graphics
- **VAPOR**
  - Converter and graphical package
  - *WRF-ARW Only*
- **WPP**
  - Converter (GrADS & GEMPAK)
- **IDL**
- **IDV**
- **MatLab**
- **GEMPAK**

Mesoscale & Mircoscale Meteorology Division / NCAR

Graphics : *ctrans*

- **Convert NCAR Graphics files (.ncgm) to ras format**
  - Single Frame in .**ncgm** file
    ctrans -d sun *file*.**ncgm** > *file*.**ras**
  - Multiple Frames in .**ncgm** file
    (med = metafile frame editor)
    med -e '1,$ split $' *file*.cgm
    ctrans -d sun *med001*.**ncgm** > *med001*.**ras**

Mesoscale & Mircoscale Meteorology Division / NCAR

Graphics : ImageMagick

- **Convert graphical files from one format to another**
  - Many options available (*rotate* frames, *trim* white space, etc.)
  - Can be used for files with single or multiple frames
  - Cannot deal with .**ncgm** files
  - http://www.imagemagick.org

convert	*file*.**pdf**	*file*.**png**
convert	*file*.**png**	*file*.**bmp**
convert	*file*.**pdf**	*file*.**gif**
convert	*file*.**ras**	*file*.**png**

Mesoscale & Mircoscale Meteorology Division / NCAR

WRF Model Domain Design

- **WPS util/ directory**
  - **plotgrids.exe**

    quick look at domains you want to create (reads namelist information to generate plot)

    create an NCAR Graphics file called 'gmeta'

    use 'idt' to view

Mesoscale & Mircoscale Meteorology Division / NCAR

WPS Intermediate Files

- **Output format of ungrid**
- **WPS util/ directory**
  - **plotfmt.exe**
    graphical interface to view intermediate file
- **Create your own intermediate files**
  - ***http://www.mmm.ucar.edu/wrf/OnLineTutorial/WPS/IM_files.htm***

Mesoscale & Mircoscale Meteorology Division / NCAR

netCDF

- **netCDF stands for *network Common Data Form***
- **netCDF is one of the current supported data formats chosen for WRF I/O API**
  - WRF I/O supports netCDF *(not fully CF compliant - Climate and Forecast Metadata Convention)*/ binary/GRIB/HDF
  - Most support graphical packages currently only support netCDF file format
- ***http://www.unidata.ucar.edu*** (*documentation*)
- **http://www.unidata.ucar.edu/software/netcdf/fguide.pdf** *(writing Fortran programs to read/write netCDF files)*

netCDF

- **Advantages of using netCDF?**
  - Platform-independent (*big_endian / little_endian*)
  - A lot of software already exist which can be used to process netCDF data
- **netCDF operators**
  - **http://nco.sourceforge.net/**
  - Stand alone programs to, which can be used to manipulate data *(performing grid point averaging / file differencing / file 'appending')*

Mesoscale & Mircoscale Meteorology Division / NCAR

NCO tools
http://nco.sourceforge.net/

- **ncdiff**
  - Difference two file
    ncdiff input1.nc input2.nc output.nc
- **ncrcat** *(nc cat)*
  - Write specified variables / times to a new file
    ncrcat -v RAINNC wrfout RAINNC.nc*
    ncrcat -d Time,0,231 -v RAINNC wrfout RAINNC.nc*
- **ncra** *(nc average)*
  - Average variables and write to a new file
    ncra -v OLR wrfout OLR.nc*
- **ncks** *(nc kitchen sink)*
  - Combination of NCO tools all in one *(handy: one tool for multiple operations)*

Mesoscale & Mircoscale Meteorology Division / NCAR

netCDF : Utilities

- **ncdump**
  - reads a netCDF dataset and prints information from the dataset
  - ncdump –h *file*
    print header (inc. list of variables in the file)
  - ncdump –v *VAR file*
    print data of the variable VAR
    ncdump –v Times *file*

Mesoscale & Mircoscale Meteorology Division / NCAR

netCDF : *ncdump -h*

```
netcdf wrfinput_d01 {
dimensions:
      Time = UNLIMITED ; // (1 currently)
      DateStrLen = 19 ;
      west_east = 73 ;
      south_north = 60 ;
      west_east_stag = 74 ;
      bottom_top = 27 ;
      south_north_stag = 61 ;
      bottom_top_stag = 28 ;
variables:
      char Times(Time, DateStrLen) ;
      float LU_INDEX(Time, south_north, west_east) ;
            LU_INDEX:FieldType = 104 ;
            LU_INDEX:MemoryOrder = "XY " ;
            LU_INDEX:description = "LAND USE CATEGORY" ;
            LU_INDEX:units = "" ;
            LU_INDEX:stagger = "" ;

………….
```


netCDF : *ncdump -h*

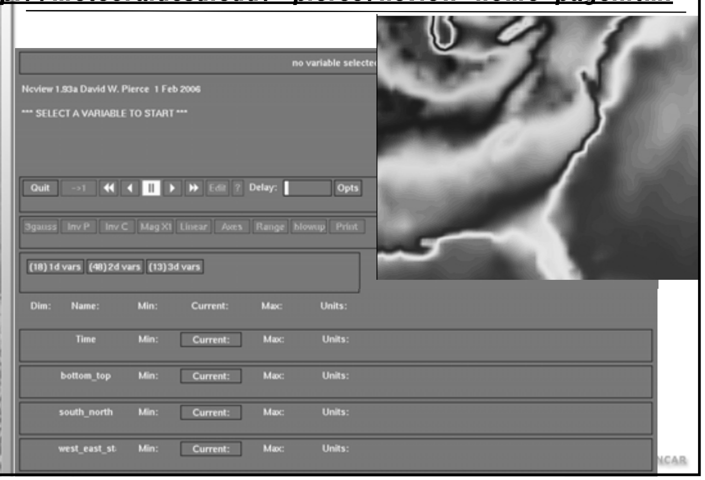
```
………..

// global attributes:
          :TITLE = " OUTPUT FROM REAL_EM V3.0 PREPROCESSOR";
          :START_DATE = "2000-01-24_12:00:00" ;
          :SIMULATION_START_DATE = "2000-01-24_12:00:00" ;
          :WEST-EAST_GRID_DIMENSION = 74 ;
          :SOUTH-NORTH_GRID_DIMENSION = 61 ;
          :BOTTOM-TOP_GRID_DIMENSION = 28 ;
          :DX = 30000.f ;
          :DY = 30000.f ;
          :GRIDTYPE = "C" ;
          :DYN_OPT = 2 ;
          :DIFF_OPT = 0 ;
          :MP_PHYSICS = 3 ;
          :SF_SFCLAY_PHYSICS = 1 ;
          :SF_SURFACE_PHYSICS = 1 ;
          :BL_PBL_PHYSICS = 1 ;
          :CU_PHYSICS = 1 ;
```


netCDF : *ncdump -v Times*

```
   ……………

data:

 Times =
  "2000-01-24_12:00:00",
  "2000-01-24_18:00:00",
  "2000-01-25_00:00:00",
  "2000-01-25_06:00:00" ;
}
```

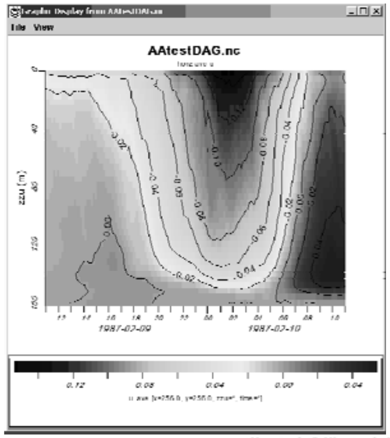

ncview

http://meteora.ucsd.edu/~pierce/ncview_home_page.html



ncBrowse

http://www.epic.noaa.gov/java/ncBrowse/



Other Utilities

- **Developed / Supported by NCAR**
- **FORTRAN program**
  - Easy to use
  - Easy to add your own code
  - Only for netCDF datasets
- **http://www.mmm.ucar.edu/wrf/users/utilities/util.htm**

Other Utilities

- **read_wrf_nc** ***(ARW & NMM)***
  - Display data inside a wrfout netCDF file
  - Specific points; min/max of fields; time series; edit data in file
- **iowrf** ***(ARW)***
  - Thinning ofnetCDF data; extracting a area; destaggering grid
- **p_interp** ***(ARW)***
  - Interpolate to pressure levels
- **v_interp** ***(ARW)***
  - Add vertical levels in wrf input and boundary files
  - For use with ndown

GRIB

- *http://dss.ucar.edu/docs/formats/grib/gribdoc/*
- **g1print.exe & g2print.exe**
  - Show data available in GRIB1 & GRIB2 files
  - Available from **util/** directory in WPS
- **grib2ctl.pl**
  - Create .ctl and .idx files, so one can plot GRIB files with GrADS *(available on web)*
- **wgrib** ***(for GRIB 1 data files)***

  wgrib –v file

  wgrib –V file

  http://www.cpc.ncep.noaa.gov/products/wesley/wgrib.html

GRIB2

- **Documentation**
  http://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc.shtml

- **GRIB2 - GRIB1 parameter conversion table**
  http://www.nco.ncep.noaa.gov/pmb/docs/grib2/GRIB2_parmeter_conversion_table.html

Product	Category	Parameter		Parameter
0	2	2	U	33
0	2	3	V	34

- **wgrib2**
  http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/

MET & Domain Wizard

- **DTC's Model Evaluation Tool Kit (MET)**
  – ***http://www.dtcenter.org/met/users/***

- **Domain Wizard**
  – GUI to create model domain and run WPS executables
  – Jeff Smith (jeff.s.smith@noaa.gov) (NOAA)
  – Tutorials and code are available from:
  http://www.wrfportal.org/tutorial.html
  http://www.wrfportal.org/DomainWizard.html
  – Suggestion – **if you are interested in using Domain Wizard**, first get familiar with WPS by running it manually