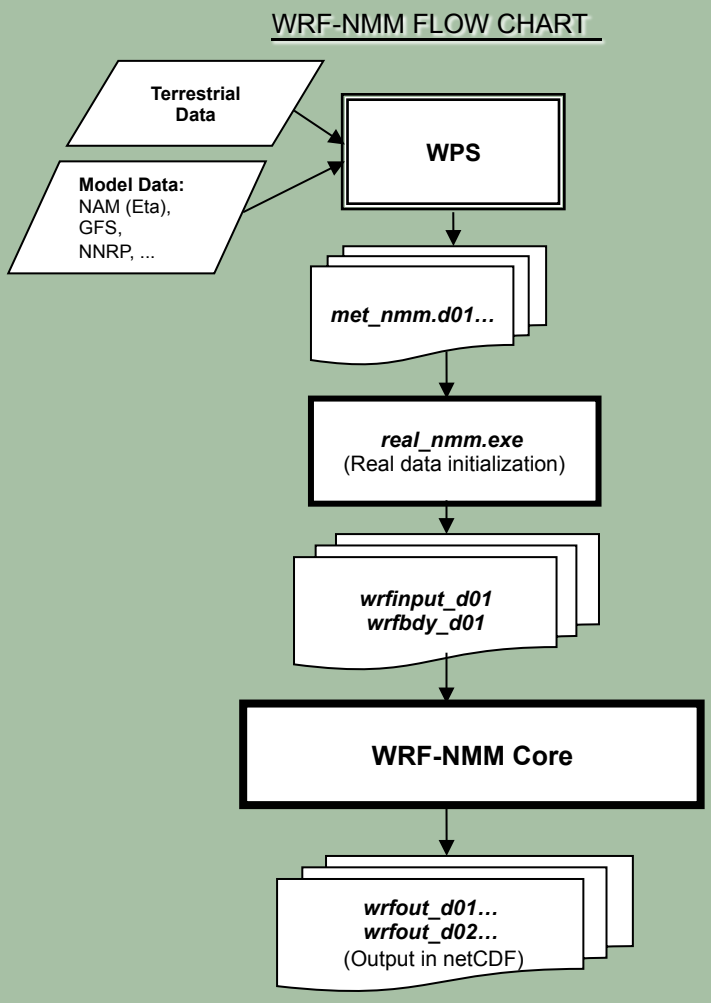
WEATHER RESEARCH & FORECASTING

NMM

Version 3 Modeling System User's Guide

October 2017



Developmental Testbed Center / National Centers for Environmental Prediction

Foreword

The Weather Research and Forecast (WRF) model system has two dynamic cores:

1. The Advanced Research WRF (ARW) developed by NCAR/MMM

2. The Non-hydrostatic Mesoscale Model (NMM) developed by NOAA/NCEP

The WRF-NMM User's Guide covers NMM specific information, as well as the common parts between the two dynamical cores of the WRF package. This document is a comprehensive guide for users of the WRF-NMM Modeling System, Version 3. The WRF-NMM is now supported only in the context of the HWRF system.

If you have any questions please refer to http://www.dtcenter.org/HurrWRF/users/support/index_help.php on how to contact HWRF helpdesk.

Contributors to this guide:
Zavisa Janjic (NOAA/NWS/NCEP/EMC)
Tom Black (NOAA/NWS/NCEP/EMC)
Matt Pyle (NOAA/NWS/NCEP/EMC)
Brad Ferrier (NOAA/NWS/NCEP/EMC)
Hui-Ya Chuang (NOAA/NWS/NCEP/EMC)
Dusan Jovic (NOAA/NWS/NCEP/EMC)
Nicole McKee (NOAA/NWS/NCEP/EMC)
Robert Rozumalski (NOAA/NWS/FDTB)
John Michalakes (NCAR/MMM)
Dave Gill (NCAR/MMM)
Jimy Dudhia (NCAR/MMM)
Michael Duda (NCAR/MMM)
Meral Demirtas (NCAR/RAL)
Michelle Harrold (NCAR/RAL)
Louisa Nance (NCAR/RAL)
Tricia Slovacek (NCAR/RAL)
Jamie Wolff (NCAR/RAL)
Ligia Bernardet (NOAA/ESRL)
Mrinal Biswas (NCAR/RAL)
Laurie Carson (NCAR/RAL)
Paula McCaslin (NOAA/ESRL)
Mark Stoelinga (University of Washington)

Acknowledgements: Parts of this document were taken from the WRF documentation provided by NCAR/MMM for the WRF User Community.

User's Guide for the NMM Core of the Weather Research and Forecast (WRF) Modeling System Version 3

User's Guide for the NMM Core of the Weather Research and Forecast (WRF) Modeling System Version 3

Chapter 1: Overview

Table of Contents

Introduction

The Nonhydrostatic Mesoscale Model (NMM) core of the Weather Research and Forecasting (WRF) system was developed by the National Oceanic and Atmospheric Adminstration (NOAA) National Centers for Environmental Prediction (NCEP). The current release is Version 3. The WRF-NMM is designed to be a flexible, state-of-the-art atmospheric simulation system that is portable and efficient on available parallel computing platforms. The WRF-NMM is suitable for use in a broad range of applications across scales ranging from meters to thousands of kilometers, including:

- Real-time NWP
- Forecast research
- Parameterization research
- Coupled-model applications
- Teaching

The NOAA/NCEP and the Developmental Testbed Center (DTC) are currently maintaining and supporting the WRF-NMM portion of the overall WRF code (Version 3) only in the context of HWRF system. That includes:

- WRF Software Framework
- WRF Preprocessing System (WPS)
- WRF-NMM dynamic solver, including one-way and two-way nesting
- Numerous physics packages contributed by WRF partners and the research community

Other components of the WRF system will be supported for community use in the future, depending on interest and available resources.

The WRF modeling system software is in the public domain and is freely available for community use.

The WRF-NMM System Program Components

Figure 1 shows a flowchart for the WRF-NMM System Version 3. As shown in the diagram, the WRF-NMM System consists of two major components:

- WRF Preprocessing System (WPS)
- WRF-NMM solver

The post-processing can be done using the Unified Post Processor (UPP). For more details on UPP software, refer to http://www.dtcenter.org/upp/users/.

WRF Preprocessing System (WPS)

This program is used for real-data simulations. Its functions include:

- Defining the simulation domain;
- Interpolating terrestrial data (such as terrain, land-use, and soil types) to the simulation domain;
- Degribbing and interpolating meteorological data from another model to the simulation domain and the model coordinate.

(For more details, see Chapter 2.)

WRF-NMM Solver

The key features of the WRF-NMM are:

- Fully compressible, non-hydrostatic model with a hydrostatic option (Janjic, 2003a).
- Hybrid (sigma-pressure) vertical coordinate.
- Arakawa E-grid.
- Forward-backward scheme for horizontally propagating fast waves, implicit scheme for vertically propagating sound waves, Adams-Bashforth Scheme for horizontal advection, and Crank-Nicholson scheme for vertical advection. The same time step is used for all terms.
- Conservation of a number of first and second order quantities, including energy and enstrophy (Janjic 1984).
- Full physics options for land-surface, planetary boundary layer, atmospheric and surface radiation, microphysics, and cumulus convection.
- One-way and two-way nesting with multiple nests and nest levels.

(For more details and references, see Chapter 4 and 5.)

The WRF-NMM code contains an initialization program ***(real_nmm.exe***; see Chapter 3) and a numerical integration program (***wrf.exe***; see Chapter 5).

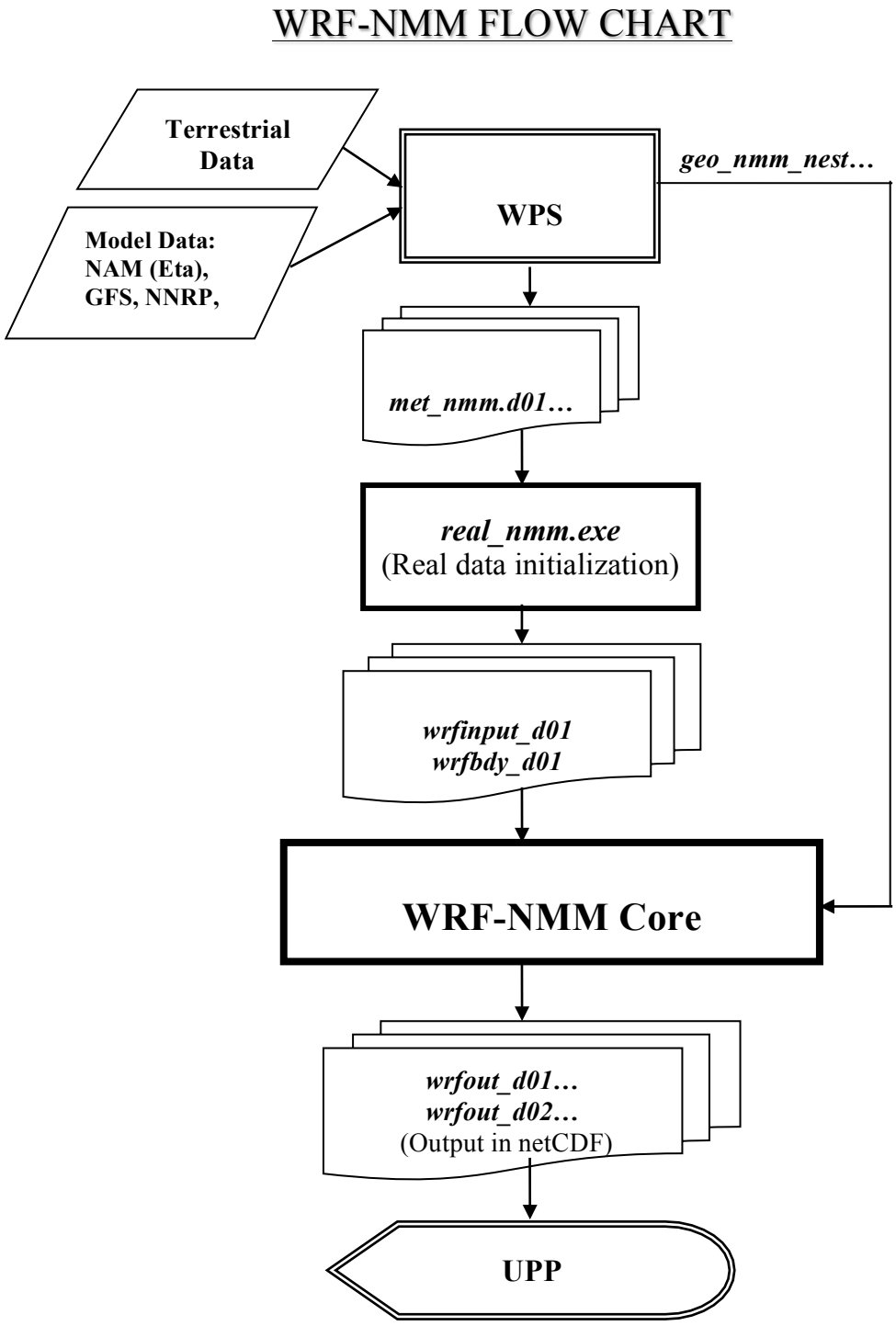


Figure 1: WRF-NMM flow chart for Version 3.

User's Guide for the NMM Core of the Weather Research and Forecast (WRF) Modeling System Version 3

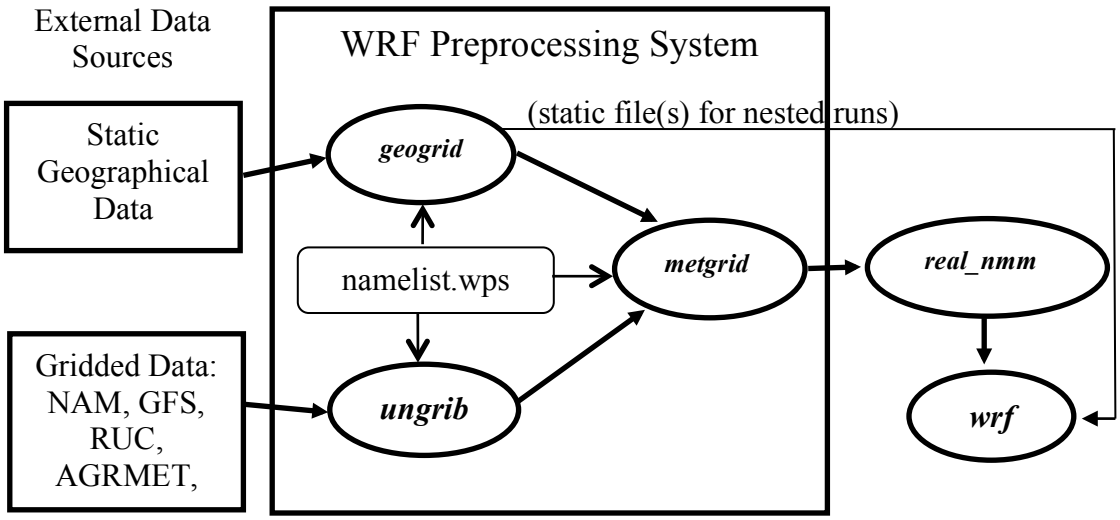
Chapter 2: WRF Preprocessing System (WPS)

Table of Contents

Introduction

The WRF Preprocessing System (WPS) is a set of three programs whose collective role is to prepare input to the *real* program for real-data simulations. Each of the programs performs one stage of the preparation: *geogrid* defines model domains and interpolates static geographical data to the grids; *ungrib* extracts meteorological fields from GRIB-formatted files; and *metgrid* horizontally interpolates the meteorological fields extracted

by ungrib to the model grids defined by geogrid. The work of vertically interpolating meteorological fields to WRF eta levels is performed within the *real* program.



The data flow between the programs of the WPS is shown in the figure above. Each of the WPS programs reads parameters from a common namelist file, as shown in the figure. This namelist file has separate namelist records for each of the programs and a shared namelist record, which defines parameters that are used by more than one WPS program. Not shown in the figure are additional table files that are used by individual programs. These tables provide additional control over the programs' operation, though they generally do not need to be changed by the user. The GEOGRID.TBL, METGRID.TBL, and Vtable files are explained later in this document.

The build mechanism for the WPS, which is very similar to the build mechanism used by the WRF model, provides options for compiling the WPS on a variety of platforms. When MPICH libraries and suitable compilers are available, the metgrid and geogrid programs may be compiled for distributed memory execution, which allows large model domains to be processed in less time. The work performed by the ungrib program is not amenable to parallelization, so ungrib may only be run on a single processor.

The purpose of this chapter is to provide overview of the WPS system, however not all the capabilities listed here are supported with the HWRF system. Users are encouraged to refer to the HWRF Users Guide for more information related to compiling, running and supported features available with the HWRF system.

Function of Each WPS Program

The WPS consists of three independent programs: *geogrid*, *ungrib*, and *metgrid*. Also included in the WPS are several utility programs, which are described in the section on

utility programs. A brief description of each of the three main programs is given below, with further details presented in subsequent sections.

Program geogrid

The purpose of geogrid is to define the simulation domains, and interpolate various terrestrial data sets to the model grids. The simulation domains are defined using information specified by the user in the "geogrid" namelist record of the WPS namelist file, namelist.wps. In addition to computing the latitude, longitude, and map scale factors at every grid point, geogrid will interpolate soil categories, land use category, terrain height, annual mean deep soil temperature, monthly vegetation fraction, monthly albedo, maximum snow albedo, and slope category to the model grids by default. Global data sets for each of these fields are provided through the WRF and HWRF download page, and, because these data are time-invariant, they only need to be downloaded once. Several of the data sets are available in only one resolution, but others are made available in resolutions of 30", 2', 5', and 10'; here, " denotes arc seconds and ' denotes arc minutes. The user need not download all available resolutions for a data set, although the interpolated fields will generally be more representative if a resolution of data near to that of the simulation domain is used. However, users who expect to work with domains having grid spacings that cover a large range may wish to eventually download all available resolutions of the static terrestrial data.

Besides interpolating the default terrestrial fields, the geogrid program is general enough to be able to interpolate most continuous and categorical fields to the simulation domains. New or additional data sets may be interpolated to the simulation domain through the use of the table file, GEOGRID.TBL. The GEOGRID.TBL file defines each of the fields that will be produced by geogrid; it describes the interpolation methods to be used for a field, as well as the location on the file system where the data set for that field is located.

Output from geogrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, geogrid can be made to write its output in NetCDF for easy visualization using external software packages, including ncview and NCL.

Program ungrib

The ungrib program reads GRIB files, "degribs" the data, and writes the data in a simple format, called the intermediate format (see the section on writing data to the intermediate format for details of the format). The GRIB files contain time-varying meteorological fields and are typically from another regional or global model, such as NCEP's NAM or GFS models. The ungrib program can read GRIB Edition 1 and, if compiled with a "GRIB2" option, GRIB Edition 2 files.

GRIB files typically contain more fields than are needed to initialize WRF. Both versions of the GRIB format use various codes to identify the variables and levels in the GRIB file. Ungrib uses tables of these codes – called Vtables, for "variable tables" – to define which fields to extract from the GRIB file and write to the intermediate format. Details

about the codes can be found in the WMO GRIB documentation and in documentation from the originating center. Vtables for common GRIB model output files are provided with the ungrib software.

Vtables are provided for NAM 104 and 212 grids, the NAM AWIP format, GFS, the NCEP/NCAR Reanalysis archived at NCAR, RUC (pressure level data and hybrid coordinate data), AFWA's AGRMET land surface model output, ECMWF, and other data sets. Users can create their own Vtable for other model output using any of the Vtables as a template; further details on the meaning of fields in a Vtable are provided in the section on creating and editing Vtables.

Ungrib can write intermediate data files in any one of three user-selectable formats: WPS – a new format containing additional information useful for the downstream programs; SI – the previous intermediate format of the WRF system; and MM5 format, which is included here so that ungrib can be used to provide GRIB2 input to the MM5 modeling system. Any of these formats may be used by WPS to initialize WRF, although the WPS format is recommended.

Program metgrid

The metgrid program horizontally interpolates the intermediate-format meteorological data that are extracted by the ungrib program onto the simulation domains defined by the geogrid program. The interpolated metgrid output can then be ingested by the WRF real program. The range of dates that will be interpolated by metgrid are defined in the "share" namelist record of the WPS namelist file, and date ranges must be specified individually in the namelist for each simulation domain. Since the work of the metgrid program, like that of the ungrib program, is time-dependent, metgrid is run every time a new simulation is initialized.

Control over how each meteorological field is interpolated is provided by the METGRID.TBL file. The METGRID.TBL file provides one section for each field, and within a section, it is possible to specify options such as the interpolation methods to be used for the field, the field that acts as the mask for masked interpolations, and the grid staggering (e.g., U, V in ARW; H, V in NMM) to which a field is interpolated.

Output from metgrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, metgrid can be made to write its output in NetCDF for easy visualization using external software packages.

Running the WPS

Note: ***For software requirements and how to compile the WRF Preprocessing System package for HWRF, see Chapter 2 of HWRF Users Guide.***

There are essentially three main steps to running the WRF Preprocessing System:

1. Define a model coarse domain and any nested domains with *geogrid*.
2. Extract meteorological fields from GRIB data sets for the simulation period with *ungrib*.
3. Horizontally interpolate meteorological fields to the model domains with *metgrid*.

When multiple simulations are to be run for the same model domains, it is only necessary to perform the first step once; thereafter, only time-varying data need to be processed for each simulation using steps two and three. Similarly, if several model domains are being run for the same time period using the same meteorological data source, it is not necessary to run ungrib separately for each simulation. Below, the details of each of the three steps are explained.

Please refer to the HWRF Users Guide to learn how to run WPS for the HWRF system.

Step 1: Define model domains with geogrid

In the root of the WPS directory structure, symbolic links to the programs geogrid.exe, ungrib.exe, and metgrid.exe should exist if the WPS software was successfully installed. In addition to these three links, a namelist.wps file should exist. Thus, a listing in the WPS root directory should look something like:

```
> ls -l
drwxr-xr-x 2   4096 arch
-rwxr-xr-x 1   1672 clean
-rwxr-xr-x 1   3510 compile
-rw-r--r-- 1  85973 compile.log
-rwxr-xr-x 1   4257 configure
-rw-r--r-- 1   2486 configure.wps
-rw-rw-r-- 1    398 c_test.c
-rw-rw-r-- 1   1528 c_test.o
-rw-rw-r-- 1    185 fort_netcdf.f
-rw-rw-r-- 1    119 f_test.f90
drwxr-xr-x 4   4096 geogrid
lrwxrwxrwx 1     23 geogrid.exe -> geogrid/src/geogrid.exe
-rwxr-xr-x 1   1328 link_grib.csh
drwxr-xr-x 3   4096 metgrid
lrwxrwxrwx 1     23 metgrid.exe -> metgrid/src/metgrid.exe
-rw-r--r-- 1   1101 namelist.wps
-rw-r--r-- 1   1987 namelist.wps.all_options
-rw-r--r-- 1   2079 namelist.wps.fire
-rw-r--r-- 1   1075 namelist.wps.global
-rw-r--r-- 1    652 namelist.wps.nmm
-rw-r--r-- 1   4786 README
drwxr-xr-x 4   4096 ungrib
lrwxrwxrwx 1     21 ungrib.exe -> ungrib/src/ungrib.exe
drwxr-xr-x 3   4096 util
```

The model coarse domain and any nested domains are defined in the `&geogrid` namelist record of the namelist.wps file, and, additionally, parameters in the `&share` namelist record need to be set. An example of these two namelist records is given below, and the

user is referred to the description of namelist variables for more information on the purpose and possible values of each variable.

```
&share
  wrf_core = "NMM",
  start_date = "2014-10-14_12:00:00",
  end_date = "2014-10-14_12:00:00",
  max_dom = 3,
  interval_seconds = 21600,
  io_form_geogrid = 2,
  nocolons = T,
/

&geogrid
  parent_id = 1, 1, 2,
  parent_grid_ratio = 1, 3, 3,
  i_parent_start = 1, 10, 10,
  j_parent_start = 1, 10, 10,
  e_we = 288, 288, 288,
  e_sn = 576, 576, 576,
  geog_data_res = "2m", "2m", "2m",
  dx = 0.135,
  dy = 0.135,
  map_proj = "rotated_ll",
  ref_lat = 19.9,
  ref_lon = -66.0,
  geog_data_path = "./geog-data/",
  opt_geogrid_tbl_path = "./",
  ref_x = 105.0,
  ref_y = 159.0,
/
```

To summarize a set of typical changes to the `&share` namelist record relevant to geogrid, the WRF dynamical core must first be selected with `wrf_core`. If WPS is being run for an ARW simulation, `wrf_core` should be set to `"ARW"`, and if running for an NMM simulation, it should be set to `"NMM"`. Since geogrid produces only time-independent data, the `start_date`, `end_date`, and `interval_seconds` variables are ignored by geogrid. The total number of domains (in the case of ARW) or nesting levels (in the case of NMM) must be chosen with `max_dom`. Optionally, a location (if not the default, which is the current working directory) where domain files should be written may be indicated with the `opt_output_from_geogrid_path` variable, and the format of these domain files may be changed with `io_form_geogrid`. The `nocolons` variable set to true replaces the colons with underscores in the output file names.

In the `&geogrid` namelist record, the projection of the simulation domain is defined, as are the size and location of all model grids. The map projection to be used for the model domains is specified with the `map_proj` variable and must be set to `rotated_ll` for WRF-NMM.

Besides setting variables related to the projection, location, and coverage of model domains, the path to the static geographical data sets must be correctly specified with the

geog_data_path variable. Also, the user may select which resolution of static data geogrid will interpolate from using the geog_data_res variable, whose value should match one of the resolutions of data in the GEOGRID.TBL. If the full set of static data are downloaded from the WRF download page, possible resolutions include '30s', '2m', '5m', and '10m', corresponding to 30-arc-second data, 2-, 5-, and 10-arc-minute data.

Depending on the value of the wrf_core namelist variable, the appropriate GEOGRID.TBL file must be used with geogrid, since the grid staggerings to which WPS interpolates differ between dynamical cores. For the ARW, the GEOGRID.TBL.ARW file should be used, and for the NMM, the GEOGRID.TBL.NMM file should be used. For the HWRF system hwrf_GEOGRID.TBL should be used. Please refer to HWRF Users Guide to find the appropriate file. Selection of the appropriate GEOGRID.TBL is accomplished by linking the correct file to GEOGRID.TBL in the geogrid directory (or in the directory specified by opt_geogrid_tbl_path, if this variable is set in the namelist).

```
> ls geogrid/GEOGRID.TBL

lrwxrwxrwx 1      15 GEOGRID.TBL -> GEOGRID.TBL.NMM
```

For more details on the meaning and possible values for each variable in the namelist, the user is referred to a description of the namelist variables.

Having suitably defined the simulation coarse domain and nested domains in the namelist.wps file, the geogrid.exe executable may be run to produce domain files. In the case of ARW domains, the domain files are named geo_em.d0N.nc, where N is the number of the nest defined in each file. When run for NMM domains, geogrid produces the file geo_nmm.d01.nc for the coarse domain, and geo_nmm_nest.l0N.nc files for each nesting level N. Also, note that the file suffix will vary depending on the io_form_geogrid that is selected. To run geogrid, issue the following command:

```
> ./geogrid.exe
```

When geogrid.exe has finished running, the message

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

should be printed, and a listing of the WPS root directory (or the directory specified by opt_output_from_geogrid_path, if this variable was set) should show the domain files. If not, the geogrid.log file may be consulted in an attempt to determine the possible cause of failure. For more information on checking the output of geogrid, the user is referred to the section on checking WPS output.

Step 2: Extracting meteorological fields from GRIB files with ungrib

Having already downloaded meteorological data in GRIB format, the first step in extracting fields to the intermediate format involves editing the `&share` and `&ungrib` namelist records of the namelist.wps file – the same file that was edited to define the simulation domains. An example of the two namelist records is given below.

```
&share
  wrf_core = "NMM",
  start_date = "2014-10-14_12:00:00",
  end_date = "2014-10-14_12:00:00",
  max_dom = 3,
  interval_seconds = 21600,
  io_form_geogrid = 2,
  nocolons = T,
/

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

In the `&share` namelist record, the variables that are of relevance to ungrib are the starting and ending times of the coarse domain (`start_date` and `end_date`; alternatively, `start_year`, `start_month`, `start_day`, `start_hour`, `end_year`, `end_month`, `end_day`, and `end_hour`) and the interval between meteorological data files (`interval_seconds`). In the `&ungrib` namelist record, the variable `out_format` is used to select the format of the intermediate data to be written by ungrib; the metgrid program can read any of the formats supported by ungrib, and thus, any of `'WPS'`, `'SI'`, and `'MM5'` may be specified for `out_format`, although `'WPS'` is recommended. Also in the `&ungrib` namelist, the user may specify a path and prefix for the intermediate files with the `prefix` variable. For example, if `prefix` were set to `'ARGRMET'`, then the intermediate files created by ungrib would be named according to AGRMET:*YYYY-MM-DD_HH*, where *YYYY-MM-DD_HH* is the valid time of the data in the file.

After suitably modifying the namelist.wps file, a Vtable must be supplied, and the GRIB files must be linked (or copied) to the filenames that are expected by ungrib. The WPS is supplied with Vtable files for many sources of meteorological data, and the appropriate Vtable may simply be symbolically linked to the file Vtable, which is the Vtable name expected by ungrib. For example, if the GRIB data are from the GFS model, this could be accomplished with

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

The ungrib program will try to read GRIB files named GRIBFILE.AAA, GRIBFILE.AAB, …, GRIBFILE.ZZZ. In order to simplify the work of linking the GRIB files to these filenames, a shell script, link_grib.csh, is provided. The link_grib.csh script takes as a command-line argument a list of the GRIB files to be linked. For example, if the GRIB data were downloaded to the directory /data/gfs, the files could be linked with link_grib.csh as follows:

```
> ls /data/gfs
```

```
-rw-r--r-- 1  42728372 gfs_080324_12_00
-rw-r--r-- 1  48218303 gfs_080324_12_06

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

After linking the GRIB files and Vtable, a listing of the WPS directory should look something like the following:

```
> ls
drwxr-xr-x 2     4096 arch
-rwxr-xr-x 1     1672 clean
-rwxr-xr-x 1     3510 compile
-rw-r--r-- 1    85973 compile.output
-rwxr-xr-x 1     4257 configure
-rw-r--r-- 1     2486 configure.wps
-rw-r--r-- 1  1957004 geo_nmm.d01.nc
-rw-r--r-- 1  4745324 geo_nmm.d02.nc
drwxr-xr-x 4     4096 geogrid
lrwxrwxrwx 1       23 geogrid.exe -> geogrid/src/geogrid.exe
-rw-r--r-- 1    11169 geogrid.log
lrwxrwxrwx 1       38 GRIBFILE.AAA -> /data/gfs/gfs_080324_12_00
lrwxrwxrwx 1       38 GRIBFILE.AAB -> /data/gfs/gfs_080324_12_06
-rwxr-xr-x 1     1328 link_grib.csh
drwxr-xr-x 3     4096 metgrid
lrwxrwxrwx 1       23 metgrid.exe -> metgrid/src/metgrid.exe
-rw-r--r-- 1     1094 namelist.wps
-rw-r--r-- 1     1987 namelist.wps.all_options
-rw-r--r-- 1     1075 namelist.wps.global
-rw-r--r-- 1      652 namelist.wps.nmm
-rw-r--r-- 1     4786 README
drwxr-xr-x 4     4096 ungrib
lrwxrwxrwx 1       21 ungrib.exe -> ungrib/src/ungrib.exe
drwxr-xr-x 3     4096 util
lrwxrwxrwx 1       33 Vtable -> ungrib/Variable_Tables/Vtable.GFS
```

After editing the namelist.wps file and linking the appropriate Vtable and GRIB files, the ungrib.exe executable may be run to produce files of meteorological data in the intermediate format. Ungrib may be run by simply typing the following:

```
> ./ungrib.exe >& ungrib.output
```

Since the ungrib program may produce a significant volume of output, it is recommended that ungrib output be redirected to a file, as in the command above. If ungrib.exe runs successfully, the message

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

will be written to the end of the ungrib.output file, and the intermediate files should appear in the current working directory. The intermediate files written by ungrib will

have names of the form `FILE:`*`YYYY-MM-DD_HH`* (unless, of course, the `prefix` variable was set to a prefix other than `'FILE'`).

```
      > ls
      drwxr-xr-x 2       4096 arch
      -rwxr-xr-x 1       1672 clean
      -rwxr-xr-x 1       3510 compile
      -rw-r--r-- 1      85973 compile.output
      -rwxr-xr-x 1       4257 configure
      -rw-r--r-- 1       2486 configure.wps
      -rw-r--r-- 1  154946888 FILE:2008-03-24_12
      -rw-r--r-- 1  154946888 FILE:2008-03-24_18
      -rw-r--r-- 1    1957004 geo_nmm.d01.nc
      -rw-r--r-- 1    4745324 geo_nmm.d02.nc
      drwxr-xr-x 4       4096 geogrid
      lrwxrwxrwx 1         23 geogrid.exe -> geogrid/src/geogrid.exe
      -rw-r--r-- 1      11169 geogrid.log
      lrwxrwxrwx 1         38 GRIBFILE.AAA ->
/data/gfs/gfs_080324_12_00
      lrwxrwxrwx 1         38 GRIBFILE.AAB ->
/data/gfs/gfs_080324_12_06
      -rwxr-xr-x 1       1328 link_grib.csh
      drwxr-xr-x 3       4096 metgrid
      lrwxrwxrwx 1         23 metgrid.exe -> metgrid/src/metgrid.exe
      -rw-r--r-- 1       1094 namelist.wps
      -rw-r--r-- 1       1987 namelist.wps.all_options
      -rw-r--r-- 1       1075 namelist.wps.global
      -rw-r--r-- 1        652 namelist.wps.nmm
      -rw-r--r-- 1       4786 README
      drwxr-xr-x 4       4096 ungrib
      lrwxrwxrwx 1         21 ungrib.exe -> ungrib/src/ungrib.exe
      -rw-r--r-- 1       1418 ungrib.log
      -rw-r--r-- 1      27787 ungrib.output
      drwxr-xr-x 3       4096 util
      lrwxrwxrwx 1         33 Vtable ->
ungrib/Variable_Tables/Vtable.GFS
```

Step 3: Horizontally interpolating meteorological data with metgrid

In the final step of running the WPS, meteorological data extracted by ungrib are horizontally interpolated to the simulation grids defined by geogrid. In order to run metgrid, the namelist.wps file must be edited. In particular, the `&share` and `&metgrid` namelist records are of relevance to the metgrid program. Examples of these records are shown below.

```
&share
  wrf_core = "NMM",
  start_date = "2014-10-14_12:00:00",
  end_date = "2014-10-14_12:00:00",
  max_dom = 3,
  interval_seconds = 21600,
  io_form_geogrid = 2,
  nocolons = T,
```

```
/

&metgrid
  fg_name = "FILE",
  io_form_metgrid = 2,
  opt_metgrid_tbl_path = "./",

/
```

By this point, there is generally no need to change any of the variables in the `&share` namelist record, since those variables should have been suitably set in previous steps. If the `&share` namelist was not edited while running geogrid and ungrib, however, the WRF dynamical core, number of domains, starting and ending times, interval between meteorological data, and path to the static domain files must be set in the `&share` namelist record, as described in the steps to run geogrid and ungrib.

In the `&metgrid` namelist record, the path and prefix of the intermediate meteorological data files must be given with `fg_name`, the full path and file names of any intermediate files containing constant fields may be specified with the `constants_name` variable, and the output format for the horizontally interpolated files may be specified with the `io_form_metgrid` variable. Other variables in the "metgrid" namelist record, namely, `opt_output_from_metgrid_path` and `opt_metgrid_tbl_path`, allow the user to specify where interpolated data files should be written by metgrid and where the METGRID.TBL file may be found.

As with geogrid and the GEOGRID.TBL file, a METGRID.TBL file appropriate for the WRF core must be linked in the metgrid directory (or in the directory specified by `opt_metgrid_tbl_path`, if this variable is set). For the WRF-NMM system the METGRID.TBL.NMM should be linked. ***For the HWRF system please refer to the HWRF Users Guide.***

```
> ls metgrid/METGRID.TBL

lrwxrwxrwx 1      15 METGRID.TBL.NMM -> METGRID.TBL
```

After suitably editing the namelist.wps file and verifying that the correct METGRID.TBL will be used, metgrid may be run by issuing the command

```
> ./metgrid.exe
```

If metgrid successfully ran, the message

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

will be printed. After successfully running, metgrid output files should appear in the WPS root directory (or in the directory specified by `opt_output_from_metgrid_path`, if this variable was set). These files will be named `met_em.d0N.YYYY-MM-DD_HH:mm:ss.nc` in

the case of ARW domains, where `N` is the number of the nest whose data reside in the file, or `met_nmm.d01.YYYY-MM-DD_HH:mm:ss.nc` in the case of NMM domains. Here, `YYYY-MM-DD_HH:mm:ss` refers to the date of the interpolated data in each file. If these files do not exist for each of the times in the range given in the `&share` namelist record, the metgrid.log file may be consulted to help in determining the problem in running metgrid.

Creating Nested Domains with the WPS

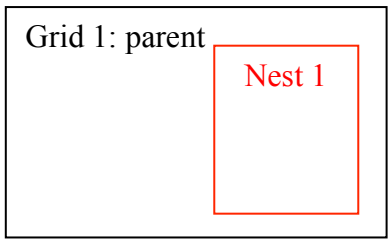
At this time, the WRF-NMM supports one-way and two-way stationary and moving (if running an HWRF configuration, see HWRF User's Guide) nests. Because the WRF-NMM nesting strategy was targeted towards the capability of moving nests, time-invariant information, such as topography, soil type, albedo, etc. for a nest must be acquired over the entire domain of the coarsest grid even though, for a stationary nest, that information will only be used over the location where the nest is initialized. ***The HWRF system uses two telescopic nests (described below) with two-way nesting. More information about how the nests work for HWRF, please refer to the HWRF Scientific Documentation.***

Running the WPS for WRF-NMM nested-domain simulations is essentially no more difficult than running for a single-domain case; the ***geogrid*** program simply processes more than one grid when it is run, rather than a single grid.

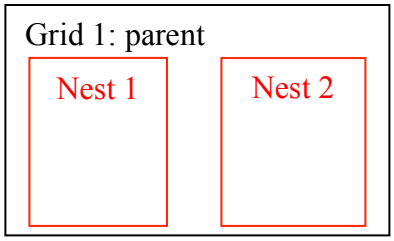
The number of grids is virtually unlimited. Grids may be located side by side (i.e., two nests may be children of the same parent and located on the same nest level), or telescopically nested. The nesting ratio for the WRF-NMM is always 3. Hence, the grid spacing of a nest is always 1/3 of its parent.

The nest level is dependent on the parent domain. If one nest is defined inside the coarsest domain, the nest level will be one and one additional static file will be created. If two nests are defined to have the same parent, again, only one additional static file will be created.

For example:



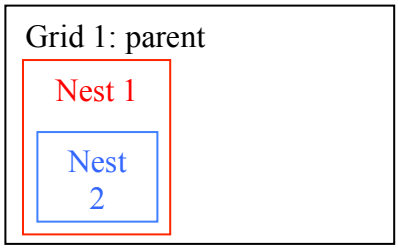
OR



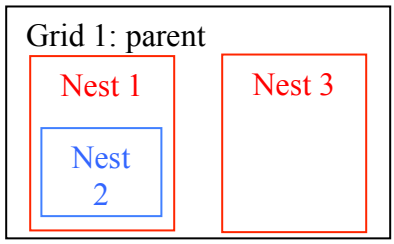
will create an output file for the parent domain: ***geo_nmm.d01.nc*** and one higher resolution output file for nest level one: ***geo_nmm_nest.l01.nc***

If, however, two telescopic nests are defined (nest 1 inside the parent and nest 2 inside nest 1), then two additional static files will be created. Even if an additional nest 3 was added at the same grid spacing as nest 1, or at the same grid spacing as nest 2, there would still be only two additional static files created.

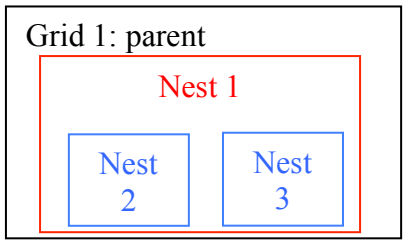
For example:



OR



OR



will create an output file for the parent domain: ***geo_nmm.d01.nc***, one output file with three times higher resolution for nest level one: ***geo_nmm_nest.l01.nc***, and one output file with nine times higher resolution for nest level two: ***geo_nmm_nest.l02.nc.***

In order to specify an additional nest level, a number of variables in the *namelist.wps* file must be given lists of values with a format of one value per nest separated by commas. The variables that need a list of values for nesting include: *parent_id*, *parent_grid_ratio*, *i_parent_start*, *j_parent_start*, *s_we*, *e_we*, *s_sn*, *e_sn*, and *geog_data_res*.

In the ***namelist.wps*,** the first change to the `&share` namelist record is to the ***max_dom*** variable, which must be set to the total number of nests in the simulation, including the coarsest domain. Having determined the number of nests, all of the other affected namelist variables must be given a list of *N* values, one for each nest. The only other change to the `&share` namelist record is to the starting and ending times. Here, a starting and ending time must be given for each nest, with the restriction that a nest cannot begin before its parent domain or end after its parent domain; also, it is suggested that nests be given starting and ending times that are identical to the desired starting times of the nest *when running WPS*. This is because the nests get their lateral boundary conditions from their parent domain, and thus, only the initial time for a nest needs to be processed by WPS. It is important to note that, *when running WRF*, the actual starting and ending times for all nests must be given in the WRF ***namelist.input*** file.

The remaining changes are to the `&geogrid` namelist record. In this record, the parent of each nest must be specified with the ***parent_id*** variable. Every nest must be a child of exactly one other nest, with the coarse domain being its own parent. Related to the identity of a nest's parent is the nest refinement ratio with respect to a nest's parent, which is given by the ***parent_grid_ratio*** variable; this ratio determines the nominal grid spacing for a nest in relation to the grid spacing of the its parent. **Note:** This ratio must always be set to 3 for the WRF-NMM.

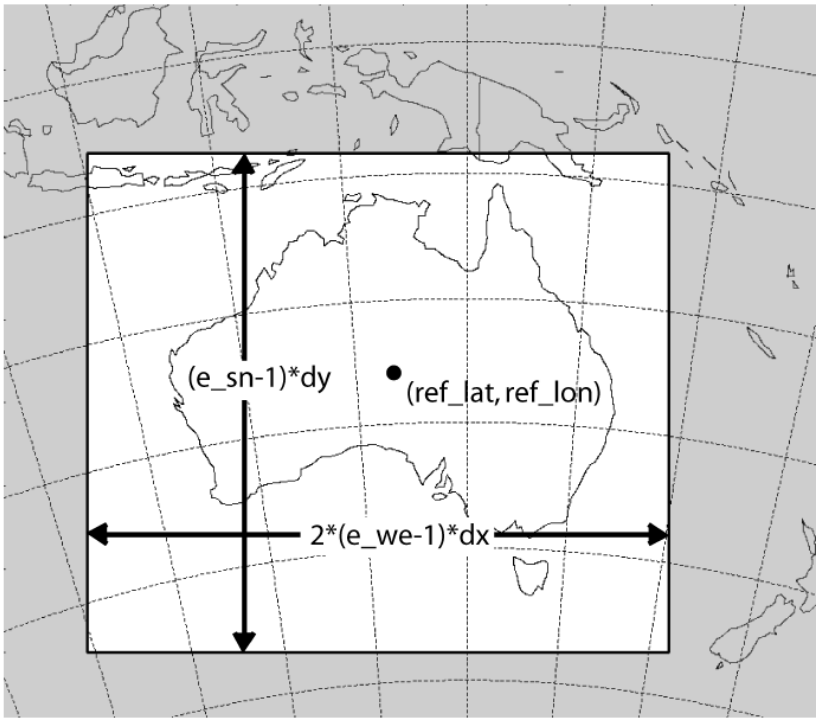
Next, the lower-left corner of a nest is specified as an **(*i*, *j*)** location in the nest's parent domain; this specification is done through the ***i_parent_start*** and ***j_parent_start*** variables, and the specified location is given with respect to a mass point on the E-grid. Finally, the dimensions of each nest, in grid points, are given for each nest using the ***s_we***, ***e_we***, ***s_sn***, and ***e_sn*** variables. An example is shown in the figure below, where it may be seen how each of the above-mentioned variables is found. Currently, the starting grid point values in the south-north (***s_sn***) and west-east (***s_we***) directions must be specified as 1, and the ending grid point values (***e_sn*** and ***e_we***) determine, essentially, the full dimensions of the nest. The following namelist section will generate domains as shown in the following figure.

```
&geogrid
  parent_id = 1, 1,
  parent_grid_ratio = 1, 3,
  i_parent_start = 1, 31
  j_parent_start = 1, 17
  e_we = 97, 112,
  e_sn = 70, 94,
```

Note: *For the WRF-NMM the variables* ***i_parent_start****,* ***j_parent_start, s_we****,* ***e_we****,* ***s_sn****, and* ***e_sn*** *are ignored during the WPS processing because the higher resolution static files for each nest level are created for the entire coarse domain. These variables, however, are used when running the WRF-NMM model.*

Finally, for each nest, the resolution of source data to interpolate from is specified with the ***geog_data_res*** variable.

For a complete description of these namelist variables, the user is referred to the description of namelist variables.



Selecting Between USGS and MODIS-based Land Use Classifications

By default, the geogrid program will interpolate land use categories from USGS 24-category data. However, the user may select an alternative set of land use categories based on the MODIS land-cover classification of the International Geosphere-Biosphere Programme and modified for the Noah land surface model. Although the MODIS-based data contain 20 categories of land use, these categories are not a subset of the 24 USGS categories; users interested in the specific categories in either data set can find a listing of

the land use classes in the section on land use and soil categories. *It must be emphasized that the MODIS-based categories should only be used with the Noah land surface model in WRF.* ***The use of MODIS Land Use is not tested with the HWRF system.***

The 20-category MODIS-based land use data may be selected instead of the USGS data at run-time through the `geog_data_res` variable in the **`&geogrid`** namelist record. This is accomplished by prefixing each resolution of static data with the string "modis_30s+". For example, in a three-domain configuration, where the `geog_data_res` variable would ordinarily be specified as

```
geog_data_res = '10m', '2m', '30s'
```

the user should instead specify

```
geog_data_res = 'modis_30s+10m', 'modis_30s+2m', 'modis_30s+30s'
```

The effect of this change is to instruct the geogrid program to look, in each entry of the GEOGRID.TBL file, for a resolution of static data with a resolution denoted by 'modis_30s', and if such a resolution is not available, to instead look for a resolution denoted by the string following the '+'. Thus, for the GEOGRID.TBL entry for the LANDUSEF field, the MODIS-based land use data, which is identified with the string 'modis_30s', would be used instead of the '10m', '2m', and '30s' resolutions of USGS data in the example above; for all other fields, the '10m', '2m', and '30s' resolutions would be used for the first, second, and third domains, respectively. As an aside, when none of the resolutions specified for a domain in `geog_data_res` are found in a GEOGRID.TBL entry, the resolution denoted by 'default' will be used.

Selecting Static Data for the Gravity Wave Drag Scheme

The gravity wave drag by orography (GWDO) scheme in the NMM (available in version 3.1) requires fourteen static fields from the WPS. In fact, these fields will be interpolated by the geogrid program regardless of whether the GWDO scheme will be used in the model. When the GWDO scheme will not be used, the fields will simply be ignored in WRF and the user need not be concerned with the resolution of data from which the fields are interpolated. However, it is recommended that these fields be interpolated from a resolution of source data that is slightly *lower* (i.e., coarser) in resolution than the model grid; consequently, if the GWDO scheme will be used, care should be taken to select an appropriate resolution of GWDO static data. Currently, five resolutions of GWDO static data are available: 2-degree, 1-degree, 30-minute, 20-minute, and 10-minute, denoted by the strings '2deg', '1deg', '30m', '20m', and '10m', respectively. To select the resolution to interpolate from, the user should prefix the resolution specified for the `geog_data_res` variable in the **`&geogrid`** namelist record by the string "XXX+", where XXX is one of the five available resolutions of GWDO static data. For example, in a model configuration with a 48-km grid spacing, the `geog_data_res` variable might typically be specified as

```
geog_data_res = '10m',
```

However, if the GWDO scheme were employed, the finest resolution of GWDO static data that is still lower in resolution than the model grid would be the 30-minute data, in which case the user should specify

```
geog_data_res = '30m+10m',
```

If none of '2deg', '1deg', '30m', or '20m' are specified in combination with other resolutions of static data in the `geog_data_res` variable, the '10m' GWDO static data will be used, since it is also designated as the 'default' resolution in the GEOGRID.TBL file. It is worth noting that, if 10-minute resolution GWDO data are to be used, but a different resolution is desired for other static fields (e.g., topography height), the user should simply omit '10m' from the value given to the `geog_data_res` variable, since specifying

```
geog_data_res = '10m+30s',
```

for example, would cause geogrid to use the 10-mintute data in preference to the 30-second data for the non-GWDO fields, such as topography height and land use category, as well as for the GWDO fields.

Using Multiple Meteorological Data Sources

The metgrid program is capable of interpolating time-invariant fields, and it can also interpolate from multiple sources of meteorological data. ***Please refer to the HWRF Users Guide for supported datasets.*** The first of these capabilities uses the `constants_name` variable in the `&metgrid` namelist record. This variable may be set to a list of filenames – including path information where necessary – of intermediate-formatted files which contains time-invariant fields, and which should be used in the output for every time period processed by metgrid. For example, short simulations may use a constant SST field; this field need only be available at a single time, and may be used by setting the `constants_name` variable to the path and filename of the SST intermediate file. Typical uses of `constants_name` might look like

```
&metgrid
 constants_name = '/data/ungribbed/constants/SST_FILE:2006-08-16_12'
/
```

or

```
&metgrid
 constants_name = 'LANDSEA', 'SOILHGT'
/
```

The second metgrid capability – that of interpolating data from multiple sources – may be useful in situations where two or more complementary data sets need to be combined to produce the full input data needed by `real`. To interpolate from multiple sources of time-varying, meteorological data, the `fg_name` variable in the `&metgrid` namelist record should be set to a list of prefixes of intermediate files, including path information when

necessary. When multiple path prefixes are given, and the same meteorological field is available from more than one of the sources, data from the last-specified source will take priority over all preceding sources. Thus, data sources may be prioritized by the order in which the sources are given.

As an example of this capability, if surface fields are given in one data source and upper-air data are given in another, the values assigned to the `fg_name` variable may look something like:

```
&metgrid
 fg_name = '/data/ungribbed/SFC', '/data/ungribbed/UPPER_AIR'
/
```

To simplify the process of extracting fields from GRIB files, the `prefix` namelist variable in the `&ungrib` record may be employed. This variable allows the user to control the names of (and paths to) the intermediate files that are created by ungrib. The utility of this namelist variable is most easily illustrated by way of an example. Suppose we wish to work with the North American Regional Reanalysis (NARR) data set, which is split into separate GRIB files for 3-dimensional atmospheric data, surface data, and fixed-field data. We may begin by linking all of the "3D" GRIB files using the `link_grib.csh` script, and by linking the NARR Vtable to the filename `Vtable`. Then, we may suitably edit the `&ungrib` namelist record before running ungrib.exe so that the resulting intermediate files have an appropriate prefix:

```
&ungrib
 out_format = 'WPS',
 prefix = 'NARR_3D',
/
```

After running ungrib.exe, the following files should exist (with a suitable substitution for the appropriate dates):

```
NARR_3D:2008-08-16_12
NARR_3D:2008-08-16_15
NARR_3D:2008-08-16_18
...
```

Given intermediate files for the 3-dimensional fields, we may process the surface fields by linking the surface GRIB files and changing the `prefix` variable in the namelist:

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

Again running ungrib.exe, the following should exist in addition to the NARR_3D files:

```
NARR_SFC:2008-08-16_12
NARR_SFC:2008-08-16_15
NARR_SFC:2008-08-16_18
```

```
...
```

Finally, the fixed file is linked with the `link_grib.csh` script, and the `prefix` variable in the namelist is again set:

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

Having run ungrib.exe for the third time, the fixed fields should be available in addition to the surface and "3D" fields:

```
NARR_FIXED:1979-11-08_00
```

For the sake of clarity, the fixed file may be renamed to remove any date information, for example, by renaming it to simply `NARR_FIXED`, since the fields in the file are static. In this example, we note that the NARR fixed data are only available at a specific time, 1979 November 08 at 0000 UTC, and thus, the user would need to set the correct starting and ending time for the data in the `&share` namelist record before running ungrib on the NARR fixed file; of course, the times should be re-set before metgrid is run.

Given intermediate files for all three parts of the NARR data set, metgrid.exe may be run after the `constants_name` and `fg_name` variables in the `&metgrid` namelist record are set:

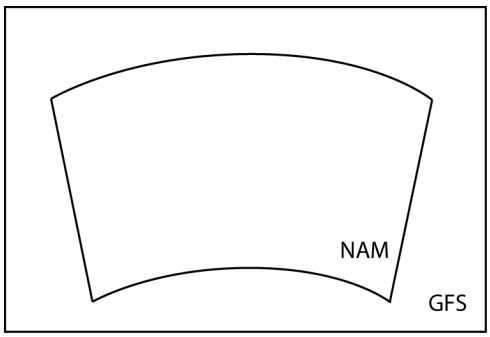
```
&metgrid
 constants_name = 'NARR_FIXED',
 fg_name = 'NARR_3D', 'NARR_SFC'
/
```

Although less common, another situation where multiple data sources would be required is when a source of meteorological data from a regional model is insufficient to cover the entire simulation domain, and data from a larger regional model, or a global model, must be used when interpolating to the remaining points of the simulation grid.

For example, to use NAM data wherever possible, and GFS data elsewhere, the following values might be assigned in the namelist:

```
&metgrid
 fg_name = '/data/ungribbed/GFS', '/data/ungribbed/NAM'
/
```

Then the resulting model domain would use data as shown in the figure below.



If no field is found in more than one source, then no prioritization need be applied by metgrid, and each field will simply be interpolated as usual; of course, each source should cover the entire simulation domain to avoid areas of missing data.

Alternative Initialization of Lake SSTs

The default treatment of sea-surface temperatures – both for oceans and lakes – in the metgrid program involves simply interpolating the SST field from the intermediate files to all water points in the WRF domain. However, if the lakes that are resolved in the WRF domain are not resolved in the GRIB data, and especially if those lakes are geographically distant from resolved water bodies, the SST field over lakes will most likely be extrapolated from the nearest resolved water bodies in the GRIB data; this situation can lead to lake SST values that are either unrealistically warm or unrealistically cold.

Without a higher-resolution SST field for metgrid to use, one alternative to extrapolating SST values for lakes is to manufacture a "best guess" at the SST for lakes. In the metgrid and real programs, this can be done using a combination of a special land use data set that distinguishes between lakes and oceans, and a field to be used as a proxy for SST over lakes. A special land use data set is necessary, since WRF's real pre-processing program needs to know where the manufactured SST field should be used instead of the interpolated SST field from the GRIB data. ***This feature is not tested with the HWRF system***.

The alternative procedure for initializing lake SSTs is summarized in the following steps:

1. If they have not already been downloaded (either as a separate tar file or as part of the 'full' geographical data tar file), obtain the special land use data sets that distinguish between lakes and oceans. Two such data sets – based on USGS and MODIS land use categories – may be downloaded through the WRF download page. For simplicity, it is recommended to place the two directories in the same directory as the other static geographical data sets (e.g., topo_30s, soiltype_top_30s, etc.) used by geogrid, since doing so will eliminate the need to modify the GEOGRID.TBL file. If the landuse_30s_with_lakes and modis_landuse_21class_30s directories are placed in a location different from the other static data sets, it will be necessary to change the

paths to these directories from relative paths to absolute paths in the GEOGRID.TBL file.

2. Before running geogrid, change the specification of `geog_data_res` in the `&geogrid` namelist record to specify either the USGS-based or the MODIS-based land use data with inland water bodies. For example, in a two-domain configuration, setting

```
geog_data_res = 'usgs_lakes+10m', 'usgs_lakes+2m',
```

would tell geogrid to use the USGS-based land use data for both domains, and to use the 10-minute resolution data for other static fields in domain 1 and the 2-minute resolution data for other static fields in domain 2; for MODIS-based data, `usgs_lakes` should be replaced by `modis_lakes`.

Running geogrid should result in output files that use a separate category for inland water bodies instead of the general water category used for oceans and seas. The lake category is identified by the global attribute ISLAKE in the geogrid output files; this attribute should be set to either 28 (in the case of USGS-based data) or 21 (in the case of the MODIS-based data). See, e.g., the list of WPS output fields, where a value of -1 for ISLAKE indicates that there is no separate lake category.

3. After running the ungrib program, use the avg_tsfc.exe utility program to create an intermediate file containing a daily-average surface air temperature field, which will be substituted for the SST field only over lakes by the real program; for more information on the avg_tsfc.exe utility, see the section on WPS utility programs.

4. Before running the metgrid program, add the TAVGSFC file created in the previous step to the specification of `constants_name` in the `&metgrid` record of the namelist.wps file.

5. Run WRF's real.exe program as usual after setting the number of land categories (`num_land_cat`) in the `&physics` record of the namelist.input file so that it matches the value of the global attribute NUM_LAND_CAT in the metgrid files. If the global attribute ISLAKE in the metgrid files indicates that there is a special land use category for lakes, the real program will substitute the TAVGSFC field for the SST field only over those grid points whose category matches the lake category; additionally, the real program will change the land use category of lakes back to the general water category (the category used for oceans), since neither the LANDUSE.TBL nor the VEGPARM.TBL files contain an entry for a lake category.

Parallelism in the WPS

If the dimensions of the domains to be processed by the WPS become too large to fit in the memory of a single CPU, it is possible to run the geogrid and metgrid programs in a distributed memory configuration. In order to compile geogrid and metgrid for distributed memory execution, the user must have MPI libraries installed on the target machine, and

must have compiled WPS using one of the "DM parallel" configuration options. Upon successful compilation, the geogrid and metgrid programs may be run with the *mpirun* or *mpiexec* commands, or through a batch queuing system, depending on the machine. ***Please refer to HWRF Users Guide on how to compile and run WPS for the HWRF system***.

As mentioned earlier, the work of the ungrib program is not amenable to parallelization, and, further, the memory requirements for ungrib's processing are independent of the memory requirements of geogrid and metgrid; thus, ungrib is always compiled for a single processor and run on a single CPU, regardless of whether a "DM parallel" configuration option was selected during configuration.

Each of the standard WRF I/O API formats (NetCDF, GRIB1, binary) has a corresponding parallel format, whose number is given by adding 100 to the io_form value (i.e., the value of `io_form_geogrid` and `io_form_metgrid`) for the standard format. It is not necessary to use a parallel io_form, but when one is used, each CPU will read/write its input/output to a separate file, whose name is simply the name that would be used during serial execution, but with a four-digit processor ID appended to the name. For example, running geogrid on four processors with `io_form_geogrid=102` would create output files named geo_em.d01.nc.0000, geo_em.d01.nc.0001, geo_em.d01.nc.0002, and geo_em.d01.nc.0003 for the coarse domain.

During distributed-memory execution, model domains are decomposed into rectangular patches, with each processor working on a single patch. When reading/writing from/to the WRF I/O API format, each processor reads/writes only its patch. Consequently, if a parallel io_form is chosen for the output of geogrid, metgrid must be run using the same number of processors as were used to run geogrid. Similarly, if a parallel io_form is chosen for the metgrid output files, the real program must be run using the same number of processors. Of course, it is still possible to use a standard io_form when running on multiple processors, in which case all data for the model domain will be distributed/collected upon input/output. As a final note, when geogrid or metgrid are run on multiple processors, each processor will write its own log file, with the log file names being appended with the same four-digit processor ID numbers that are used for the I/O API files.

Checking WPS Output

When running the WPS, it may be helpful to examine the output produced by the programs. For example, when determining the location of nests, it may be helpful to see the interpolated static geographical data and latitude/longitude fields. As another example, when importing a new source of data into WPS – either static data or meteorological data – it can often be helpful to check the resulting interpolated fields in order to make adjustments to the interpolation methods used by geogrid or metgrid.

By using the NetCDF format for the geogrid and metgrid I/O forms, a variety of visualization tools that read NetCDF data may be used to check the domain files

processed by geogrid or the horizontally interpolated meteorological fields produced by metgrid. In order to set the file format for geogrid and metgrid to NetCDF, the user should specify 2 as the `io_form_geogrid` and `io_form_metgrid` in the WPS namelist file (Note: 2 is the default setting for these options):

```
&share
 io_form_geogrid = 2,
/

&metgrid
 io_form_metgrid = 2,
/
```

Among the available tools, the ncdump and ncview programs may be of interest. The ncdump program is a compact utility distributed with the NetCDF libraries that lists the variables and attributes in a NetCDF file. This can be useful, in particular, for checking the domain parameters (e.g., west-east dimension, south-north dimension, or domain center point) in geogrid domain files, or for listing the fields in a file. The ncview program provides an interactive way to view fields in NetCDF files.

Output from the ungrib program is always written in a simple binary format (either 'WPS', 'SI', or 'MM5'), so software for viewing NetCDF files will almost certainly be of no use. However, an NCAR Graphics-based utility, *plotfmt*, is supplied with the WPS source code. This utility produces contour plots of the fields found in an intermediate-format file. If the NCAR Graphics libraries are properly installed, the plotfmt program is automatically compiled, along with other utility programs, when WPS is built.

WPS Utility Programs

Besides the three main WPS programs – geogrid, ungrib, and metgrid – there are a number of utility programs that come with the WPS, and which are compiled in the util directory. These utilities may be used to examine data files, visualize the location of nested domains, compute pressure fields, and compute average surface temperature fields.

A. avg_tsfc.exe

The avg_tsfc.exe program computes a daily mean surface temperature given input files in the intermediate format. Based on the range of dates specified in the "share" namelist section of the namelist.wps file, and also considering the interval between intermediate files, avg_tsfc.exe will use as many complete days' worth of data as possible in computing the average, beginning at the starting date specified in the namelist. If a complete day's worth of data is not available, no output file will be written, and the program will halt as soon as this can be determined. Similarly, any intermediate files for dates that cannot be used as part of a complete 24-hour period are ignored; for example, if there are five intermediate files available at a six-hour interval, the last file would be ignored. The computed average field is written to a new file named TAVGSFC using the same intermediate format version as the input files. This daily mean surface temperature

field can then be ingested by metgrid by specifying 'TAVGSFC' for the `constants_name` variable in the "metgrid" namelist section.

B. mod_levs.exe

The mod_levs.exe program is used to remove levels of data from intermediate format files. The levels which are to be kept are specified in new namelist record in the namelist.wps file:

```
&mod_levs
 press_pa = 201300 , 200100 , 100000 ,
             95000 ,  90000 ,
             85000 ,  80000 ,
             75000 ,  70000 ,
             65000 ,  60000 ,
             55000 ,  50000 ,
             45000 ,  40000 ,
             35000 ,  30000 ,
             25000 ,  20000 ,
             15000 ,  10000 ,
              5000 ,   1000
/
```

Within the `&mod_levs` namelist record, the variable `press_pa` is used to specify a list of levels to keep; the specified levels should match values of `xlvl` in the intermediate format files (see the discussion of the WPS intermediate format for more information on the fields of the intermediate files). The mod_levs program takes two command-line arguments as its input. The first argument is the name of the intermediate file to operate on, and the second argument is the name of the output file to be written.

Removing all but a specified subset of levels from meteorological data sets is particularly useful, for example, when one data set is to be used for the model initial conditions and a second data set is to be used for the lateral boundary conditions. This can be done by providing the initial conditions data set at the first time period to be interpolated by metgrid, and the boundary conditions data set for all other times. If the both data sets have the same number of vertical levels, then no work needs to be done; however, when these two data sets have a different number of levels, it will be necessary, at a minimum, to remove ($m - n$) levels, where $m > n$ and m and n are the number of levels in each of the two data sets, from the data set with m levels. The necessity of having the same number of vertical levels in all files is due to a limitation in `real`, which requires a constant number of vertical levels to interpolate from.

The mod_levs utility is something of a temporary solution to the problem of accommodating two or more data sets with differing numbers of vertical levels. Should a user choose to use mod_levs, it should be noted that, although the vertical locations of the levels need not match between data sets, all data sets should have a surface level of data, and, when running real_nmm.exe and wrf.exe, the value of `p_top` must be chosen to be below the lowest top among the data sets.

C. calc_ecmwf_p.exe

In the course of vertically interpolating meteorological fields, the real program requires 3-d pressure and geopotential height fields on the same levels as the other atmospheric fields. The calc_ecmwf_p.exe utility may be used to create such these fields for use with ECMWF sigma-level data sets. Given a surface pressure field (or log of surface pressure field) and a list of coefficients A and B, calc_ecmwf_p.exe computes the pressure at an ECMWF sigma level *k* at grid point (*i*,*j*) as $P_{ijk} = A_k + B_k * Psfc_{ij}$. The list of coefficients used in the pressure computation can be copied from a table appropriate to the number of sigma levels in the data set from http://old.ecmwf.int/products/data/technical/model_levels/index.html. This table should be written in plain text to a file named ecmwf_coeffs in the current working directory; for example, with 16 sigma levels, the file emcwf_coeffs would contain something like:

```
 0         0.000000      0.000000000
 1      5000.000000      0.000000000
 2      9890.519531      0.001720764
 3     14166.304688      0.013197623
 4     17346.066406      0.042217135
 5     19121.152344      0.093761623
 6     19371.250000      0.169571340
 7     18164.472656      0.268015683
 8     15742.183594      0.384274483
 9     12488.050781      0.510830879
10      8881.824219      0.638268471
11      5437.539063      0.756384850
12      2626.257813      0.855612755
13       783.296631      0.928746223
14         0.000000      0.972985268
15         0.000000      0.992281914
16         0.000000      1.000000000
```

Additionally, if soil height (or soil geopotential), 3-d temperature, and 3-d specific humidity fields are available, calc_ecmwf_p.exe computes a 3-d geopotential height field, which is required to obtain an accurate vertical interpolation in the real program.

Given a set of intermediate files produced by ungrib and the file ecmwf_coeffs, calc_ecmwf_p loops over all time periods in namelist.wps, and produces an additional intermediate file, PRES:*YYYY-MM-DD_HH*, for each time, which contains pressure and geopotential height data for each full sigma level, as well as a 3-d relative humidity field. This intermediate file should be specified to metgrid, along with the intermediate data produced by ungrib, by adding 'PRES' to the list of prefixes in the `fg_name` namelist variable.

D. plotgrids.exe

The plotgrids.exe program is an NCAR Graphics-based utility whose purpose is to plot the locations of all nests defined in the namelist.wps file. The program operates on the namelist.wps file, and thus, may be run without having run any of the three main WPS programs. Upon successful completion, plotgrids produces an NCAR Graphics metafile,

gmeta, which may be viewed using the idt command. The coarse domain is drawn to fill the plot frame, a map outline with political boundaries is drawn over the coarse domain, and any nested domains are drawn as rectangles outlining the extent of each nest. This utility may be useful particularly during initial placement of domains, at which time the user can iteratively adjust the locations of nests by editing the namelist.wps file, running plotgrids.exe, and determining a set of adjustments to the nest locations. *Currently, this utility does not work for ARW domains that use the latitude-longitude projection* (i.e., when `map_proj = 'lat-lon'`).

E. g1print.exe

The g1print.exe program takes as its only command-line argument the name of a GRIB Edition 1 file. The program prints a listing of the fields, levels, and dates of the data in the file.

F. g2print.exe

Similar to g1print.exe, the g2print.exe program takes as its only command-line argument the name of a GRIB Edition 2 file. The program prints a listing of the fields, levels, and dates of the data in the file.

G. plotfmt.exe

The plotfmt.exe is an NCAR Graphics program that plots the contents of an intermediate format file. The program takes as its only command-line argument the name of the file to plot, and produces an NCAR Graphics metafile, which contains contour plots of each field in input file. The graphics metafile output, gmeta, may be viewed with the idt command, or converted to another format using utilities such as ctrans.

H. rd_intermediate.exe

Given the name of a singe intermediate format file on the command line, the rd_intermediate.exe program prints information about the fields contained in the file.

Writing Meteorological Data to the Intermediate Format

The role of the ungrib program is to decode GRIB data sets into a simple intermediate format that is understood by metgrid. If meteorological data are not available in GRIB Edition 1 or GRIB Edition 2 formats, the user is responsible for writing such data into the intermediate file format. Fortunately, the intermediate format is relatively simple, consisting of a sequence of unformatted Fortran writes. It is important to note that these *unformatted writes use big-endian byte order*, which can typically be specified with compiler flags. Below, we describe the WPS intermediate format; users interested in the SI or MM5 intermediate formats can first gain familiarity with the WPS format, which is very similar, and later examine the Fortran subroutines that read and write all three

intermediate formats (metgrid/src/read_met_module.F90 and metgrid/src/write_met_module.F90, respectively).

When writing data to the WPS intermediate format, 2-dimensional fields are written as a rectangular array of real values. 3-dimensional arrays must be split across the vertical dimension into 2-dimensional arrays, which are written independently. It should also be noted that, for global data sets, either a Gaussian or cylindrical equidistant projection must be used, and for regional data sets, either a Mercator, Lambert conformal, polar stereographic, or cylindrical equidistant may be used. The sequence of writes used to write a single 2-dimensional array in the WPS intermediate format is as follows (note that not all of the variables declared below are used for a given projection of the data).

```
integer :: version            ! Format version (must =5 for WPS format)
integer :: nx, ny             ! x- and y-dimensions of 2-d array
integer :: iproj              ! Code for projection of data in array:
                              !       0 = cylindrical equidistant
                              !       1 = Mercator
                              !       3 = Lambert conformal conic
                              !       4 = Gaussian (global only!)
                              !       5 = Polar stereographic
real :: nlats                 ! Number of latitudes north of equator
                              !       (for Gaussian grids)
real :: xfcst                 ! Forecast hour of data
real :: xlvl                  ! Vertical level of data in 2-d array
real :: startlat, startlon    ! Lat/lon of point in array indicated by
                              !       startloc string
real :: deltalat, deltalon    ! Grid spacing, degrees
real :: dx, dy                ! Grid spacing, km
real :: xlonc                 ! Standard longitude of projection
real :: truelat1, truelat2    ! True latitudes of projection
real :: earth_radius          ! Earth radius, km
real, dimension(nx,ny) :: slab ! The 2-d array holding the data
logical :: is_wind_grid_rel   ! Flag indicating whether winds are
                              !       relative to source grid (TRUE) or
                              !       relative to earth (FALSE)
character (len=8)  :: startloc ! Which point in array is given by
                              !       startlat/startlon; set either
                              !       to 'SWCORNER' or 'CENTER  '
character (len=9)  :: field   ! Name of the field
character (len=24) :: hdate   ! Valid date for data YYYY:MM:DD_HH:00:00
character (len=25) :: units   ! Units of data
character (len=32) :: map_source  !  Source model / originating center
character (len=46) :: desc    ! Short description of data


!  1) WRITE FORMAT VERSION
write(unit=ounit) version

!  2) WRITE METADATA
! Cylindrical equidistant
if (iproj == 0) then
      write(unit=ounit) hdate, xfcst, map_source, field, &
                        units, desc, xlvl, nx, ny, iproj
      write(unit=ounit) startloc, startlat, startlon, &
                        deltalat, deltalon, earth_radius
```

```
! Mercator
else if (iproj == 1) then
      write(unit=ounit) hdate, xfcst, map_source, field, &
                        units, desc, xlvl, nx, ny, iproj
      write(unit=ounit) startloc, startlat, startlon, dx, dy, &
                        truelat1, earth_radius

! Lambert conformal
else if (iproj == 3) then
      write(unit=ounit) hdate, xfcst, map_source, field, &
                        units, desc, xlvl, nx, ny, iproj
      write(unit=ounit) startloc, startlat, startlon, dx, dy, &
                        xlonc, truelat1, truelat2, earth_radius

! Gaussian
else if (iproj == 4) then
      write(unit=ounit) hdate, xfcst, map_source, field, &
                        units, desc, xlvl, nx, ny, iproj
      write(unit=ounit) startloc, startlat, startlon, &
                              nlats, deltalon, earth_radius

! Polar stereographic
else if (iproj == 5) then
      write(unit=ounit) hdate, xfcst, map_source, field, &
                        units, desc, xlvl, nx, ny, iproj
      write(unit=ounit) startloc, startlat, startlon, dx, dy, &
                        xlonc, truelat1, earth_radius

end if

!  3) WRITE WIND ROTATION FLAG
write(unit=ounit) is_wind_grid_rel

!  4) WRITE 2-D ARRAY OF DATA
write(unit=ounit) slab
```

Creating and Editing Vtables

Although Vtables are provided for many common data sets, it would be impossible for ungrib to anticipate every possible source of meteorological data in GRIB format. When a new source of data is to be processed by ungrib.exe, the user may create a new Vtable either from scratch, or by using an existing Vtable as an example. In either case, a basic knowledge of the meaning and use of the various fields of the Vtable will be helpful.

Each Vtable contains either seven or eleven fields, depending on whether the Vtable is for a GRIB Edition 1 data source or a GRIB Edition 2 data source, respectively. The fields of a Vtable fall into one of three categories: fields that describe how the data are identified within the GRIB file, fields that describe how the data are identified by the ungrib and metgrid programs, and fields specific to GRIB Edition 2. Each variable to be extracted by ungrib.exe will have one or more lines in the Vtable, with multiple lines for data that are split among different level types – for example, a surface level and upper-air

levels. The fields that must be specified for a line, or entry, in the Vtable depends on the specifics of the field and level.

The first group of fields – those that describe how the data are identified within the GRIB file – are given under the column headings of the Vtable shown below.

```
GRIB1| Level| From |  To  |
Param| Type |Level1|Level2|
-----+------+------+------+
```

The "GRIB1 Param" field specifies the GRIB code for the meteorological field, which is a number unique to that field within the data set. However, different data sets may use different GRIB codes for the same field – for example, temperature at upper-air levels has GRIB code 11 in GFS data, but GRIB code 130 in ECMWF data. To find the GRIB code for a field, the g1print.exe and g2print.exe utility program may be used.

Given a GRIB code, the "Level Type", "From Level1", and "To Level2" fields are used to specify the levels at which levels a field may be found. As with the "GRIB1 Param" field, the g1print.exe and g2print.exe programs may be used to find the GRIB code values for the "Level Type", "From Level1", and "From Level2" fields. The "Level Type" field is a GRIB code that defines the level(s) at which a particular field may be found, and its values are summarized in the following table. The "From Level1" and "To Level2" fields give the lower and upper bounds of the levels where appropriate.

Level	Level Type	From Level1	To Level2
Upper-air	100	*	(blank)
Surface	1	0	(blank)
Sea-level	102	0	(blank)
Levels at a specified height AGL	105	Height, in meters, of the level above ground	(blank)
Fields given as layers	112	Starting level for the layer	Ending level for the layer

When layer fields (Level Type 112) are specified, the starting and ending points for the layer have units that are dependent on the field itself; appropriate values may be found with the g1print.exe and g2print.exe utility programs.

The second group of fields in a Vtable, those that describe how the data are identified within the metgrid and real programs, fall under the column headings shown below.

```
| metgrid  | metgrid | metgrid                                  |
| Name     |  Units  | Description                              |
+----------+---------+------------------------------------------+
```

The most important of these three fields is the "metgrid Name" field, which determines the variable name that will be assigned to a meteorological field when it is written to the intermediate files by ungrib. This name should also match an entry in the METGRID.TBL file, so that the metgrid program can determine how the field is to be horizontally interpolated. The "metgrid Units" and "metgrid Description" fields specify the units and a short description for the field, respectively; here, it is important to note that if no description is given for a field, then *ungrib will not write that field out to the intermediate files*.

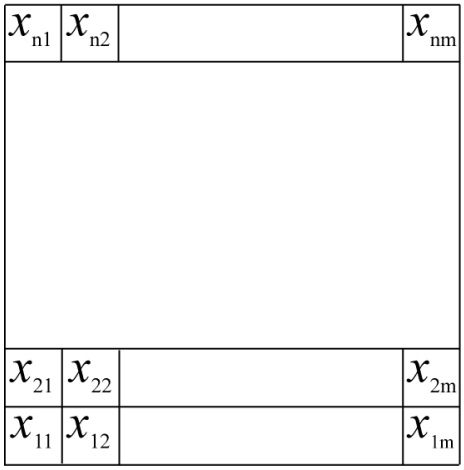
The final group of fields, which provide GRIB2-specific information, are found under the column headings below.

```
|GRIB2|GRIB2|GRIB2|GRIB2|
|Discp|Catgy|Param|Level|
+---------------------+
```

The GRIB2 fields are only needed in a Vtable that is to be used for GRIB Edition 2 data sets, although having these fields in a Vtable does not prevent that Vtable from also being used for GRIB Edition 1 data. For example, the Vtable.GFS file contains GRIB2 Vtable fields, but is used for both 1-degree (GRIB1) GFS and 0.5-degree (GRIB2) GFS data sets. Since Vtables are provided for most known GRIB Edition 2 data sets, the corresponding Vtable fields are not described here at present.

Writing Static Data to the Geogrid Binary Format

The static geographical data sets that are interpolated by the geogrid program are stored as regular 2-d and 3-d arrays written in a simple binary raster format. Users with a new source for a given static field can ingest their data with WPS by writing the data set into this binary format. The geogrid format is capable of supporting single-level and multi-level continuous fields, categorical fields represented as dominant categories, and categorical fields given as fractional fields for each category. The most simple of these field types in terms of representation in the binary format is a categorical field given as a dominant category at each source grid point, an example of which is the 30-second USGS land use data set.



For a categorical field given as dominant categories, the data must first be stored in a regular 2-d array of integers, with each integer giving the dominant category at the corresponding source grid point. Given this array, the data are written to a file, row-by-row, beginning at the bottom, or southern-most, row. For example, in the figure above, the elements of the $n \times m$ array would be written in the order x_{11}, x_{12}, ..., x_{1m}, x_{21}, ..., x_{2m}, ..., x_{n1}, ..., x_{nm}. When written to the file, every element is stored as a 1-, 2-, 3-, or 4-byte integer in big-endian byte order (i.e., for the 4-byte integer *ABCD*, byte *A* is stored at the lowest address and byte *D* at the highest), although little-endian files may be used by setting `endian=little` in the "index" file for the data set. Every element in a file must use the same number of bytes for its storage, and, of course, it is advantageous to use the fewest number of bytes needed to represent the complete range of values in the array.

When writing the binary data to a file, no header, record marker, or additional bytes should be written. For example, a 2-byte 1000×1000 array should result in a file whose size is exactly 2,000,000 bytes. Since Fortran unformatted writes add record markers, *it is not possible to write a geogrid binary-formatted file directly from Fortran*; instead, it is recommended that the C routines in read_geogrid.c and write_geogrid.c (in the geogrid/src directory) be called when writing data, either from C or Fortran code.

Similar in format to a field of dominant categories is the case of a field of continuous, or real, values. Like dominant-category fields, single-level continuous fields are first organized as a regular 2-d array, then written, row-by-row, to a binary file. However, because a continuous field may contain non-integral or negative values, the storage representation of each element within the file is slightly more complex. All elements in the array must first be converted to integral values. This is done by first scaling all elements by a constant, chosen to maintain the required precision, and then removing any remaining fractional part through rounding. For example, if three decimal places of precision are required, the value -2.71828 would need to be divided by 0.001 and rounded to -2718. Following conversion of all array elements to integral values, if any negative values are found in the array, a second conversion must be applied: if elements are stored using 1 byte each, then 2^8 is added to each negative element; for storage using 2 bytes, 2^{16} is added to each negative element; for storage using 3 bytes, 2^{24} is added to each negative element; and for storage using 4 bytes, a value of 2^{32} is added to each

negative element. It is important to note that no conversion is applied to positive elements. Finally, the resulting positive, integral array is written as in the case of a dominant-category field.

Multi-level continuous fields are handled much the same as single-level continuous fields. For an $n \times m \times r$ array, conversion to a positive, integral field is first performed as described above. Then, each $n \times m$ sub-array is written contiguously to the binary file as before, beginning with the smallest r-index. Categorical fields that are given as fractional fields for each possible category can be thought of as multi-level continuous fields, where each level k, $1 \leq k \leq r$, is the fractional field for category k.

When writing a field to a file in the geogrid binary format, the user should adhere to the naming convention used by the geogrid program, which expects data files to have names of the form *xstart-xend.ystart-yend*, where *xstart*, *xend*, *ystart*, and *yend* are five-digit positive integers specifying, respectively, the starting x-index of the array contained in the file, the ending x-index of the array, the starting y-index of the array, and the ending y-index of the array; here, indexing begins at 1, rather than 0. So, for example, an 800 × 1200 array (i.e., 800 rows and 1200 columns) might be named 00001-01200.00001-00800.

When a data set is given in several pieces, each of the pieces may be formed as a regular rectangular array, and each array may be written to a separate file. In this case, the relative locations of the arrays are determined by the range of x- and y-indices in the file names for each of the arrays. It is important to note, however, that *every tile in a data set must have the same* x- *and* y-*dimensions*, and that tiles of data within a data set must not overlap; furthermore, all tiles must start and end on multiples of the index ranges. For example, the global 30-second USGS topography data set is divided into arrays of dimension 1200 × 1200, with each array containing a 10-degree × 10-degree piece of the data set; the file whose south-west corner is located at (90S, 180W) is named 00001-01200.00001-01200, and the file whose north-east corner is located at (90N, 180E) is named 42001-43200.20401-21600.

If a data set is to be split into multiple tiles, and the number of grid points in, say, the x-direction is not evenly divided by the number of tiles in the x-direction, then the last column of tiles must be padded with a flag value (specified in the index file using the `missing_value` keyword) so that all tiles have the same dimensions. For example, if a data set has 2456 points in the x-direction, and three tiles in the x-direction will be used, the range of x-coordinates of the tiles might be 1 – 820, 821 – 1640, and 1641 – 2460, with columns 2457 through 2460 being filled with a flag value.

Clearly, since the starting and ending indices must have five digits, a field cannot have more than 99999 data points in either of the x- or y-directions. In case a field has more than 99999 data points in either dimension, the user can simply split the data set into several smaller data sets which will be identified separately to geogrid. For example, a very large global data set may be split into data sets for the Eastern and Western hemispheres.

Besides the binary data files, geogrid requires one extra metadata file per data set. This metadata file is always named 'index', and thus, two data sets cannot reside in the same directory. Essentially, this metadata file is the first file that geogrid looks for when processing a data set, and the contents of the file provide geogrid with all of the information necessary for constructing names of possible data files. The contents of an example index file are given below.

```
type = continuous
signed = yes
projection = regular_ll
dx = 0.00833333
dy = 0.00833333
known_x = 1.0
known_y = 1.0
known_lat = -89.99583
known_lon = -179.99583
wordsize = 2
tile_x = 1200
tile_y = 1200
tile_z = 1
tile_bdr=3
units="meters MSL"
description="Topography height"
```

For a complete listing of keywords that may appear in an index file, along with the meaning of each keyword, the user is referred to the section on index file options.

Description of the Namelist Variables

A. SHARE section

This section describes variables that are used by more than one WPS program. For example, the `wrf_core` variable specifies whether the WPS is to produce data for the ARW or the NMM core – information which is needed by both the geogrid and metgrid programs.

1. WRF_CORE : A character string set to either `'ARW'` or `'NMM'` that tells the WPS which dynamical core the input data are being prepared for. Default value is `'ARW'`.

2. MAX_DOM : An integer specifying the total number of domains/nests, including the parent domain, in the simulation. Default value is 1.

3. START_YEAR : A list of MAX_DOM 4-digit integers specifying the starting UTC year of the simulation for each nest. No default value.

4. START_MONTH : A list of MAX_DOM 2-digit integers specifying the starting UTC month of the simulation for each nest. No default value.

5. START_DAY : A list of MAX_DOM 2-digit integers specifying the starting UTC day of the simulation for each nest. No default value.

6. START_HOUR : A list of MAX_DOM 2-digit integers specifying the starting UTC hour of the simulation for each nest. No default value.
7. END_YEAR : A list of MAX_DOM 4-digit integers specifying the ending UTC year of the simulation for each nest. No default value.

8. END_MONTH : A list of MAX_DOM 2-digit integers specifying the ending UTC month of the simulation for each nest. No default value.

9. END_DAY : A list of MAX_DOM 2-digit integers specifying the ending UTC day of the simulation for each nest. No default value.

10. END_HOUR : A list of MAX_DOM 2-digit integers specifying the ending UTC hour of the simulation for each nest. No default value.

11. START_DATE : A list of MAX_DOM character strings of the form `'YYYY-MM-DD_HH:mm:ss'` specifying the starting UTC date of the simulation for each nest. The `start_date` variable is an alternate to specifying `start_year`, `start_month`, `start_day`, and `start_hour`, and if both methods are used for specifying the starting time, the `start_date` variable will take precedence. No default value.

12. END_DATE : A list of MAX_DOM character strings of the form `'YYYY-MM-DD_HH:mm:ss'` specifying the ending UTC date of the simulation for each nest. The `end_date` variable is an alternate to specifying `end_year`, `end_month`, `end_day`, and `end_hour`, and if both methods are used for specifying the ending time, the `end_date` variable will take precedence. No default value.

13. INTERVAL_SECONDS : The integer number of seconds between time-varying meteorological input files. No default value.

14. ACTIVE_GRID : A list of MAX_DOM logical values specifying, for each grid, whether that grid should be processed by geogrid and metgrid. Default value is `.TRUE.`.

15. IO_FORM_GEOGRID : The WRF I/O API format that the domain files created by the geogrid program will be written in. Possible options are: 1 for binary; 2 for NetCDF; 3 for GRIB1. When option 1 is given, domain files will have a suffix of .int; when option 2 is given, domain files will have a suffix of .nc; when option 3 is given, domain files will have a suffix of .gr1. Default value is 2 (NetCDF).

16. OPT_OUTPUT_FROM_GEOGRID_PATH : A character string giving the path, either relative or absolute, to the location where output files from geogrid should be written to and read from. Default value is `'./'`.

17. DEBUG_LEVEL : An integer value indicating the extent to which different types of messages should be sent to standard output. When `debug_level` is set to 0, only generally useful messages and warning messages will be written to standard output. When `debug_level` is greater than 100, informational messages that provide further runtime details are also written to standard output. Debugging messages and messages specifically intended for log files are never written to standard output, but are always written to the log files. Default value is 0.

B. GEOGRID section

This section specifies variables that are specific to the geogrid program. Variables in the geogrid section primarily define the size and location of all model domains, and where the static geographical data are found.

1. PARENT_ID : A list of MAX_DOM integers specifying, for each nest, the domain number of the nest's parent; for the coarsest domain, this variable should be set to 1. Default value is 1.

2. PARENT_GRID_RATIO : A list of MAX_DOM integers specifying, for each nest, the nesting ratio relative to the domain's parent. **This must be set to 3 for WRF-NMM.** No default value.

3. I_PARENT_START : A list of MAX_DOM integers specifying, for each nest, the x-coordinate of the lower-left corner of the nest in the parent *unstaggered* grid. For the coarsest domain, a value of 1 should be specified. No default value. **For WRF-NMM nests, see note on page 2-15.**

4. J_PARENT_START : A list of MAX_DOM integers specifying, for each nest, the y-coordinate of the lower-left corner of the nest in the parent *unstaggered* grid. For the coarsest domain, a value of 1 should be specified. No default value. **For WRF-NMM nests, see note on page 2-15.**

5. S_WE : A list of MAX_DOM integers which should all be set to 1. Default value is 1. **For WRF-NMM nests, see note on page 2-15.**

6. E_WE : A list of MAX_DOM integers specifying, for each nest, the nest's full west-east dimension. For nested domains, `e_we` must be one greater than an integer multiple of the nest's `parent_grid_ratio` (i.e., `e_ew` = *n**`parent_grid_ratio`+1 for some positive integer *n*). No default value. **For WRF-NMM nests, E_WE is the number of mas points in odd numbered rows; see also note on page 2-15.**

7. S_SN : A list of MAX_DOM integers which should all be set to 1. Default value is 1. **For WRF-NMM nests, see note on page 2-15.**

8. E_SN : A list of MAX_DOM integers specifying, for each nest, the nest's full south-north dimension. For nested domains, `e_sn` must be one greater than an integer multiple of the nest's `parent_grid_ratio` (i.e., `e_sn` = *n**`parent_grid_ratio`+1 for some positive integer *n*). No default value. **For WRF-NMM nests, E_SN is the number of rows; see also note on page 2-15.**

Note: For WRF-NMM, the schematic below illustrates how ***e_we*** and ***e_sn*** apply on the E-grid:

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)

In this schematic, H represents mass variables (e.g., temperature, pressure, moisture) and V represents vector wind quantities. The (H) and (V) at the end of the row are a so-called phantom column that is used so arrays will be completely filled (***e_we***, ***e_sn***) for both mass and wind quantities, but the phantom column does not impact the integration. In this example, the x-dimension of the computational grid is 4, wheras the y-dimension is 5.By definition, ***e_we*** and ***e_sn*** are one plus the computational grid, such that, for this example, ***e_we***=5 and ***e_sn***=6. Note, also, that the number of computational rows must be odd, so the value for ***e_sn*** must always be EVEN.

9. GEOG_DATA_RES : A list of MAX_DOM character strings specifying, for each nest, a corresponding resolution or list of resolutions separated by + symbols of source data to be used when interpolating static terrestrial data to the nest's grid. For each nest, this string should contain a resolution matching a string preceding a colon in a `rel_path` or `abs_path` specification (see the description of GEOGRID.TBL options) in the GEOGRID.TBL file for each field. If a resolution in the string does not match any such string in a `rel_path` or `abs_path` specification for a field in GEOGRID.TBL, a default resolution of data for that field, if one is specified, will be used. If multiple resolutions match, the first resolution to match a string in a `rel_path` or `abs_path` specification in the GEOGRID.TBL file will be used. Default value is `'default'`.

10. DX : A real value specifying the grid distance in the x-direction where the map scale factor is 1. For ARW, the grid distance is in meters for the `'polar'`, `'lambert'`, and `'mercator'` projection, and in degrees longitude for the `'lat-lon'` projection; for NMM, the grid distance is in degrees longitude. Grid distances for nests are determined recursively based on values specified for `parent_grid_ratio` and `parent_id`. No default value.

11. DY : A real value specifying the nominal grid distance in the y-direction where the map scale factor is 1. For ARW, the grid distance is in meters for the `'polar'`,

'lambert', and 'mercator' projection, and in degrees latitude for the 'lat-lon' projection; for NMM, the grid distance is in degrees latitude. Grid distances for nests are determined recursively based on values specified for parent_grid_ratio and parent_id. No default value.

Note: For the rotated latitude-longitude grid used by WRF-NMM, the grid center is the equator. ***DX*** and ***DY*** are constant within this rotated grid framework. However, in a true Earth sense, the grid spacing in kilometers varies slightly between the center latitude and the northern and southern edges due to convergence of meridians away from the equator. This behavior is more notable for domains covering a wide range of latitudes. Typically, ***DX*** is set to be slightly larger than ***DY*** to counter the effect of meridional convergence, and keep the unrotated, "true earth" grid spacing more uniform over the entire grid.

The relationship between the fraction of a degree specification for the E-grid and the more typical grid spacing specified in kilometers for other grids can be approximated by considering the following schematic:

V -DX- H
| / |
DY dx DY
| / |
H - DX- V

The horizontal grid resolution is taken to be the shortest distance between two mass (H) points (diagonal – dx), while ***DX*** and ***DY*** refer to distances between adjacent H and V points. The distance between the H points in the diagram above is the hypotenuse of the triangle with legs DX and DY. Assuming 111 km/degree (a reasonable assumption for the rotated latitude-longitude grid) the grid spacing in km is approximately equal to: 111.0 * (SQRT (DX**2 + DY**2)).

12. MAP_PROJ : A character string specifying the projection of the simulation domain. For ARW, accepted projections are 'lambert', 'polar', 'mercator', and 'lat-lon'; for NMM, a projection of 'rotated_ll' must be specified. Default value is 'lambert'.

13. REF_LAT : A real value specifying the latitude part of a (latitude, longitude) location whose (i,j) location in the simulation domain is known. For ARW, ref_lat gives the latitude of the center-point of the coarse domain by default (i.e., when ref_x and ref_y are not specified). For NMM, ref_lat always gives the latitude to which the origin is rotated. No default value.

14. REF_LON : A real value specifying the longitude part of a (latitude, longitude) location whose (i, j) location in the simulation domain is known. For ARW, ref_lon gives the longitude of the center-point of the coarse domain by default (i.e., when ref_x and ref_y are not specified). For NMM, ref_lon always gives the longitude to which the origin is rotated. For both ARW and NMM, west longitudes are negative, and the value of ref_lon should be in the range [-180, 180]. No default value.

15. REF_X : A real value specifying the i part of an (i, j) location whose (latitude, longitude) location in the simulation domain is known. The (i, j) location is always given with respect to the mass-staggered grid, whose dimensions are one less than the dimensions of the unstaggered grid. Default value is (((`E_WE`-1.)+1.)/2.) = (`E_WE`/2.).

16. REF_Y : A real value specifying the j part of an (i, j) location whose (latitude, longitude) location in the simulation domain is known. The (i, j) location is always given with respect to the mass-staggered grid, whose dimensions are one less than the dimensions of the unstaggered grid. Default value is (((`E_SN`-1.)+1.)/2.) = (`E_SN`/2.).

17. TRUELAT1 : A real value specifying, for ARW, the first true latitude for the Lambert conformal projection, or the only true latitude for the Mercator and polar stereographic projections. For NMM, `truelat1` is ignored. No default value.

18. TRUELAT2 : A real value specifying, for ARW, the second true latitude for the Lambert conformal conic projection. For all other projections, `truelat2` is ignored. No default value.

19. STAND_LON : A real value specifying, for ARW, the longitude that is parallel with the y-axis in the Lambert conformal and polar stereographic projections. For the regular latitude-longitude projection, this value gives the rotation about the earth's geographic poles. For NMM, `stand_lon` is ignored. No default value.

20. POLE_LAT : For the latitude-longitude projection for ARW, the latitude of the North Pole with respect to the computational latitude-longitude grid in which -90.0° latitude is at the bottom of a global domain, 90.0° latitude is at the top, and 180.0° longitude is at the center. Default value is 90.0.

21. POLE_LON : For the latitude-longitude projection for ARW, the longitude of the North Pole with respect to the computational lat/lon grid in which -90.0° latitude is at the bottom of a global domain, 90.0° latitude is at the top, and 180.0° longitude is at the center. Default value is 0.0.

22. GEOG_DATA_PATH : A character string giving the path, either relative or absolute, to the directory where the geographical data directories may be found. This path is the one to which `rel_path` specifications in the GEOGRID.TBL file are given in relation to. No default value.

23. OPT_GEOGRID_TBL_PATH : A character string giving the path, either relative or absolute, to the GEOGRID.TBL file. The path should not contain the actual file name, as GEOGRID.TBL is assumed, but should only give the path where this file is located. Default value is `'./geogrid/'`.

C. UNGRIB section

Currently, this section contains only two variables, which determine the output format written by ungrib and the name of the output files.

1. OUT_FORMAT : A character string set either to `'WPS'`, `'SI'`, or `'MM5'`. If set to `'MM5'`, ungrib will write output in the format of the MM5 pregrid program; if set to `'SI'`, ungrib will write output in the format of grib_prep.exe; if set to `'WPS'`, ungrib will write data in the WPS intermediate format. Default value is `'WPS'`.

2. PREFIX : A character string that will be used as the prefix for intermediate-format files created by ungrib; here, prefix refers to the string *PREFIX* in the filename *PREFIX*:*YYYY-MM-DD_HH* of an intermediate file. The prefix may contain path information, either relative or absolute, in which case the intermediate files will be written in the directory specified. This option may be useful to avoid renaming intermediate files if ungrib is to be run on multiple sources of GRIB data. Default value is `'FILE'`.

D. METGRID section

This section defines variables used only by the metgrid program. Typically, the user will be interested in the `fg_name` variable, and may need to modify other variables of this section less frequently.

1. FG_NAME : A list of character strings specifying the path and prefix of ungribbed data files. The path may be relative or absolute, and the prefix should contain all characters of the filenames up to, but not including, the colon preceding the date. When more than one `fg_name` is specified, and the same field is found in two or more input sources, the data in the last encountered source will take priority over all preceding sources for that field. Default value is an empty list (i.e., no meteorological fields).

2. CONSTANTS_NAME : A list of character strings specifying the path and full filename of ungribbed data files which are time-invariant. The path may be relative or absolute, and the filename should be the complete filename; since the data are assumed to be time-invariant, no date will be appended to the specified filename. Default value is an empty list (i.e., no constant fields).

3. IO_FORM_METGRID : The WRF I/O API format that the output created by the metgrid program will be written in. Possible options are: 1 for binary; 2 for NetCDF; 3 for GRIB1. When option 1 is given, output files will have a suffix of .int; when option 2 is given, output files will have a suffix of .nc; when option 3 is given, output files will have a suffix of .gr1. Default value is 2 (NetCDF).

4. OPT_OUTPUT_FROM_METGRID_PATH : A character string giving the path, either relative or absolute, to the location where output files from metgrid should be written to. The default value is the current working directory (i.e., the default value is `'./'`).

5. OPT_METGRID_TBL_PATH : A character string giving the path, either relative or absolute, to the METGRID.TBL file; the path should not contain the actual file name, as METGRID.TBL is assumed, but should only give the path where this file is located. Default value is `'./metgrid/'`.

6. OPT_IGNORE_DOM_CENTER : A logical value, either `.TRUE.` or `.FALSE.`, specifying whether, for times other than the initial time, interpolation of meteorological fields to points on the interior of the simulation domain should be avoided in order to decrease the runtime of metgrid. This option currently has no effect. Default value is `.FALSE.`.

Description of GEOGRID.TBL Options

The GEOGRID.TBL file is a text file that defines parameters of each of the data sets to be interpolated by geogrid. Each data set is defined in a separate section, with sections being delimited by a line of equality symbols (e.g., '==========='). Within each section, there are specifications, each of which has the form of *keyword=value*. Some keywords are required in each data set section, while others are optional; some keywords are mutually exclusive with other keywords. Below, the possible keywords and their expected range of values are described.

1. NAME : A character string specifying the name that will be assigned to the interpolated field upon output. No default value.

2. PRIORITY : An integer specifying the priority that the data source identified in the table section takes with respect to other sources of data for the same field. If a field has *n* sources of data, then there must be *n* separate table entries for the field, each of which must be given a unique value for `priority` in the range [1, *n*]. No default value.

3. DEST_TYPE : A character string, either `categorical` or `continuous`, that tells whether the interpolated field from the data source given in the table section is to be treated as a continuous or a categorical field. No default value.

4. INTERP_OPTION : A sequence of one or more character strings, which are the names of interpolation methods to be used when horizontally interpolating the field. Available interpolation methods are: `average_4pt`, `average_16pt`, `wt_average_4pt`, `wt_average_16pt`, `nearest_neighbor`, `four_pt`, `sixteen_pt`, `search`, `average_gcell(`*r*`)`; for the grid cell average method (`average_gcell`), the optional argument *r* specifies the minimum ratio of source data resolution to simulation grid resolution at which the method will be applied; if a ratio is not specified, *r* = 0.0, and the option is used for any ratio. When a sequence of two or more methods are given, the methods should be separated by a + sign. No default value.

5. SMOOTH_OPTION : A character string giving the name of a smoothing method to be applied to the field after interpolation. Available smoothing options are: `1-2-1`, `smth-desmth`, and `smth-desmth_special` (ARW only). Default value is null (i.e., no smoothing is applied).

6. SMOOTH_PASSES : If smoothing is to be performed on the interpolated field, `smooth_passes` specifies an integer number of passes of the smoothing method to apply to the field. Default value is 1.

7. REL_PATH : A character string specifying the path relative to the path given in the namelist variable `geog_data_path`. A specification is of the general form *RES_STRING*:*REL_PATH*, where *RES_STRING* is a character string identifying the source or resolution of the data in some unique way and may be specified in the namelist variable `geog_data_res`, and *REL_PATH* is a path relative to `geog_data_path` where the index and data tiles for the data source are found. More than one `rel_path` specification may be given in a table section if there are multiple sources or resolutions for the data source, just as multiple resolutions may be specified (in a sequence delimited by + symbols) for `geog_data_res`. See also `abs_path`. No default value.

8. ABS_PATH : A character string specifying the absolute path to the index and data tiles for the data source. A specification is of the general form *RES_STRING*:*ABS_PATH*, where *RES_STRING* is a character string identifying the source or resolution of the data in some unique way and may be specified in the namelist variable `geog_data_res`, and *ABS_PATH* is the absolute path to the data source's files. More than one `abs_path` specification may be given in a table section if there are multiple sources or resolutions for the data source, just as multiple resolutions may be specified (in a sequence delimited by + symbols) for `geog_data_res`. See also `rel_path`. No default value.

9. OUTPUT_STAGGER : A character string specifying the grid staggering to which the field is to be interpolated. For ARW domains, possible values are `U`, `V`, and `M`; for NMM domains, possible values are `HH` and `VV`. Default value for ARW is `M`; default value for NMM is `HH`.

10. LANDMASK_WATER : One or more comma-separated integer values giving the indices of the categories within the field that represents water. When `landmask_water` is specified in the table section of a field for which `dest_type=categorical`, the LANDMASK field will be computed from the field using the specified categories as the water categories. The keywords `landmask_water` and `landmask_land` are mutually exclusive. Default value is null (i.e., a landmask will not be computed from the field).

11. LANDMASK_LAND : One or more comma-separated integer values giving the indices of the categories within the field that represents land. When `landmask_water` is specified in the table section of a field for which `dest_type=categorical`, the LANDMASK field will be computed from the field using the specified categories as the

land categories. The keywords `landmask_water` and `landmask_land` are mutually exclusive. Default value is null (i.e., a landmask will not be computed from the field).

12. MASKED : Either `land` or `water`, indicating that the field is not valid at land or water points, respectively. If the `masked` keyword is used for a field, those grid points that are of the masked type (land or water) will be assigned the value specified by `fill_missing`. Default value is null (i.e., the field is not masked).

13. FILL_MISSING : A real value used to fill in any missing or masked grid points in the interpolated field. Default value is 1.E20.

14. HALT_ON_MISSING : Either `yes` or `no`, indicating whether geogrid should halt with a fatal message when a missing value is encountered in the interpolated field. Default value is `no`.

15. DOMINANT_CATEGORY : When specified as a character string, the effect is to cause geogrid to compute the dominant category from the fractional categorical field, and to output the dominant category field with the name specified by the value of `dominant_category`. This option can only be used for fields with `dest_type=categorical`. Default value is null (i.e., no dominant category will be computed from the fractional categorical field).

16. DOMINANT_ONLY : When specified as a character string, the effect is similar to that of the `dominant_category` keyword: geogrid will compute the dominant category from the fractional categorical field and output the dominant category field with the name specified by the value of `dominant_only`. Unlike with `dominant_category`, though, when `dominant_only` is used, the fractional categorical field will not appear in the geogrid output. This option can only be used for fields with `dest_type=categorical`. Default value is null (i.e., no dominant category will be computed from the fractional categorical field).

17. DF_DX : When `df_dx` is assigned a character string value, the effect is to cause geogrid to compute the directional derivative of the field in the x-direction using a central difference along the interior of the domain, or a one-sided difference at the boundary of the domain; the derivative field will be named according to the character string assigned to the keyword `df_dx`. Default value is null (i.e., no derivative field is computed).

18. DF_DY : When `df_dy` is assigned a character string value, the effect is to cause geogrid to compute the directional derivative of the field in the y-direction using a central difference along the interior of the domain, or a one-sided difference at the boundary of the domain; the derivative field will be named according to the character string assigned to the keyword `df_dy`. Default value is null (i.e., no derivative field is computed).

19. Z_DIM_NAME : For 3-dimensional output fields, a character string giving the name of the vertical dimension, or z-dimension. A continuous field may have multiple levels, and thus be a 3-dimensional field, and a categorical field may take the form of a 3-

dimensional field if it is written out as fractional fields for each category. No default value.

Description of index Options

Related to the GEOGRID.TBL are the index files that are associated with each static data set. An index file defines parameters specific to that data set, while the GEOGRID.TBL file describes how each of the data sets should be treated by geogrid. As with the GEOGRID.TBL file, specifications in an index file are of the form *keyword=value*. Below are possible keywords and their possible values.

1. PROJECTION : A character string specifying the projection of the data, which may be either `lambert`, `polar`, `mercator`, `regular_ll`, `albers_nad83`, or `polar_wgs84`. No default value.

2. TYPE : A character string, either `categorical` or `continuous`, that determines whether the data in the data files should be interpreted as a continuous field or as discrete indices. For categorical data represented by a fractional field for each possible category, `type` should be set to `continuous`. No default value.

3. SIGNED : Either `yes` or `no`, indicating whether the values in the data files (which are always represented as integers) are signed in two's complement form or not. Default value is `no`.

4. UNITS : A character string, enclosed in quotation marks ("), specifying the units of the interpolated field; the string will be written to the geogrid output files as a variable time-independent attribute. No default value.

5. DESCRIPTION : A character string, enclosed in quotation marks ("), giving a short description of the interpolated field; the string will be written to the geogrid output files as a variable time-independent attribute. No default value.

6. DX : A real value giving the grid spacing in the x-direction of the data set. If `projection` is one of `lambert`, `polar`, `mercator`, `albers_nad83`, or `polar_wgs84`, `dx` gives the grid spacing in meters; if `projection` is `regular_ll`, `dx` gives the grid spacing in degrees. No default value.

7. DY : A real value giving the grid spacing in the y-direction of the data set. If `projection` is one of `lambert`, `polar`, `mercator`, `albers_nad83`, or `polar_wgs84`, `dy` gives the grid spacing in meters; if `projection` is `regular_ll`, `dy` gives the grid spacing in degrees. No default value.

8. KNOWN_X : A real value specifying the i-coordinate of an (i,j) location corresponding to a (latitude, longitude) location that is known in the projection. Default value is 1.

9. KNOWN_Y : A real value specifying the j-coordinate of an (i,j) location corresponding to a (latitude, longitude) location that is known in the projection. Default value is 1.

10. KNOWN_LAT : A real value specifying the latitude of a (latitude, longitude) location that is known in the projection. No default value.

11. KNOWN_LON : A real value specifying the longitude of a (latitude, longitude) location that is known in the projection. No default value.

12. STDLON : A real value specifying the longitude that is parallel with the y-axis in conic and azimuthal projections. No default value.

13. TRUELAT1 : A real value specifying the first true latitude for conic projections or the only true latitude for azimuthal projections. No default value.

14. TRUELAT2 : A real value specifying the second true latitude for conic projections. No default value.

15. WORDSIZE : An integer giving the number of bytes used to represent the value of each grid point in the data files. No default value.

16. TILE_X : An integer specifying the number of grid points in the x-direction, *excluding any halo points*, for a single tile of source data. No default value.

17. TILE_Y : An integer specifying the number of grid points in the y-direction, *excluding any halo points*, for a single tile of source data. No default value.

18. TILE_Z : An integer specifying the number of grid points in the z-direction for a single tile of source data; this keyword serves as an alternative to the pair of keywords `tile_z_start` and `tile_z_end`, and when this keyword is used, the starting z-index is assumed to be 1. No default value.

19. TILE_Z_START : An integer specifying the starting index in the z-direction of the array in the data files. If this keyword is used, `tile_z_end` must also be specified. No default value.

20. TILE_Z_END : An integer specifying the ending index in the z-direction of the array in the data files. If this keyword is used, `tile_z_start` must also be specified. No default value

21. CATEGORY_MIN : For categorical data (`type=categorical`), an integer specifying the minimum category index that is found in the data set. If this keyword is used, `category_max` must also be specified. No default value.

22. CATEGORY_MAX : For categorical data (`type=categorical`), an integer specifying the maximum category index that is found in the data set. If this keyword is used, `category_min` must also be specified. No default value.

23. TILE_BDR : An integer specifying the halo width, in grid points, for each tile of data. Default value is 0.

24. MISSING_VALUE : A real value that, when encountered in the data set, should be interpreted as missing data. No default value.

25. SCALE_FACTOR : A real value that data should be scaled by (through multiplication) after being read in as integers from tiles of the data set. Default value is 1.

26. ROW_ORDER : A character string, either `bottom_top` or `top_bottom`, specifying whether the rows of the data set arrays were written proceeding from the lowest-index row to the highest (`bottom_top`) or from highest to lowest (`top_bottom`). This keyword may be useful when utilizing some USGS data sets, which are provided in `top_bottom` order. Default value is `bottom_top`.

27. ENDIAN : A character string, either `big` or `little`, specifying whether the values in the static data set arrays are in big-endian or little-endian byte order. Default value is `big`.

28. ISWATER : An integer specifying the land use category of water. Default value is 16.

29. ISLAKE : An integer specifying the land use category of inland water bodies. Default value is -1 (i.e., no separate inland water category).

30. ISICE : An integer specifying the land use category of ice. Default value is 24.

31. ISURBAN : An integer specifying the land use category of urban areas. Default value is 1.

32. ISOILWATER : An integer specifying the soil category of water. Default value is 14.

33. MMINLU : A character string, enclosed in quotation marks ("), indicating which section of WRF's LANDUSE.TBL and VEGPARM.TBL will be used when looking up parameters for land use categories. Default value is `"USGS"`.

Description of METGRID.TBL Options

The METGRID.TBL file is a text file that defines parameters of each of the meteorological fields to be interpolated by metgrid. Parameters for each field are defined in a separate section, with sections being delimited by a line of equality symbols (e.g., '==========='). Within each section, there are specifications, each of which has the form of *keyword=value*. Some keywords are required in a section, while others are

optional; some keywords are mutually exclusive with other keywords. Below, the possible keywords and their expected range of values are described.

1. NAME : A character string giving the name of the meteorological field to which the containing section of the table pertains. The name should exactly match that of the field as given in the intermediate files (and, thus, the name given in the Vtable used in generating the intermediate files). This field is required. No default value.

2. OUTPUT : Either `yes` or `no`, indicating whether the field is to be written to the metgrid output files or not. Default value is `yes`.

3. MANDATORY : Either `yes` or `no`, indicating whether the field is required for successful completion of metgrid. Default value is `no`.

4. OUTPUT_NAME : A character string giving the name that the interpolated field should be output as. When a value is specified for `output_name`, the interpolation options from the table section pertaining to the field with the specified name are used. Thus, the effects of specifying `output_name` are two-fold: The interpolated field is assigned the specified name before being written out, and the interpolation methods are taken from the section pertaining to the field whose name matches the value assigned to the `output_name` keyword. No default value.

5. FROM_INPUT : A character string used to compare against the values in the `fg_name` namelist variable; if `from_input` is specified, the containing table section will only be used when the time-varying input source has a filename that contains the value of `from_input` as a substring. Thus, `from_input` may be used to specify different interpolation options for the same field, depending on which source of the field is being processed. No default value.

6. OUTPUT_STAGGER : The model grid staggering to which the field should be interpolated. For ARW, this must be one of `U`, `V`, and `M`; for NMM, this must be one of `HH` and `VV`. Default value for ARW is `M`; default value for NMM is `HH`.

7. IS_U_FIELD : Either `yes` or `no`, indicating whether the field is to be used as the wind U-component field. For ARW, the wind U-component field must be interpolated to the U staggering (`output_stagger=U`); for NMM, the wind U-component field must be interpolated to the V staggering (`output_stagger=VV`). Default value is `no`.

8. IS_V_FIELD : Either `yes` or `no`, indicating whether the field is to be used as the wind V-component field. For ARW, the wind V-component field must be interpolated to the V staggering (`output_stagger=V`); for NMM, the wind V-component field must be interpolated to the V staggering (`output_stagger=VV`). Default value is `no`.

9. INTERP_OPTION : A sequence of one or more names of interpolation methods to be used when horizontally interpolating the field. Available interpolation methods are: `average_4pt`, `average_16pt`, `wt_average_4pt`, `wt_average_16pt`,

`nearest_neighbor`, `four_pt`, `sixteen_pt`, `search`, `average_gcell(`*r*`)`; for the grid cell average method (`average_gcell`), the optional argument *r* specifies the minimum ratio of source data resolution to simulation grid resolution at which the method will be applied; if a ratio is not specified, *r* = 0.0, and the option is used for any ratio. When a sequence of two or more methods are given, the methods should be separated by a + sign. Default value is `nearest_neighbor`.

10. INTERP_MASK : The name of the field to be used as an interpolation mask, along with the value within that field which signals masked points and an optional relational symbol, < or >. A specification takes the form *field*(?*maskval*), where *field* is the name of the field, *?* is an optional relational symbol (< or >), and *maskval* is a real value. Source data points will not be used in interpolation if the corresponding point in the *field* field is equal, greater than, or less than, the value of *maskval* for no relational symbol, a > symbol, or a < symbol, respectively. Default value is no mask.

11. INTERP_LAND_MASK : The name of the field to be used as an interpolation mask when interpolating to water points (determined by the static LANDMASK field), along with the value within that field which signals land points and an optional relational symbol, < or >. A specification takes the form *field*(?*maskval*), where *field* is the name of the field, *?* is an optional relational symbol (< or >), and *maskval* is a real value. Default value is no mask.

12. INTERP_WATER_MASK : The name of the field to be used as an interpolation mask when interpolating to land points (determined by the static LANDMASK field), along with the value within that field which signals water points and an optional relational symbol, < or >. A specification takes the form *field*(?*maskval*), where *field* is the name of the field, *?* is an optional relational symbol (< or >), and *maskval* is a real value. Default value is no mask.

13. FILL_MISSING : A real number specifying the value to be assigned to model grid points that received no interpolated value, for example, because of missing or incomplete meteorological data. Default value is 1.E20.

14. Z_DIM_NAME : For 3-dimensional meteorological fields, a character string giving the name of the vertical dimension to be used for the field on output. Default value is `num_metgrid_levels`.

15. DERIVED : Either `yes` or `no`, indicating whether the field is to be derived from other interpolated fields, rather than interpolated from an input field. Default value is `no`.

16. FILL_LEV : The `fill_lev` keyword, which may be specified multiple times within a table section, specifies how a level of the field should be filled if that level does not already exist. A generic value for the keyword takes the form *DLEVEL*:*FIELD*(*SLEVEL*), where *DLEVEL* specifies the level in the field to be filled, *FIELD* specifies the source

field from which to copy levels, and *SLEVEL* specifies the level within the source field to use. *DLEVEL* may either be an integer or the string `all`. *FIELD* may either be the name of another field, the string `const`, or the string `vertical_index`. If *FIELD* is specified as `const`, then *SLEVEL* is a constant value that will be used to fill with; if *FIELD* is specified as `vertical_index`, then (*SLEVEL*) must not be specified, and the value of the vertical index of the source field is used; if *DLEVEL* is 'all', then all levels from the field specified by the `level_template` keyword are used to fill the corresponding levels in the field, one at a time. No default value.

17. LEVEL_TEMPLATE : A character string giving the name of a field from which a list of vertical levels should be obtained and used as a template. This keyword is used in conjunction with a `fill_lev` specification that uses `all` in the *DLEVEL* part of its specification. No default value.

18. MASKED : Either `land`, `water`, or `both`. Setting `MASKED` to `land` or `water` indicates that the field should not be interpolated to WRF land or water points, respectively; however, setting `MASKED` to `both` indicates that the field should be interpolated to WRF land points using only land points in the source data and to WRF water points using only water points in the source data. When a field is masked, or invalid, the static LANDMASK field will be used to determine which model grid points the field should be interpolated to; invalid points will be assigned the value given by the `FILL_MISSING` keyword. Whether a source data point is land or water is determined by the masks specified using the `INTERP_LAND_MASK` and `INTERP_WATER_MASK` options. Default value is null (i.e., the field is valid for both land and water points).

19. MISSING_VALUE : A real number giving the value in the input field that is assumed to represent missing data. No default value.

20. VERTICAL_INTERP_OPTION : A character string specifying the vertical interpolation method that should be used when vertically interpolating to missing points. Currently, this option is not implemented. No default value.

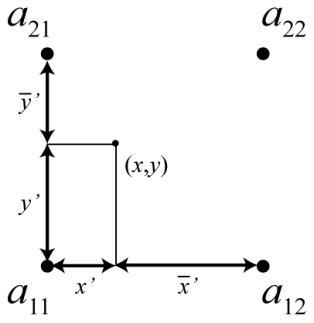
21. FLAG_IN_OUTPUT : A character string giving the name of a global attribute which will be assigned a value of 1 and written to the metgrid output if the interpolated field is to be output (output=yes). Default value is null (i.e., no flag will be written for the field).

Available Interpolation Options in Geogrid and Metgrid

Through the GEOGRID.TBL and METGRID.TBL files, the user can control the method by which source data – either static fields in the case of geogrid or meteorological fields in the case of metgrid – are interpolated. In fact, a list of interpolation methods may be given, in which case, if it is not possible to employ the *i*-th method in the list, the (*i*+1)-st method will be employed, until either some method can be used or there are no methods left to try in the list. For example, to use a four-point bi-linear interpolation scheme for a

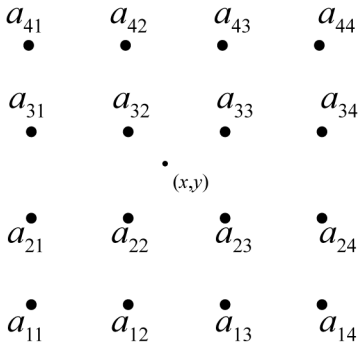
field, we could specify `interp_option=four_pt`. However, if the field had areas of missing values, which could prevent the `four_pt` option from being used, we could request that a simple four-point average be tried if the `four_pt` method couldn't be used by specifying `interp_option=four_pt+average_4pt` instead. Below, each of the available interpolation options in the WPS are described conceptually; for the details of each method, the user is referred to the source code in the file WPS/geogrid/src/interp_options.F.

1. four_pt : Four-point bi-linear interpolation



The four-point bi-linear interpolation method requires four valid source points a_{ij}, $1 \le i, j \le 2$, surrounding the point (x,y), to which geogrid or metgrid must interpolate, as illustrated in the figure above. Intuitively, the method works by linearly interpolating to the x-coordinate of the point (x,y) between a_{11} and a_{12}, and between a_{21} and a_{22}, and then linearly interpolating to the y-coordinate using these two interpolated values.

2. sixteen_pt : Sixteen-point overlapping parabolic interpolation



The sixteen_pt overlapping parabolic interpolation method requires sixteen valid source points surrounding the point (x,y), as illustrated in the figure above. The method works by fitting one parabola to the points a_{i1}, a_{i2}, and a_{i3}, and another parabola to the points a_{i2}, a_{i3}, and a_{i4}, for row i, $1 \le i \le 4$; then, an intermediate interpolated value p_i within row i at the x-coordinate of the point is computed by taking an average of the values of the two parabolas evaluated at x, with the average being weighted linearly by the distance of x from a_{i2} and a_{i3}. Finally, the interpolated value at (x,y) is found by performing the same

operations as for a row of points, but for the column of interpolated values p_i to the y-coordinate of (x,y).

3. average_4pt : Simple four-point average interpolation

The four-point average interpolation method requires at least one valid source data point from the four source points surrounding the point (x,y). The interpolated value is simply the average value of all valid values among these four points.

4. wt_average_4pt : Weighted four-point average interpolation

The weighted four-point average interpolation method can handle missing or masked source data points, and the interpolated value is given as the weighted average of all valid values, with the weight w_{ij} for the source point a_{ij}, $1 \le i, j \le 2$, given by

$$w_{ij} = \max\{0, 1 - \sqrt{(x - x_i)^2 + (y - y_j)^2}\}.$$

Here, x_i is the x-coordinate of a_{ij} and y_j is the y-coordinate of a_{ij}.

5. average_16pt : Simple sixteen-point average interpolation

The sixteen-point average interpolation method works in an identical way to the four-point average, but considers the sixteen points surrounding the point (x,y).

6. wt_average_16pt : Weighted sixteen-point average interpolation

The weighted sixteen-point average interpolation method works like the weighted four-point average, but considers the sixteen points surrounding (x,y); the weights in this method are given by

$$w_{ij} = \max\{0, 2 - \sqrt{(x - x_i)^2 + (y - y_j)^2}\},$$

where x_i and y_j are as defined for the weighted four-point method, and $1 \le i, j \le 4$.

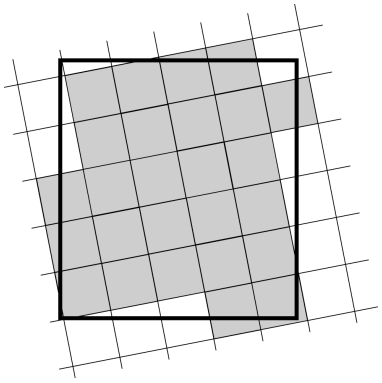
7. nearest_neighbor : Nearest neighbor interpolation

The nearest neighbor interpolation method simply sets the interpolated value at (x,y) to the value of the nearest source data point, regardless of whether this nearest source point is valid, missing, or masked.

8. search : Breadth-first search interpolation

The breadth-first search option works by treating the source data array as a 2-d grid graph, where each source data point, whether valid or not, is represented by a vertex. Then, the value assigned to the point (x,y) is found by beginning a breadth-first search at the vertex corresponding to the nearest neighbor of (x,y), and stopping once a vertex representing a valid (i.e., not masked or missing) source data point is found. In effect, this method can be thought of as "nearest *valid* neighbor".

9. average_gcell : Model grid-cell average



The grid-cell average interpolator may be used when the resolution of the source data is higher than the resolution of the model grid. For a model grid cell Γ, the method takes a simple average of the values of all source data points that are nearer to the center of Γ than to the center of any other grid cell. The operation of the grid-cell average method is illustrated in the figure above, where the interpolated value for the model grid cell – represented as the large rectangle – is given by the simple average of the values of all of the shaded source grid cells.

Land Use and Soil Categories in the Static Data

The default land use and soil category data sets that are provided as part of the WPS static data tar file contain categories that are matched with the USGS categories described in the VEGPARM.TBL and SOILPARM.TBL files in the WRF run directory. Descriptions of the 24 land use categories and 16 soil categories are provided in the tables below.

Table 1: USGS 24-category Land Use Categories

Land Use Category	Land Use Description
1	Urban and Built-up Land
2	Dryland Cropland and Pasture
3	Irrigated Cropland and Pasture

4	Mixed Dryland/Irrigated Cropland and Pasture
5	Cropland/Grassland Mosaic
6	Cropland/Woodland Mosaic
7	Grassland
8	Shrubland
9	Mixed Shrubland/Grassland
10	Savanna
11	Deciduous Broadleaf Forest
12	Deciduous Needleleaf Forest
13	Evergreen Broadleaf
14	Evergreen Needleleaf
15	Mixed Forest
16	Water Bodies
17	Herbaceous Wetland
18	Wooden Wetland
19	Barren or Sparsely Vegetated
20	Herbaceous Tundra
21	Wooded Tundra
22	Mixed Tundra
23	Bare Ground Tundra
24	Snow or Ice

Table 2: IGBP-Modified MODIS 20-category Land Use Categories

Land Use Category	Land Use Description
1	Evergreen Needleleaf Forest
2	Evergreen Broadleaf Forest
3	Deciduous Needleleaf Forest
4	Deciduous Broadleaf Forest
5	Mixed Forests
6	Closed Shrublands
7	Open Shrublands
8	Woody Savannas
9	Savannas
10	Grasslands
11	Permanent Wetlands
12	Croplands
13	Urban and Built-Up
14	Cropland/Natural Vegetation Mosaic
15	Snow and Ice
16	Barren or Sparsely Vegetated
17	Water
18	Wooded Tundra
19	Mixed Tundra
20	Barren Tundra

Table 3: 16-category Soil Categories

Soil Category	Soil Description
1	Sand
2	Loamy Sand
3	Sandy Loam
4	Silt Loam
5	Silt
6	Loam
7	Sandy Clay Loam
8	Silty Clay Loam
9	Clay Loam
10	Sandy Clay
11	Silty Clay
12	Clay
13	Organic Material
14	Water
15	Bedrock
16	Other (land-ice)

User's Guide for the NMM Core of the Weather Research and Forecast (WRF) Modeling System Version 3

Chapter 3: WRF-NMM Initialization

Table of Contents

Introduction

The ***real_nmm.exe*** portion of the code generates initial and boundary conditions for the WRF-NMM model (***wrf.exe***) that are derived from output files provided by the ***WPS.*** Inputs required for the WRF-NMM model are not restricted to WPS alone. Several variables are defined/re-defined using the ***real_nmm*** part of the routines. For instance, the WRF-NMM core uses the definition of the Coriolis parameter in ***real_nmm***, rather than that in WPS.

The ***real_nmm*** program performs the following tasks:

- Reads data from the namelist
- Allocates space
- Initializes remaining variables
- Reads input data from the WRF Preprocessing System (WPS)
- Prepares soil fields for use in the model (usually vertical interpolation to the requested levels)
- Checks to verify soil categories, land use, land mask, soil temperature, and sea surface temperature are all consistent with each other
- Vertically interpolates to the models computational surfaces.
- Generates initial condition file
- Generates lateral condition file

The ***real_nmm.exe*** program may be run as a distributed memory job, but there may be no computational speed up since this program relies heavily on I/O and does few computations.

Initialization for Real Data Cases

The ***real_nmm.exe*** code uses data files provided by the WRF Preprocessing System (WPS) as input. The data processed by the WPS typically come from a previously run, large-scale forecast model. The original data are generally in GRIB format and are ingested into the WPS. ***Please refer to the HWRF Users' Guide on how to generate initial and boundary files for the HWRF system.***

For example, a forecast from 2005 January 23 0000 UTC to 2005 January 24 0000 UTC which has original GRIB data available at 3h increments will have the following files previously generated by the WPS:

met_nmm.d01.2005-01-23_00_00_00.nc
met_nmm.d01.2005-01-23_03_00_00.nc
met_nmm.d01.2005-01-23_06_00_00.nc
met_nmm.d01.2005-01-23_09_00_00.nc
met_nmm.d01.2005-01-23_12_00_00.nc
met_nmm.d01.2005-01-23_15_00_00.nc
met_nmm.d01.2005-01-23_18_00_00.nc
met_nmm.d01.2005-01-23_21_00_00.nc
met_nmm.d01.2005-01-24_00_00_00.nc

The convention is to use "***met_nmm"*** to signify data that are output from the WPS and used as input into the ***real_nmm.exe*** program. The "***d01"*** part of the name is used to identify to which domain this data refers. The trailing characters are the date, where each WPS output file has only a single time-slice of processed data. The WPS package delivers data that are ready to be used in the WRF-NMM system.

The following statements apply to these data:

- The data adheres to the WRF IO API.
- The data has already been horizontally interpolated to the correct grid-point staggering for each variable, and the winds are correctly rotated to the WRF model map projection.
- 3-D meteorological data required from the WPS: ***pressure***, ***u***, ***v***, ***temperature***, ***relative humidity***, ***geopotential height***
- Optional 3-D hydrometeor data may be provided to the real program at run-time, but these fields will not be used in the coarse-grid lateral boundary file. Fields named: ***QR, QC, QS, QI, QG, QH, QNI*** (mixing ratio for rain, cloud, snow, ice, graupel, hail, and number concentration) are eligible for input from the metgrid output files.
- 3D soil data from the WPS: ***soil temperature***, ***soil moisture***, ***soil liquid*** (optional, depending on physics choices in the WRF model)
- 2D meteorological data from the WPS: ***sea level pressure***, ***surface pressure***, ***surface u and v***, ***surface temperature***, ***surface relative humidity***, ***input elevation***

- 2-D meteorological optional data from WPS: ***sea surface temperature***, ***physical snow depth***, ***water equivalent snow depth***
- 2D static data for the physical surface: ***terrain elevation***, ***land use categories***, ***soil texture categories***, ***temporally-interpolated monthly data***, ***land sea mask***, ***elevation of the input model's topography***
- 2D static data for the projection: ***map factors***, ***Coriolis***, ***projection rotation***, ***computational latitude***
- constants: ***domain size***, ***grid distances***, ***date***
- The WPS data may either be isobaric or some more-generalized vertical coordinate, where each column is monotonic in pressure
- All 3-D meteorological data (wind, temperature, height, moisture, pressure) must have the same number of levels, and variables must have the exact same levels. For example, it is not acceptable to have more levels for temperature (for example) than height. Likewise, it is not acceptable to have an extra level for the horizontal wind components, but not for moisture.

Running *real_nmm.exe*:

The procedure outlined below is used for single or multiple (nested) grid runs.

1. Change to the working directory of choice (***cd test/nmm_real*** or ***cd run***).

2. Make sure the files listed below reside in or are linked to the working-directory chosen to run the model:

aerosol.formatted
aerosol_lat.formatted
aerosol_lon.formatted
aerosol_plev.formatted
bulkdens.asc_s_0_03_0_9
bulkradii.asc_s_0_03_0_9
CAM_ABS_DATA
CAM_AEROPT_DATA
CAMtr_volume_mixing_ratio.A1B
CAMtr_volume_mixing_ratio.A2
CAMtr_volume_mixing_ratio.RCP4.5
CAMtr_volume_mixing_ratio.RCP6
CAMtr_volume_mixing_ratio.RCP8.5
capacity.asc
CCN_ACTIVATE.BIN
CLM_ALB_ICE_DFS_DATA
CLM_ALB_ICE_DRC_DATA
CLM_ASM_ICE_DFS_DATA
CLM_ASM_ICE_DRC_DATA
CLM_DRDSDT0_DATA
CLM_EXT_ICE_DFS_DATA

CLM_EXT_ICE_DRC_DATA
CLM_KAPPA_DATA
CLM_TAU_DATA
co2_trans
coeff_p.asc
coeff_q.asc
constants.asc
eta_micro_lookup.dat
ETAMPNEW_DATA
ETAMPNEW_DATA_DBL
ETAMPNEW_DATA.expanded_rain
ETAMPNEW_DATA.expanded_rain_DBL
GENPARM.TBL
grib2map.tbl
gribmap.txt
kernels.asc_s_0_03_0_9
kernels_z.asc
LANDUSE.TBL
masses.asc
MPTABLE.TBL
ozone.formatted
ozone_lat.formatted
ozone_plev.formatted
README.namelist
README.tslist
real_nmm.exe
RRTM_DATA
RRTM_DATA_DBL
RRTMG_LW_DATA
RRTMG_LW_DATA_DBL
RRTMG_SW_DATA
RRTMG_SW_DATA_DBL
SOILPARM.TBL
termvels.asc
tr49t67
tr49t85
tr67t85
URBPARM.TBL
URBPARM_UZE.TBL
VEGPARM.TBL
wind-turbine-1.tbl
wrf.exe
namelist.input

3. Make sure the ***met_nmm.d01**** files from the WPS either reside in or are linked to the working directory chosen to run the model. If nest(s) were run, also link in the ***geo_nmm_nest**** file(s).

4. Edit the ***namelist.input*** file in the working directory for dates, domain size, time step, output options, and physics options (see Chapter 4, Description of Namelist Variables section for details).

5. The command issued to run ***real_nmm.exe*** in the working directory will depend on the operating system.

When ***real_nmm.exe*** is successful, the following files that are used by ***wrf.exe*** should be found in the working-directory:

wrfinput_d01	(Initial conditions, single time level data.)
wrfbdy_d01	(Boundary conditions data for multiple time steps.)

To check whether the run is successful, look for "SUCCESS COMPLETE REAL_NMM INIT" at the end of the log file (e.g.., ***rsl.out.0000***, ***real_nmm.out***).

The ***real_nmm.exe*** portion of the code does not input or output any file relevant to nested domains. Initial and boundary conditions for WRF-NMM nests are interpolated down from the parent grids during the WRF model run.

User's Guide for the NMM Core of the Weather Research and Forecast (WRF) Modeling System Version 3

Chapter 4: WRF NMM Model

Table of Contents

Introduction

The WRF-NMM is a fully compressible, non-hydrostatic mesoscale model with a hydrostatic option (Janjic et al. 2001, Janjic 2003a,b). The model uses a terrain following hybrid sigma-pressure vertical coordinate. The grid staggering is the Arakawa E-grid. The same time step is used for all terms. The dynamics conserve a number of first and second order quantities including energy and enstrophy (Janjic 1984).

The WRF-NMM code contains an initialization program ***(real_nmm.exe***; see Chapter 3) and a numerical integration program (***wrf.exe***). The WRF-NMM model Version 3 supports a variety of capabilities. These include:

- Real-data simulations
- Non-hydrostatic and hydrostatic (runtime option)
- Full physics options
- One-way and two-way nesting
- Applications ranging from meters to thousands of kilometers
- Digital filter initialization

WRF-NMM Dynamics:

Time stepping:

Horizontally propagating fast-waves: Forward-backward scheme
Vertically propagating sound waves: Implicit scheme

Horizontal: Adams-Bashforth scheme
Vertical: Crank-Nicholson scheme
TKE, water species: Explicit, iterative, flux-corrected (called every two time steps).

Advection (space) for T, U, V:

Horizontal: Energy and enstrophy conserving, quadratic conservative, second order
Vertical: Quadratic conservative, second order
TKE, Water species: Upstream, flux-corrected, positive definite, conservative

Diffusion:

Diffusion in the WRF-NMM is categorized as lateral diffusion and vertical diffusion. The vertical diffusion in the PBL and in the free atmosphere is handled by the surface layer scheme and by the boundary layer parameterization scheme (Janjic 1996a, 1996b, 2002a, 2002b). The lateral diffusion is formulated following the Smagorinsky non-linear approach (Janjic 1990). The control parameter for the lateral diffusion is the square of Smagorinsky constant.

Divergence damping:

The horizontal component of divergence is damped (Sadourny 1975). In addition, if applied, the technique for coupling the elementary subgrids of the E grid (Janjic 1979) damps the divergent part of flow.

Physics Options

WRF offers multiple physics options that can be combined in many ways. The options typically range from simple and efficient to sophisticated and more computationally

costly, and from newly developed schemes to well tried schemes such as those in current operational models. All available WRF System physics package options available in WRF Version 3 are listed below. Some of these options have not yet been tested for WRF-NMM. Indication of the options that have been tested, as well as the level of the testing, is included in the discussion below. ***Please refer to the HWRF Users Guide for tested and supported physics option with the HWRF system.***

It is recommended that the same physics be used in all grids (coarsest and nests). The only exception is that the cumulus parameterization may be activated on coarser grids and turned off on finer grids.

Please refer to the most recent WRF-ARW Users Guide for detailed and descriptions of the schemes.

Microphysics (*mp_physics*)

a. Kessler scheme: A warm-rain (i.e. no ice) scheme used commonly in idealized cloud modeling studies (*mp_physics* = 1).

b. Lin et al. scheme: A sophisticated scheme that has ice, snow and graupel processes, suitable for real-data high-resolution simulations (2).

c. WRF Single-Moment 3-class scheme: A simple efficient scheme with ice and snow processes suitable for mesoscale grid sizes (3).

d. WRF Single-Moment 5-class scheme: A slightly more sophisticated version of (c) that allows for mixed-phase processes and super-cooled water (4). (This scheme has been preliminarily tested for WRF-NMM.)

e. Ferrier-Aligo microphysics: Adopted from Eta HWRF Ferrier microphysics. The modified microphysics assumes the maximum number concentration of large ice varies in different cloud regimes. (5). (This scheme is well tested and used operationally at NCEP for HWRF.) New in Version 3.7. The Eta HWRF has been moved to 85.

f. Ferrier-Aligo microphysics: Similar to Ferrier-Aligo microphysics, but advects individual species (15).

g. Eta HWRF microphysics: Similar to Eta microphysics (e), but modified to be suitable for the tropics (85). (This scheme is well tested for HWRF.) New in Version 3.2.

h. Etamp microphysics: A simple efficient scheme with diagnostic mixed-phase processes. For coarse resolutions use option (95). (This scheme is well tested for WRF-NMM.)

i. WRF Single-Moment 6-class scheme: A scheme with ice, snow and graupel processes suitable for high-resolution simulations (6). (This scheme has been preliminarily tested for WRF-NMM.)

j. Goddard microphysics scheme. A scheme with ice, snow and graupel processes suitable for high-resolution simulations (7).

k. New Thompson et al. scheme: A new scheme with ice, snow and graupel processes suitable for high-resolution simulations (8). This adds rain number concentration and updates the scheme from the one in Version 3.0. New in Version 3.1. (This scheme has been well tested for WRF-NMM.)

l. Milbrandt-Yau Double-Moment 7-class scheme (9). This scheme includes separate categories for hail and graupel with double-moment cloud, rain, ice, snow, graupel and hail. New in Version 3.2.

m. Morrison double-moment scheme (10). Double-moment ice, snow, rain and graupel for cloud-resolving simulations. New in Version 3.0.

n. Stony Brook University (Y. Lin) scheme (13). This is a 5-class scheme with riming intensity predicted to account for mixed-phase processes. New in Version 3.3.

o. WRF Double-Moment 5-class scheme (14). This scheme has double-moment rain. Cloud and CCN for warm processes, but is otherwise like WSM5. New in Version 3.1.

p. WRF Double-Moment 6-class scheme (16). This scheme has double-moment rain. Cloud and CCN for warm processes, but is otherwise like WSM6. New in Version 3.1.

q. NSSL 2-moment scheme (17, 18). This is a two-moment scheme for cloud droplets, rain drops, ice crystals, snow, graupel, and hail. It also predicts average graupel particle density, which allows graupel to span the range from frozen drops to low-density graupel. There is an additional option to predict cloud condensation nuclei (CCN, option 18) concentration (intended for idealized simulations). The scheme is intended for cloud-resolving simulations (dx <= 2km) in research applications. New in Version 3.4.

r. Thompson aerosol-aware (28). This scheme considers water- and ice-friendly aerosols. A climatology dataset may be used to specify initial and boundary conditions for the aerosol variables (Thompson and Eidhammer, 2014, JAS.) New in Version 3.6.

Summary of Microphysics Options

mp_physics	Scheme	Reference	Added
1	Kessler	Kessler (1969)	2000

2	Lin (Purdue)	Lin, Farley and Orville (1983, JCAM)	2000
3	WSM3	Hong, Dudhia and Chen (2004, MWR)	2004
4	WSM5	Hong, Dudhia and Chen (2004, MWR)	2004
5	Ferrier-Aligo	Aligo, Ferrier, Carley, Rogers, Pyle, Weiss and Jirak (2014, AMS Severe local storms)	2015
6	WSM6	Hong and Lim (2006, JKMS)	2004
7	Goddard	Tao, Simpson and McCumber (1989, MWR)	2008
8	Thompson	Thompson, Field, Rasmussen and Hall (2008, MWR)	2009
9	Milbrandt 2-mom	Milbrandt and Yau (2005, JAS)	2010
10	Morrison 2-mom	Morrison, Thompson and Tatarskii (2009, MWR)	2008
13	SBU-YLin	Lin and Colle (2011, MWR)	2011
14	WDM5	Lim and Hong (2010)	2009
16	WDM6	Lim and Hong (2010)	2009
17	NSSL 2-mom	Mansell, Ziegler and Brunning (2010)	2012
18	NSSL 2-mom w/CCN prediction	Mansell, Ziegler and Brunning (2010)	2012
19	NSSL 1-mom		2013
21	NSSL 1-momlfo		2013
28	Thompson aerosol-aware	Thompson and Eidhammer (2014, JAS)	2014
85	Eta HWRF	Tallapragada, V et al. (2014, HWRF Sci Doc)	2010
95	Etamp	Rogers, Black, Ferrier, Lin, Parrish and DiMego (2001)	2000

mp_physics	Scheme	Cores	Mass Variables	Number Variables
1	Kessler	ARW	Qc Qr	
2	Lin (Purdue)	ARW (Chem)	Qc Qr Qi Qs Qg	
3	WSM3	ARW	Qc Qr	
4	WSM5	ARW/NMM	Qc Qr Qi Qs	

5	Ferrier-Aligo	ARW/NMM	Qc Qr Qs (Qt*)	
6	WSM6	ARW/NMM	Qc Qr Qi Qs Qg	
8	Thompson	ARW/NMM	Qc Qr Qi Qs Qg	Ni Nr
9	Milbrandt 2-mom	ARW	Qc Qr Qi Qs Qg Qh	Nc Nr Ni Ns Ng Nh
10	Morrison 2-mom	ARW (Chem)	Qc Qr Qi Qs Qg	Nr Ni Ns Ng
13	SBU-YLin	ARW	Qc Qr Qi Qs	
14	WDM5	ARW	Qc Qr Qi Qs	Nn** Nc Nr
15	Ferrier-Aligo advected	NMM	Qc Qr Qs	
16	WDM6	ARW	Qc Qr Qi Qs Qg	Nn** Nc Nr
17	NSSL 2-mom	ARW	Qc Qr Qi Qs Qg Qh	Nc Nr Ni Ns Ng Nh
18	NSSL 2-mom +CCN	ARW	Qc Qr Qi Qs Qg Qh	Nc Nr Ni Ns Ng Nh
19	NSSL 1-mom	ARW	Qc Qr Qi Qs Qg Qh	Vg***
21	NSSL 1-momlfo	ARW	Qc Qr Qi Qs Qg	
28	Thompson aerosol-aware	ARW/NMM	Qc Qr Qi Qs Qg	Ni Nr Nwf Nif
85	Eta HWRF	NMM	Qc Qr Qs (Qt*)	
95	Etamp	ARW/NMM	Qc Qr Qs (Qt*)	

* Advects only total condensates ** Nn = CCN number

Longwave Radiation (*ra_lw_physics*)

a. RRTM scheme: Rapid Radiative Transfer Model. An accurate scheme using look-up tables for efficiency. Accounts for multiple bands, trace gases, and microphysics species (*ra_lw_physics* = 1). For trace gases, the volume-mixing ratio values for CO2=330e-6, N2O=0. and CH4=0. in pre-V3.5 code; in V3.5, CO2=379e-6, N2O=319e-9 and CH4=1774e-9. See section 2.3 for time-varying option. (This scheme has been preliminarily tested for WRF-NMM.)

b. GFDL scheme: Eta operational radiation scheme. An older multi-band scheme with carbon dioxide, ozone and microphysics effects (99). (This scheme is well tested for WRF-NMM, used operationally at NCEP.) ***Note:*** *If it is desired to run GFDL with a microphysics scheme other than Ferrier, a modification to module_ra_gfdleta.F is needed to comment out (!) #define FERRIER_GFDL.*

c. Modified GFDL scheme: Similar to the GFDL scheme (b) but modified to be suitable for the tropics (98). (This scheme is well tested for HWRF.) New in Version 3.2.

d. CAM scheme: from the CAM 3 climate model used in CCSM. Allows for aerosols and trace gases (3). It uses yearly CO2, and constant N2O (311e-9) and CH4 (1714e-9). See section 2.3 for the time-varying option.

e. RRTMG scheme. A new version of RRTM (4). It includes the MCICA method of random cloud overlap. For major trace gases, CO2=379e-6, N2O=319e-9, CH4=1774e-9. See section 2.3 for the time-varying option. (This scheme is well tested and operational for HWRF.)

f. New Goddard scheme (5). Efficient, multiple bands, ozone from climatology. It uses constant CO2=337e-6, N2O=320e-9, CH4=1790e-9. New in Version 3.3.

g. Fu-Liou-Gu scheme (7). Multiple bands, cloud and cloud fraction effects, ozone profile from climatology and tracer gases. CO_2=345e-6. New in Version 3.4.

Shortwave Radiation (*ra_sw_physics*)

a. Dudhia scheme: Simple downward integration allowing efficiently for clouds and clear-sky absorption and scattering. When used in high-resolution simulations, sloping and shadowing effects may be considered (*ra_sw_physics* = 1). (This scheme has been preliminarily tested for WRF-NMM.)

b. Goddard shortwave: Two-stream multi-band scheme with ozone from climatology and cloud effects (2).

c. GFDL shortwave: Eta operational scheme. Two-stream multi-band scheme with ozone from climatology and cloud effects (99). (This scheme is well-tested for WRF-NMM) ***Note:*** *If it is desired to run GFDL with a microphysics scheme other than Ferrier, a modification to module_ra_gfdleta.F is needed to comment out (!) #define FERRIER_GFDL.*

d. Modified GFDL shortwave: Similar to the GFDL shortwave (c), but modified to be suitable for tropics (98). (This scheme is well tested for HWRF.) New in Version 3.2.

e. CAM scheme: from the CAM 3 climate model used in CCSM. Allows for aerosols and trace gases (3).

f. RRTMG shortwave. A new shortwave scheme with the MCICA method of random cloud overlap (4). New in Version 3.1. (This scheme is well tested and operational for HWRF.)

g. New Goddard scheme (5). Efficient, multiple bands, ozone from climatology. New in Version 3.3.

h. Fu-Liou-Gu scheme (7). multiple bands, cloud and cloud fraction effects, ozone profile from climatology, can allow for aerosols. New in Version 3.4.

i. Held-Suarez relaxation. A temperature relaxation scheme designed for idealized tests only (31).

j. *swrad_scat*: scattering turning parameter for *ra_sw_physics* = 1. Default value is 1, which is equivalent to 1.e-5 m^2/kg. When the value is greater than 1, it increases the scattering.

Input to radiation options

a. CAM Green House Gases: Provides yearly green house gases from 1765 to 2500. The option is activated by compiling WRF with the macro –DCLWRFGHG added in configure.wrf. Once compiled, CAM, RRTM and RRTMG long-wave schemes will see these gases. Five scenario files are available: from IPCC AR5: CAMtr_volume_mixing_ratio .RCP4.5, CAMtr_volume_mixing_ratio.RCP6, and CAMtr_volume_mixing_ratio.RCP8.5; from IPCC AR4: CAMtr_volume_mixing_ratio.A1B, and CAMtr_volume_mixing_ratio.A2. The default points to the RCP8.5 file. New in Version 3.5.

b. Climatological ozone and aerosol data for RRTMG: The ozone data is adapted from CAM radiation (ra_*_physics=3), and it has latitudinal (2.82 degrees), height and temporal (monthly) variation, as opposed to the default ozone used in the scheme that only varies with height. This is activated by the namelist option *o3input* = 2. The aerosol data is based on Tegen et al. (1997), which has 6 types: organic carbon, black carbon, sulfate, sea salt, dust and stratospheric aerosol (volcanic ash, which is zero). The data also has spatial (5 degrees in longitude and 4 degrees in latitudes) and temporal (monthly) variations. The option is activated by the namelist option *aer_opt* = 1. New in Version 3.5.

c. Aerosol input for RRTMG and Goddard radiation options (*aer_opt = 2*). Either AOD or AOD plus Angstrom exponent, single scattering albedo, and cloud asymmetry parameter can be provided via constant values from namelist or 2D input fields via auxiliary input stream 15. Aerosol type can be set too. New in V3.6.

Summary of Radiation Physics Options

ra_sw_physics	Scheme	Reference	Added
1	Dudhia	Dudhia (1989, JAS)	2000

2	Goddard	Chou and Suarez (1994, NASA Tech Memo)	2000
3	CAM	Collins et al. (2004, NCAR Tech Note)	2006
4	RRTMG	Iacono et al. (2008, JGR)	2009
5	New Goddard	Chou and Suarez (1999, NASA Tech Memo)	2011
7	FLG	Gu et al. (2011, JGR), Fu and Liou (1992, JAS)	2012
99	GFDL	Fels and Schwarzkopf (1981, JGR)	2004

ra_sw_ physics	Scheme	Cores+Chem	Microphysics Interaction	Cloud Fraction	Ozone
1	Dudhia	ARW NMM + Chem(PM2.5)	Qc Qr Qi Qs Qg	1/0	none
2	GSFC	ARW+Chem(τ)	Qc Qi	1/0	5 profiles
3	CAM	ARW	Qc Qi Qs	max-rand overlap	lat/month
4	RRTMG	ARW + Chem (τ), NMM	Qc Qr Qi Qs	max-rand overlap	1 profile or lat/month
5	New Goddard	ARW	Qc Qr Qi Qs Qg	1/0	5 profiles
7	FLG	ARW	Qc Qr Qi Qs Qg	1/0	5 profiles
99	GFDL	ARW NMM	Qc Qr Qi Qs	max-rand overlap	lat/date

ra_lw_physics	**Scheme**	**Reference**	**Added**
1	RRTM	Mlawer et al. (1997, JGR)	2000
3	CAM	Collins et al. (2004, NCAR Tech Note)	2006
4	RRTMG	Iacono et al. (2008, JGR)	2009
5	New Goddard	Chou and Suarez (1999, NASA Tech Memo)	2011
7	FLG	Gu et al. (2011, JGR), Fu and Liou (1992, JAS)	2012
31	Held-Suarez		2008

99	GFDL	Fels and Schwarzkopf (1981, JGR)	2004

ra_lw_ physics	Scheme	Cores+Chem	Microphysics Interaction	Cloud Fraction	Ozone	GHG
1	RRTM	ARW NMM	Qc Qr Qi Qs Qg	1/0	1 profile	constant or yearly GHG
3	CAM	ARW	Qc Qi Qs	max-rand overlap	lat/month	yearly CO2 or yearly GHG
4	RRTMG	ARW + Chem (τ), NMM	Qc Qr Qi Qs	max-rand overlap	1 profile or lat/month	constant or yearly GHG
5	New Goddard	ARW	Qc Qr Qi Qs Qg	1/0	5 profiles	constant
7	FLG	ARW	Qc Qr Qi Qs Qg	1/0	5 profiles	constant
31	Held-Suarez	ARW	none	none		none
99	GFDL	ARW NMM	Qc Qr Qi Qs	max-rand overlap	lat/date	constant

Surface Layer (*sf_sfclay_physics*)

a. MM5 similarity: Based on Monin-Obukhov with Carslon-Boland viscous sub-layer and standard similarity functions from look-up tables (*sf_sfclay_physics* = 1). (This scheme has been preliminarily tested for WRF-NMM.)

b. Eta similarity: Used in Eta model. Based on Monin-Obukhov with Zilitinkevich thermal roughness length and standard similarity functions from look-up tables (2). (This scheme is well tested for WRF-NMM.)

c. NCEP Global Forecasting System (GFS) scheme: The Monin-Obukhov similarity profile relationship is applied to obtain the surface stress and latent heat fluxes using a formulation based on Miyakoda and Sirutis (1986) modified for very stable and unstable situations. Land surface evaporation has three components (direct evaporation from the soil and canopy, and transpiration from vegetation) following the formulation of Pan and Mahrt (1987) (3). (This scheme has been preliminarily tested for WRF-NMM.)

d. Pleim-Xiu surface layer (7). New in Version 3.0.

e. QNSE surface layer. Quasi-Normal Scale Elimination PBL scheme's surface layer option (4). New in Version 3.1.

f. TEMF surface layer. Total Energy – Mass Flux surface layer scheme (10). New in Version 3.3.

g. Revised MM5 surface layer scheme (11): Remove limits and use updated stability functions. New in Version 3.4. (Jimenez et al. MWR 2012).

h. GFDL surface layer (88): (This scheme is well tested and used operationally at NCEP for HWRF.)

i. *iz0tlnd* = 1 (for *sf_sfclay_physics* = 1 or 2), Chen-Zhang thermal roughness length over land, which depends on vegetation height, 0 = original thermal roughness length in each sfclay option. New in Version 3.2.

Land Surface (*sf_surface_physics*)

a. 5-layer thermal diffusion: Soil temperature only scheme, using five layers (*sf_surface_physics* = 1).

b. Noah Land Surface Model: Unified NCEP/NCAR/AFWA scheme with soil temperature and moisture in four layers, fractional snow cover and frozen soil physics. New modifications are added in Version 3.1 to better represent processes over ice sheets and snow covered area (2). (This scheme is well tested for WRF-NMM, used operationally at NCEP for HWRF.)

- In V3.6, a sub-tiling option is introduced, and it is activated by namelist *sf_surface_mosaic* = 1, and the number of tiles in a grid box is defined by namelist *mosaic_cat*, with a default value of 3.

c. RUC Land Surface Model: RUC operational scheme with soil temperature and moisture in six layers, multi-layer snow and frozen soil physics (3). (This scheme has been preliminarily tested for WRF-NMM.)

d. Pleim-Xiu Land Surface Model. Two-layer scheme with vegetation and sub-grid tiling (7). New in Version 3.0: The Pleim-Xiu land surface model (PX LSM; Pleim and Xiu 1995; Xiu and Pleim 2001) was developed to provide realistic ground temperature, soil moisture, and surface sensible and latent heat fluxes in mesoscale meteorological models. The PX LSM is based on the ISBA model (Noilhan and Planton 1989), and includes a 2-layer force-restore soil temperature and moisture model.

e. GFDL slab model. Used together with GFDL surface layer scheme (88). (This scheme is well tested for HWRF.) New in Version 3.2.

f. Noah-MP (multi-physics) Land Surface Model: uses multiple options for key land-atmosphere interaction processes (4). (This scheme has been preliminarily tested for WRF-NMM.)

g. SSiB Land Surface Model: This is the third generation of the Simplified Simple Biosphere Model (Xue et al. 1991; Sun and Xue, 2001) (8). New in Version 3.4.

h. CLM4 (Community Land Model Version 4, Oleson et al. 2010; Lawrence et al. 2010): It contains sophisticated treatment of biogeophysics, hydrology, biogeochemistry, and dynamic vegetation. (5) New in Version 3.5. Updated for 20/21 category MODIS landuse data in V3.6.

Urban Surface (sf_urban_physics)

a. Urban canopy model (1): 3-category UCM option with surface effects for roofs, walls, and streets. In V3.7, a green roof option is added.

b. BEP (2). Building Environment Parameterization: Multi-layer urban canopy model that allows for buildings higher than the lowest model levels. Only works with Noah LSM and Boulac and MYJ PBL options. New in Version 3.1.

c. BEM (3). Building Energy Model. Adds to BEP, building energy budget with heating and cooling systems. Works with same options as BEP. New in Version 3.2.

Lake Physics (sf_lake_physics)

a. CLM 4.5 lake model (1). The lake scheme was obtained from the Community Land Model version 4.5 (Oleson et al. 2013) with some modifications by Gu et al. (2013).

Planetary Boundary layer (*bl_pbl_physics*)

a. Yonsei University scheme: Non-local-K scheme with explicit entrainment layer and parabolic K profile in unstable mixed layer (*bl_pbl_physics* = 1). (This scheme has been preliminarily tested for WRF-NMM.)

b. Mellor-Yamada-Janjic scheme: Eta operational scheme. One-dimensional prognostic turbulent kinetic energy scheme with local vertical mixing (2). (This scheme is well-tested for WRF-NMM, used operationally at NCEP.)

c. NCEP Global Forecast System scheme: First-order vertical diffusion scheme of Troen and Mahrt (1986) further described in Hong and Pan (1996). The PBL height is determined using an iterative bulk-Richardson approach working from the ground upward whereupon the profile of the diffusivity coefficient is specified as a cubic function of the PBL height. Coefficient values are obtained by matching the surface-layer fluxes. A counter-gradient flux parameterization is included (93).

d. MRF scheme: Older version of (a) with implicit treatment of entrainment layer as part of non-local-K mixed layer (99).

e. ACM2 PBL: Asymmetric Convective Model with non-local upward mixing and local downward mixing (7).

f. Quasi-Normal Scale Elimination PBL (4). A TKE-prediction option that uses a new theory for stably stratified regions. Daytime part uses eddy diffusivity mass-flux method with shallow convection (mfshconv = 1).

g. Mellor-Yamada Nakanishi and Niino Level 2.5 PBL (5). Predicts sub-grid TKE terms.

h. Mellor-Yamada Nakanishi and Niino Level 3 PBL (6). Predicts TKE and other second-moment terms.

i. BouLac PBL (8): Bougeault-Lacarrère PBL. A TKE-prediction option. Designed for use with BEP urban model.

j. UW (Bretherton and Park) scheme (9). TKE scheme from CESM climate model.

k. Total Energy - Mass Flux (TEMF) scheme (10). Sub-grid total energy prognostic variable, plus mass-flux type shallow convection.

l. *topo_wind*: Topographic correction for surface winds to represent extra drag from sub-grid topography and enhanced flow at hill tops (1) (Jimenez and Dudhia, JAMC

2012). Works with YSU PBL only. New in Version 3.4. A simpler terrain variance-related correction (2). New in Version 3.5.

m. GFS Hybrid-EDMF: Adapted from GFS PBL scheme. Mass flux approach is used to represent non-local mixing under convective conditions (3). (This scheme is well tested and used operationally at NCEP for HWRF.)

Note: *Two-meter temperatures are only available when running with MYJ scheme (2).*

Summary of PBL Physics Options

bl_pbl_physics	Scheme	Reference	Added
1	YSU	Hong, Noh and Dudhia (2006, MWR)	2004
2	MYJ	Janjic (1994, MWR)	2000
3	GFS	Hong and Pan (1996, MWR)	2005
4	QNSE	Sukoriansky, Galperin and Perov (2005, BLM)	2009
7	ACM2	Pleim (2007, JAMC)	2008
8	BouLac	Bougeault and Lacarrere (1989, MWR)	2009
9	UW	Bretherton and Park (2009, JC)	2011
10	TEMF	Angevine, Jiang and Mauriten (2010, MWR)	2011
12	GBM	Grenier and Bretherton (2001, MWR)	2013
99	MRF	Hong and Pan (1996, MWR)	2000
93	GFSEDMF	Han et al (2016, WAF)	2016

bl_pbl_ physics	Scheme	Cores	sf_sfclay_ physics	Prognostic variables	Diagnostic variables	Cloud mixing
1	YSU	ARW/ NMM	1		exch_h	QC,QI
2	MYJ	ARW/ NMM	2	TKE_PBL	EL_MYJ, exch_h	QC,QI

3	GFS	NMM	3			QC,QI
4	QNSE	ARW/ NMM	4	TKE_PBL	EL_MYJ, exch_h, exch_m	QC,QI
5	MYNN2	ARW	1,2,5	QKE	Tsq, Qsq, Cov, exch_h, exch_m	QC
6	MYNN3	ARW	1,2,5	QKE, Tsq, Qsq, Cov	exch_h, exch_m	QC
7	ACM2	ARW	1,7			QC,QI
8	BouLac	ARW	1,2	TKE_PBL	EL_PBL, exch_h, exch_m, wu_tur, wv_tur, wt_tur, wq_tur	QC
9	UW	ARW	2	TKE_PBL	exch_h, exch_m	QC
10	TEMF	ARW	10	TE_TEMF	*_temf	QC, QI
99	MRF	ARW/ NMM	1			QC,QI
93	GFSEDMF (hwrf)	NMM	88			

Cumulus Parameterization (*cu_physics*)

a. Kain-Fritsch scheme: Deep and shallow convection sub-grid scheme using a mass flux approach with downdrafts and CAPE removal time scale (*cu_physics* = 1). (This scheme has been preliminarily tested for WRF-NMM.)

- kfeta_trigger = 1 – default trigger; = 2 – moisture-advection modulated trigger function [based on Ma and Tan (2009, Atmospheric Research)]. May improve results in subtropical regions when large-scale forcing is weak.

b. Betts-Miller-Janjic scheme. Operational Eta scheme. Column moist adjustment scheme relaxing towards a well-mixed profile (2). (This scheme is well tested for WRF-NMM, used operationally at NCEP.)

c. Grell-Devenyi ensemble scheme: Multi-closure, multi-parameter, ensemble method with typically 144 sub-grid members (moved to option 93 in V3.5). (This scheme has been preliminarily tested for WRF-NMM.)

e. Simplified Arakawa-Schubert scheme (94): Simple mass-flux scheme with quasi-equilibrium closure with shallow mixing scheme (and momentum transport in NMM only). Adapted for ARW in Version 3.3.

f. Grell 3D is an improved version of the GD scheme that may also be used on high resolution (in addition to coarser resolutions) if subsidence spreading (option cugd_avedx) is turned on (5). New in Version 3.0.

g. Tiedtke scheme (U. of Hawaii version) (6). Mass-flux type scheme with CAPE-removal time scale, shallow component and momentum transport. New in Version 3.3.

h. Zhang-McFarlane scheme (7). Mass-flux CAPE-removal type deep convection from CESM climate model with momentum transport.

i. New Simplified Arakawa-Schubert (14). New mass-flux scheme with deep and shallow components and momentum transport..

j. Simplified Arakawa-Schubert (84). New mass-flux scheme with deep and shallow components and momentum transport. (This scheme is well tested for HWRF)

k. Grell-Freitas (GF) scheme (3): An improved GD scheme that tries to smooth the transition to cloud-resolving scales, as proposed by Arakawa et al. (2004).

l. Old Kain-Fritsch scheme: Deep convection scheme using a mass flux approach with downdrafts and CAPE removal time scale (99). (This scheme has been preliminarily tested for WRF-NMM.)

m. Scale Aware SAS (SASAS) scheme (4): Based on SAS (84). The SASAS scheme uses a scale-aware feature to modulate the updraft area according to the horizontal grid spacing. (This scheme is well tested for HWRF, used operationally at NCEP.)

Summary of Cumulus Parameterization Options

cu_physics	Scheme	Reference	Added
1	Kain-Fritsch	Kain (2004, JAM)	2000
2	Betts-Miller-Janjic	Janjic (1994, MWR; 2000, JAS)	2002
3	Grell-Devenyi	Grell and Devenyi (2002, GRL)	2002
4	Old SAS	(Pan and Wu 1995, Hong and Pan 1998, Pan 2003)	2010

5	Grell-3	-	2008
6	Tiedtke	Tiedtke (1989, MWR), Zhang et al. (2011, submitted)	2011
7	Zhang-McFarlane	Zhang and McFarlane (1995, AO)	2011
14	New SAS	Han and Pan (2011)	2011
84	Simplified Arakawa-Schubert	Han and Pan (2011)	2005/ 2011
86	Scale Aware SAS	Han and Pan (2011), Biswas et al (2016)	2016
93	Grell-Devenyi	Grell and Devenyi (2002, GRL)	2002
99	Old Kain-Fritsch	Kain and Fritsch (1990, JAS; 1993, Meteo. Monogr.)	2000

cu_physics	Scheme	Cores	Moisture Tendencies	Momentum Tendencies	Shallow Convection
1	Kain-Fritsch	ARW / NMM	Qc Qr Qi Qs	no	yes
2	BMJ	ARW / NMM	-	no	yes
3	GD	ARW	Qc Qi	no	no
4	Old SAS	ARW/NMM	Qc Qi	yes (NMM)	yes
5	G3	ARW	Qc Qi	no	yes
6	Tiedtke	ARW/NMM	Qc Qi	yes	yes
7	Zhang-McFarlane	ARW	Qc Qi	yes	no
14	NSAS	ARW/NMM	Qc Qr Qi Qs	yes	yes
84	SAS	ARW / NMM	Qc Qi	yes (NMM)	yes
86	SASAS	NMM	Qc Qi	yes (NMM)	yes
93	GD	ARW	Qc Qi	no	no
99	Old KF	ARW	Qc Qr Qi Qs	no	no

Shallow convection option (shcu_physics)

a. *ishallow* = 1, shallow convection option on. Works together with Grell 3D scheme (cu_physics = 5) – will move to shcu_physics category in the future.

b. UW (Bretherton and Park) scheme (2). Shallow cumulus option from CESM climate model with momentum transport. New in Version 3.3.

c. GRIMS (Global/Regional Integrated Modeling System) scheme: it represents the shallow convection process by using eddy-diffusion and the pal algorithm, and couples directly to the YSU PBL scheme. New in Version 3.5.

Other physics options

a. *gwd_opt*: Gravity wave drag option. Can be activated when grid size is greater than 10 km. May be beneficial for simulations longer than 5 days and over a large domain with mountain ranges. Default *gwd_opt*=0. Use *gwd_opt=2* for WRF-NMM.

b. *mommix:* Coefficient used in the calculation of momentum mixing tendency terms. Default *mommix*=1.0. Only used with SAS cumulus scheme (84, 94).

c. *h_diff:* Coefficient used in the calculation of horizontal momentum diffusion terms. Default *h_diff*=0.1. Only used when environment variable HWRF is set.

d. *sfenth*: Enthalpy flux factor. Default *sfenth*=0.0. Only used with GFDL surface scheme.

d. *co_2tf:* CO2 transmission coefficient option. Default *co_2tf*=1.

e. *sas_pgcon:* Convectively forced pressure gradient factor (SAS schemes 84 and 94) Default *sas_pgcon*=0.55.

f. *gfs_alpha:* Boundary layer depth factor. Default *gfs_alpha*=-1.0 (GFS PBL scheme 3).

g. *sas_mass_flux:* Mass flux limit (SAS scheme). Default *sas_mass_flux*=0.5 (SAS scheme 84)

h. *var_ric*: Use of variable critical Richardson number (Ric) in GFS PBL scheme.. Default *var_ric*=1 to use constant Ric, else set *var_ric*=0 to use constant Ric.

i. *coef_ric_l*: Placeholder for the coefficient used in the calculation of the variable critical Richardson number (Ric) in GFS PBL scheme. Default *coef_ric_l*=0.16.

j. *coef_ric_s*: Placeholder for the coefficient used in the calculation of the variable critical Richardson number (Ric) in GFS PBL scheme. Default *coef_ric_s*=0.25.

k. *icloud*: When set to 0, it turns off cloud effect on optical depth in shortwave radiation options 1, 4 and longwave radiation option 1, 4. Note since V3.6, this namelist also controls which cloud fraction method to use for radiation. When set to 3, it turns on partial cloudiness scheme in RRTMG (4) scheme.

Other dynamics options

a. *euler_adv*: Logical switch that turns on/off highly-conservative passive advection. Default *euler_adv*=.false. *(**Note:** ONLY compatible with Ferrier-Aligo MP (5), Ferrier MP (95) else set to .false.)*

b. *codamp*: Divergence damping weighting factor (larger = more damping). Default *codamp*=6.4

c. *coac*: Horizontal diffusion weighting factor (larger = more diffusion).

d. *slophc*: Maximum model level slope (dZ/dy) for which horizontal diffusion is applied. Larger values applies horizontal diffusion over more mountainous terrain. Default *slophc*=6.363961e-3

e. *wp*: Off-centering weight in the updating of nonhyrostatic epsilon term in the nonhydrostatic solver. Very high-resolution runs (sub-1.5 km scale), particularly if model layers near the top of atmosphere are thick, will benefit from *wp* of about 0.10 (0.15 as an absolute upper limit) to stabilize the integration. Default *wp*=0.00

f. *vortex_tracker:* Vortex tracking algorithm for HWRF. Default *vortex_tracker*=2

g. *movemin:* Frequency with which nest tracker routine will be called in HWRF (multiples of nphs). Default *movemin*=10.

h. *nomove_freq*: To prevent noise in the output files, disable nest movement at initialization time or multiples of this interval, if this interval is set to a positive number (hours). By default, this is disabled (*nomove_freq=-1*).

Operational Configuration

Below is a summary of physics options that are well-tested for WRF-NMM and are used operationally at NCEP for the Hurricane WRF (HWRF) Model:

&physics	Identifying	Physics options

	Number	
mp_physics (max_dom)	5	Ferrier-Aligo
ra_lw_physics	4	RRTMG
ra_sw_physics	4	RRTMG
sf_sfclay_physics	88	HWRF surface layer scheme
sf_surface_physics	2	Noah Land Surface
bl_pbl_physics	3	GFS Hybrid-EDMF pbl scheme
cu_physics	4	SASAS scheme

Description of Namelist Variables

The settings in the ***namelist.input*** file are used to configure WRF-NMM. This file should be edited to specify: dates, number and size of domains, time step, physics options, and output options. When modifying the ***namelist.input*** file, be sure to take into account the following points:

time_step: The general rule for determining the time step of the coarsest grid follows from the CFL criterion. If ***d*** is the grid distance between two neighboring points (in diagonal direction on the WRF-NMM's E-grid), ***dt*** is the time step, and ***c*** is the phase speed of the fastest process, the CFL criterion requires that:

(c*dt)/[d/sqrt(2.)] ≤1

This gives**:** **dt ≤ d/[sqrt(2.)*c]**

A very simple approach is to use ***2.25 x (grid spacing in km)*** or about ***330 x (angular grid spacing)*** to obtain an integer number of time steps per hour.

For example: If the grid spacing of the coarsest grid is 12km, then this gives *dt=27 s,* with a *dt=26 2/3 s* corresponding to 135 time steps per hour.

The following are pre-tested time-steps for WRF-NMM:

Approximate Grid Spacing (km)	**DELTA_X (in degrees)**	**DELTA_Y (in degrees)**	**Time Step (seconds)**
4	0.026726057	0.026315789	9-10s
8	0.053452115	0.052631578	18s
10	0.066666666	0.065789474	24s
12	0.087603306	0.075046904	25-30s
22	0.154069767	0.140845070	60s

Approximate Grid Spacing (km)	DELTA_X (in degrees)	DELTA_Y (in degrees)	Time Step (seconds)
32	0.222222222	0.205128205	90s

e_we and e_sn: Given WRF-NMM's E-grid staggering, the end index in the east-west direction **(*e_we*)** and the south-north direction (***e_sn*)** for the coarsest grid need to be set with care and the ***e_sn*** value must be **EVEN** for WRF-NMM.

When using WPS, the coarsest grid dimensions should be set as:

e_we (namelist.input) = e_ew (namelist.wps),
e_sn (namelist.input) = e_sn (namelist.wps).

For example: The parent grid ***e_we*** and ***e_sn*** are set up as follows:

namelist.input	***namelist.wps***
e_we = 124,	***e_we*** = 124,
e_sn = 202,	***e_sn*** = 202,

Other than what was stated above, there are no additional rules to follow when choosing ***e_we*** and ***e_sn*** for nested grids.

dx and dy: For WRF-NMM, ***dx*** and ***dy*** are the horizontal grid spacing in degrees, rather than meters (unit used for WRF-ARW). Note that ***dx*** should be slightly larger than ***dy*** due to the convergence of meridians approaching the poles on the rotated grid. The grid spacing in ***namelist.input*** should have the same values as in ***namelist.wps***.

When using WPS,

dx (namelist.input) = dx (namelist.wps),
dy (namelist.input) = dy (namelist.,wps).

When running a simulation with multiple (*N*) nests, the namelist should have *N* values of ***dx***, ***dy***, ***e_we***, ***e_sn*** separated by commas.

For more information about the horizontal grid spacing for WRF-NMM, please see Chapter 2, WRF Preprocessing System (WPS).

nio_tasks_per_group: The number of ***I/O*** tasks ***(nio_tasks_per_group)*** should evenly divide into the number of compute tasks in the ***J-direction*** on the grid (that is the value of ***nproc_y***). For example, if there are 6 compute tasks in the ***J-direction***, then ***nio_tasks_per_group*** could legitimately be set to 1, 2, 3, or 6. The user needs to use a number large enough that the quilting for a given output time is finished before the next

output time is reached. If one had 6 compute tasks in the ***J-direction*** (and the number in the ***I-direction*** was similar), then one would probably choose either 1 or 2 quilt tasks.

The following table provides an overview of the parameters specified in ***namelist.input***. Note that "***namelist.input***" is common for both WRF cores (WRF-ARW and WRF-NMM). Most of the parameters are valid for both cores. However, some parameters are only valid for one of the cores. Core specific parameters are noted in the table. In addition, some physics options have not been tested for WRF-NMM. Those options that have been tested are highlighted by indicating whether they have been "fully" or "preliminarily" tested for WRF-NMM.

Please note that the list is not exhaustive. You will find more info on the most recent WRF-ARW Users Guide. This is geared more towards the HWRF system.

Variable Names	**Value (Example)**	**Description**
&time_control		Time control
start_year	2014	Four digit year of starting time
start_month	10	Two digit month of starting time
start_day	14	Two digit day of starting time
start_hour	12	Two digit hour of starting time
start_minute	00	Two digit minute of starting time
start_second	00	Two digit second of starting time
end_year	2014	Four digit year of ending time
end_month	10	Two digit month of ending time
end_day	19	Two digit day of ending time
end_hour	18	Two digit hour of ending time
end_minute	00	Two digit minute of ending time
end_second	00	Two digit second of ending time **Note:** All end times also control when the nest domain integrations end. **Note:** All start and end times are used by ***real_nmm.exe***. One may use either ***run_days***/***run_hours*** etc. or ***end_year***/***month***/***day***/***hour*** etc. to control the length of model integration, but ***run_days***/***run_hours*** takes precedence over the end times. The program ***real_nmm.exe*** uses start and end times only.
interval_seconds	21600	Time interval between incoming real data,

Variable Names	Value (Example)	Description
		which will be the interval between the lateral boundary condition files. This parameter is only used by ***real_nmm.exe***.
history_interval (max_dom)	180	History output file interval in minutes
auxhist1_interval	60	interval in minutes for the output
auxhist12_interval	60	interval in minutes for the output
auxhist13_interval	180	interval in minutes for the output
history_end	540	End of History output (in minutes)
auxhist2_end	540	End of auxhist2 output (in minutes)
auxhist1_outname	wrfdiag_d<domain>	file name to write additional output to a different unit or output stream.. If not specified, auxhist1_d<domain>_<date> will be used.
auxhist2_outname	wrfout_d<domain>_<date>	file name to write additional output to a different unit or output stream.. If not specified, auxhist2_d<domain>_<date> will be used.
auxhist3_outname	wrfdiag_d<domain>_<date>	file name to write additional output to a different unit or output stream.. If not specified, auxhist3_d<domain>_<date> will be used.
frames_per_auxhist[1-3] (max_dom)	1	Output times per aadditional output file, used to split output files into smaller pieces
frames_per_outfile (max_dom)	1	Output times per history output file, used to split output files into smaller pieces
analysis	.false.	**This flag is only for the HWRF configuration.** True: Nested domain will read in initial conditions from a file (instead of being initialized by interpolation from the coarse domain). False: Nested domain will get its initial condition by interpolation from the coarse domain. Will output an analysis file containing the variables with restart IO characteristics for the nested domain.

Variable Names	Value (Example)	Description
restart	.false.	Logical indicating whether run is a restart run [Not supported with HWRF]
restart_interval	60	Restart output file interval in minutes
reset_simulation_start	F	Whether to overwrite simulation_start_date with forecast start time
io_form_history	2	Format of history file wrfout 1 = binary format (no supported post-processing software available) 2 = netCDF; 102 = split netCDF files on per processor (no supported post-processing software for split files) 4 = PHDF5 format (no supported post-processing software available) 5 = GRIB 1 10 = GRIB 2 11 = Parallel netCDF
io_form_restart	2	Format of restart file wrfrst 2 = netCDF; 102 = split netCDF files on per processor (must restart with the same number of processors) 11 = Parallel netCDF
io_form_input	2	Format of input file wrfinput_d01 2 = netCDF 11 = Parallel netCDF
io_form_boundary	2	Format of boundary file wrfbdy_d01 2 = netCDF 11 = Parallel netCDF
io_form_auxhist[4,5,6]	11	
io_form_auxinput2	2	IO format for input stream 2 data
auxinput1_inname	***met_nmm_.d01.***<*date*>	Name of input file from WPS
debug_level	1	Control for amount of debug printouts 0 - for standard runs, no debugging. 1 - netcdf error messages about missing fields. 50,100,200,300 values give increasing prints. Large values trace the job's progress through physics and time steps.
nocolons	.T.	when set to .true. this replaces the colons with

Variable Names	Value (Example)	Description
		underscores in the output file names
tg_reset_stream	1	Stream number of history stream that resets Tornado Genesis products
&domains		Domain definition
time_step	30	Time step for integration of coarsest grid in integer seconds
time_step_fract_num	0	Numerator for fractional coarse grid time step
time_step_fract_den	1	Denominator for fractional coarse grid time step. Example, if you want to use 60.3 sec as your time step, set ***time_step***=60, ***time_step_fract_num***=3, and ***time_step_fract_den***=10
max_dom	3	Number of domains (1 for a single grid, >1 for nests)
s_we (max_dom)	1	Start index in x (west-east) direction (leave as is)
e_we (max_dom)	124	End index in x (west-east) direction (staggered dimension)
s_sn (max_dom)	1	Start index in y (south-north) direction (leave as is)
e_sn (max_dom)	62	End index in y (south-north) direction (staggered dimension). **For WRF-NMM this value must be even.**
s_vert (max_dom)	1	Start index in z (vertical) direction (leave as is)
e_vert (max_dom)	61	End index in z (vertical) direction (staggered dimension). This parameter refers to full levels including surface and top. Note: Vertical dimensions need to be the same for all nests.
dx (max_dom)	.0534521	Grid length in x direction, units in **degrees** for WRF-NMM.
dy (max_dom)	.0526316	Grid length in y direction, units in **degrees** for WRF-NMM.
p_top_requested	5000	P top used in the model (Pa); must be available in WPS data
grid_id	1	Domain number, for parent id grid_id=1

Variable Names	Value (Example)	Description
ptsgm	15000.	Pressure level (Pa) in which the WRF-NMM hybrid coordinate transitions from sigma to pressure
eta_levels	1.00, 0.99, …0.00	Model eta levels. If this is not specified ***real_nmm.exe*** will provide a set of levels.
num_metgrid_levels	40	Number of vertical levels in the incoming data: type ***ncdump –h*** to find out
num_metgrid_soil_levels	4	number of soil levels or layers in WPS output (type ncdump -h on one of the met_em* files to find out this number)
p_top_requested	200.0	Model top pressure level (Pa).
use_prep_hybrid	T	Use GFS spectral files for IC and BC (T/F)
parent_id (max_dom)	0	ID of the parent domain. Use 0 for the coarsest grid.
i_parent_start (max_dom)	1	Defines the LLC of the nest as this I-index of the parent domain. Use 1 for the coarsest grid.
j_parent_start (max_dom)	1	Defines the LLC of the nest in this J-index of the parent domain. Use 1 for the coarsest grid.
parent_grid_ratio (max_dom)	3	Parent-to-nest domain grid size ratio. **For WRF-NMM this ratio must be 3.**
parent_time_step_ratio (max_dom)	3	Parent-to-nest time step ratio. **For WRF-NMM this ratio must be 3.**
feedback	1	Feedback from nest to its parent domain; 0 = no feedback
smooth_option	0	no smoothing
	1	1-2-1 smoothing option for parent domain; used only with `feedback=1`
	2	(default) smoothing-desmoothing option for parent domain; used only with `feedback=1`
num_moves	-99	0: Stationary nest -99: Vortex-following moving nest throughout the entire simulation **This flag is only for the HWRF configuration.**
tile_sz_x	0	Number of points in tile x direction.
tile_sz_y	0	Number of points in tile y direction.
numtiles	1	Number of tiles per patch (alternative to above

Variable Names	Value (Example)	Description
		two items).
nproc_x	-1	Number of processors in x–direction for decomposition.
nproc_y	-1	Number of processors in y-direction for decomposition: If -1: code will do automatic decomposition. If >1 for both: will be used for decomposition.
coral_x	6	Sets the minimum distance from its parent's edge (units: parent gridpoints) for each nests
coral_y	6	Sets the minimum distance from its parent's edge (units: parent gridpoints) for each nests
&physics		Physics options
mp_physics (max_dom)	5	Microphysics options: See above
ra_lw_physics (max_dom)	99	Long-wave radiation options: See above
ra_sw_physics (max_dom)	99	Short-wave radiation options: See above
nrads (max_dom)	100	**This flag is only for the WRF-NMM core**. Number of fundamental time steps between calls to shortwave radiation scheme. NCEP's operational setting: ***nrads*** is on the order of "3600/dt". For more detailed results, use: ***nrads=1800/dt***
nradl (max_dom)	100	**This flag is only for the WRF-NMM core**. Number of fundamental time steps between calls to longwave radiation scheme. Note that ***nradl*** must be set equal to ***nrads***.
tprec (max_dom)	3	**This flag is only for the WRF-NMM core**. Number of hours of precipitation accumulation in WRF output.
theat (max_dom)	6	**This flag is only for the WRF-NMM core**. Number of hours of accumulation of gridscale and convective heating rates in WRF output.
tclod (max_dom)	6	**This flag is only for the WRF-NMM core**. Number of hours of accumulation of cloud amounts in WRF output.

Variable Names	Value (Example)	Description
trdsw (max_dom)	6	**This flag is only for the WRF-NMM core**. Number of hours of accumulation of shortwave fluxes in WRF output.
trdlw (max_dom)	6	**This flag is only for the WRF-NMM core**. Number of hours of accumulation of longwave fluxes in WRF output.
tsrfc (max_dom)	6	**This flag is only for the WRF-NMM core**. Number of hours of accumulation of evaporation/sfc fluxes in WRF output.
pcpflg (max_dom)	.false.	**This flag is only for the WRF-NMM core**. Logical switch that turns on/off the precipitation assimilation used operationally at NCEP.
co2tf	1	**This flag is only for the WRF-NMM core.** Controls CO2 input used by the GFDL radiation scheme. 0: Read CO2 functions data from pre- generated file 1: Generate CO2 functions data internally
sf_sfclay_physics (max_dom)	2	Surface-layer options: See above
iz0tlnd	0	Thermal roughness length for sfclay and myjsfc (0 - old, 1 - veg dependent Czil)
sf_surface_physics (max_dom)	99	Land-surface options: See above
bl_pbl_physics (max_dom)	2	Boundary-layer options: See above
nphs (max_dom)	10	**This flag is only for WRF-NMM core**. Number of fundamental time steps between calls to turbulence and microphysics. It can be defined as: ***nphs=x/dt***, where ***dt*** is the time step (s), and ***x*** is typically in the range of 60s to 180s. (Traditionally it has been ***an even number***, which may be a consequence of portions of horizontal advection only being called every other time step.)
topo_wind (max_dom)	0	1=turn on topographic surface wind correction (Jimenez); requires extra input from geogrid, and works with YSU PBL scheme only (0 = off, default)

Variable Names	Value (Example)	Description
bl_mynn_tkebudget	1	adds MYNN tke budget terms to output
cu_physics (max_dom)	2	Cumulus scheme options: See above
mommix	0.7	momentum mixing coefficient (used in SAS cumulus scheme). **This flag is for the SAS scheme only.**
h_diff	0.1	Horizontal diffusion coefficient. **This flag is only for the HWRF configuration.**
sfenth	1.0	Enthalpy flux factor. **This flag is for the GFDL surface scheme only.**
ncnvc (max_dom)	10	**This flag is only for WRF-NMM core**. Number of fundamental time steps between calls to convection. ***Note that ncnvc should be set equal to nphs.***
isfflx	1	heat and moisture fluxes from the surface for real-data cases and when a PBL is used (only works with `sf_sfclay_physics=1, 5, 7, or 11`) 1 = fluxes are on 0 = fluxes are off
ifsnow	1	Snow-cover effects for "Thermal Diffusion scheme" (sf_surface_physics=1): 0. No snow-cover effect 1. With snow-cover effect
icloud	0	Cloud effect to the optical depth in the Dudhia shortwave (ra_sw_physics=1) and RRTM longwave radiation (ra_lw_physics=1) schemes. 0. No cloud effect 1. With cloud effect 3. With partial cloudiness with RRTMG schemes
swrad_scat	1	Scattering tuning parameter (default 1 is 1.e-5 m^2/kg) (only for *ra_sw_physics* = 1)
num_soil_layers	4	Number of soil layers in land surface model. For more information consult the recent WRF-ARE Users Guide
maxiens	1	Grell-Devenyi and G3 only. **Note:** For more information consult the recent WRF-ARE Users Guide

Variable Names	Value (Example)	Description
mp_zero_out	0	**For WRF-NMM, mp_zero_out MUST BE set to 0.**
gwd_opt	0	Gravity wave drag option; use with grid spacing > 10 km 0. Off (default) 1. ARW GWD on 2. NMM GWD on
sst_update	0	Option to use time-varying SST, seaice, vegetation fraction, and abledo during a model simulation (set before running ***real_nmm.exe***) For more information consult the recent WRF-ARE Users Guide
sas_pgcon	0.55	convectively forced pressure gradient factor (SAS schemes 4 and 94)
gfs_alpha	1	boundary depth factor for GFS PBL scheme (3)
var_ric	0	Placeholder for the use of variable critical Richardson number (Ric) in GFS PBL scheme. Default var_ric=0 to use constant Ric, else set var_ric=1 to use variable.
coef_ric_l	0.16	Placeholder for the coefficient used in the calculation of the variable critical Richardson number (Ric) in GFS PBL scheme.
coef_ric_s	0.25	Placeholder for the coefficient used in the calculation of the variable critical Richardson number (Ric) in GFS PBL scheme.
sas_mass_flux	0.5	mass flux limit (SAS scheme 84)
vortex_tracker	1	Vortex Tracking Algorithm for HWRF 1. (default) follow vortex using MSLP 2. follow vortex using MSLP (revised) 3. track vortex in nest and use that result to move this domain 4. follow vortex using storm centroid 5. follow vortex using dynamic pressure 6. follow vortex using the tracking algorithm of the GFDL vortex tracker 7. Improves tracking for small eyes
nomove_freq:	-1	Disable nest movement at certain intervals to prevent noise in the output files, so that nest will not move at analysis time or multiples of this interval, if this interval is set to a positive

Variable Names	Value (Example)	Description
		number.
movemin	5	Frequency with which nest tracker routine will be called in HWRF (multiples of nphs) It is the number of nphs (physics) steps between track timesteps. Where track timesteps are how often the grid tracking routine run to potentially move the nest.
Lcurr_sf	F	Option to include ocean currents in the surface flux calculations
icoef	4	Option for exchange coefficients in the surface flux scheme
iwavecpl	0	Option for activate coupling to sea surface wave model
&noah_mp		For more information consult the recent WRF-ARE Users Guide
ens_random_seed	99	Ensemble random generator seed
ens_sasamp	50	SAS perturbation amplitude (Units: hPa)
ens_pblamp	0.2	SAS perturbation amplitude (Units 100*%)
pert_pbl	F	If T, pert PBL
pert_sas	F	If T, pert SAS
pert_cd	F	If T, pert Cd
&dynamics		Dynamics options:
non_hydrostatic	.true.	Whether running the model in hydrostatic or non-hydrostatic model.
euler_adv	.F.	Logical switch that turns on/off passive advection (new in v3.2 – **Compatible with all Ferrier MP, else set to .false.**)
codamp	6.4	Divergence damping weighting factor (larger = more damping)
coac	1.6	Horizontal diffusion weighting factor (larger = more diffusion)
wp	0.15	Off-centering weight in the updating of nonhyrostatic eps

Variable Names	Value (Example)	Description
dw_dt_damping_lev	1500	specify the non-hydro dw/dt damping level in stratosphere (in Pa), 0: no damping
&bdy_control		Boundary condition control.
spec_bdy_width	1	Total number of rows for specified boundary value nudging. **It MUST be set to 1 for WRF-NMM core.**
specified (max_dom)	.true.	Specified boundary conditions (only applies to domain 1)
&namelist_quilt		Option for asynchronized I/O for MPI applications.
poll_servers		Logical flag to turn on an option to poll for an idle i/o server rather than using the next one in sequence. This can be useful when files of different sizes are written to different streams, but only applies when there are multiple (>1) i/o groups available.
nio_tasks_per_group	0	Default value is 0, means no quilting; value > 0 quilting I/O
nio_groups	1	Default is 1. May be set higher for nesting IO, or history and restart IO.
&logging		
compute_tasks_silent	T	Switch to enable (compute_tasks_silent=.false.) or disable (compute_tasks_silent=.true.) the wrf_message calls on the task nodes (where the wrf_dm_on_monitor() =.false.)
io_servers_silent	T	Switch to enable (io_servers_silent=.false.) or disable (io_servers_silent=.true.) the wrf_message calls on the IO servers.
stderr_logging	0	Switch to enable (stderr_logging=1) or disable (stderr_logging=0) the output of stderr.

How to Run WRF for the NMM Core

Note: Running a real-data case requires first successfully running the WRF Preprocessing System (WPS) (See HWRF Users' Guide for directions for installing the WPS and Chapter 2 for a description of the WPS and how to run the package for the HWRF system).

Running *wrf.exe*:

Note: Running ***wrf.exe*** requires a successful run of ***real_nmm.exe*** as explained in Chapter 3.

1. If the working directory used to run ***wrf.exe*** is different than the one used to run ***real_nmm.exe***, make sure ***wrfinput_d01*** and ***wrfbdy_d01,*** as well as the files listed above in the ***real_nmm.exe*** discussion, are in your working directory (you may link the files to this directory).

2. The command issued to run ***wrf.exe*** in the working directory will depend on the operating system:

   On LINUX-MPI systems, the command is:

DM parallel build:	or	Serial build:
mpirun -np n wrf.exe		***./wrf.exe >& wrf.out***

   where **"*n*"** defines the number of processors to use.

   For batch jobs on some IBM systems (such as NCAR's IBM), the command is:

   mpirun.lsf wrf.exe

   and for interactive runs (Interactive MPI job is not an option on NCAR IBMs), the command is:

   mpirun.lsf wrf.exe -rmpool 1 -procs n

where ***"n"*** stands for the number of processors (CPUs) to be used.

Checking *wrf.exe* output

A successful run of ***wrf.exe*** will produce output files with the following naming convention:

wrfout_d01_yyyy-mm-dd_hh:mm:ss

For example, the first output file for a run started at 0000 UTC, 23rd January 2005 would be:

wrfout_d01_2005-01-23_00:00:00

If multiple grids were used in the simulation, additional output files named
wrfout_d02_yyyy-mm-dd_hh:mm:ss
wrfout_d03_yyyy-mm-dd_hh:mm:ss
(…)
will be produced.

To check whether the run is successful, look for "SUCCESS COMPLETE WRF" at the end of the log file (e.g., ***rsl.out.0000***, ***wrf.out***).

The times written to an output file can be checked by typing:

ncdump -v Times wrfout_d01_2005-01-23_00:00:00

The number of ***wrfout*** files generated by a successful run of ***wrf.exe*** and the number of output times per ***wrfout*** file will depend on the output options specified in ***namelist.input*** (i.e***., frames_per_outfile*** and ***history interval***).

Restart Run

A restart run allows a user to extend a run to a longer simulation period. It is effectively a continuous run made of several shorter runs. Hence the results at the end of one or more restart runs should be identical to a single run without any restart. ***The restart capability for the HWRF system is not supported.***

In order to do a restart run, one must first create a restart file. This is done by setting namelist variable ***restart_interval*** (unit is in minutes) to be equal to or less than the simulation length in the first model run, as specified by ***run_ **** variables or ***start_ **** and ***end_ **** times. When the model reaches the time to write a restart file, a restart file named ***wrfrst_d***<*domain*>_<*date*> will be written. The date string represents the time when the restart file is valid.

When one starts the restart run, the ***namelist.input*** file needs to be modified so that the ***start_ **** time will be set to the restart time (which is the time the restart file is written). The other namelist variable that must be set is ***restart***, this variable should be set to ***.true.*** for a restart run.

In summary, these namelists should be modified:

start_*, end_: start and end times for restart model integration
restart: logical to indicate whether the run is a restart or not

Hint*:** Typically, the restart file is a lot bigger in size than the history file, hence one may find that even it is ok to write a single model history output time to a file in netCDF format (frame_per_outfile=1***), it may fail to write a restart file. This is because the basic netCDF file support is only 2Gb. There are two solutions to the problem. The first is to simply set namelist option ***io_form_restart = 102*** (instead of 2), and this will force the restart file to be written into multiple pieces, one per processor. As long as one restarts the model using the same number of processors, this option works well (and one should restart the model with the same number of processors in any case). The second solution is to recompile the code using the netCDF large file support option (see section on "Installing WRF" in this chapter).

Configuring a run with multiple domains

WRF-NMM supports stationary one-way (Gopalakrishnan et al. 2006) and two-way nesting. By setting the ***feedback*** switch in the ***namelist.input*** file to 0 or 1, the domains behave as one-way or two-way nests, respectively. The model can handle multiple domains at the same nest level (no overlapping nest), and/or multiple nest levels (telescoping). Make sure that you compile the code with nest options turned on. ***Please refer to the HWRF Users' Guide to see how the code should be compiled for the HWRF system. Also, refer to the HWRF Scientific Documentation to get more information on HWRF nest algorithm.***

The nest(s) can be located anywhere inside the parent domain as long as they are at least 5 parent grid points away from the boundaries of the parent grid. Similar to the coarsest domain, nests use an E-staggered grid with a rotated latitude-longitude projection. The horizontal grid spacing ratio between the parent and the nest is 1:3, and every third point of the nest coincides with a point in the parent domain. The time step used in the nest must be 1/3 that of the parent time step.

No nesting is applied in the vertical, that is, the nest has the same number of vertical levels as its parent. Note that, while the hybrid levels of the nest and parent in sigma space coincide, the nest and the parent do not have the same levels in pressure or height space. This is due to the differing topography, and consequently different surface pressure between the nest and the parent.

Nests can be introduced in the beginning of the model forecast or later into the run. Similarly, nests can run until the end of the forecast or can be turned off earlier in the run. Namelist variables ***start_ **** and ***end_ **** control the starting and ending time for nests.

When a nest is initialized, its topography is obtained from the static file created for that nest level by the WPS (see Chapter 2). Topography is the only field used from the static file. All other information for the nest is obtained from the lower-resolution parent

domain. Land variables, such as land-sea mask, SST, soil temperature and moisture are obtained through a nearest-neighbor approach.

To obtain the temperature, geopotential, and moisture fields for the nest initialization, the first step is to use cubic splines to vertically interpolate those fields from hybrid levels to constant pressure levels in each horizontal grid point of the parent grid. The second step is to bilinearly interpolate those fields in the horizontal from the parent grid to the nest. The third step is to use the high-resolution terrain and the geopotential to determine the surface pressure on the nest. Next, the pressure values in the nest hybrid surfaces are calculated. The final step is to compute the geopotential, temperature and moisture fields over the nest hybrid surface using a cubic spline interpolation in the vertical.

The zonal and meridional components of the wind are obtained by first performing a horizontal interpolation from the parent to the nest grid points using a bi-linear algorithm. The wind components are then interpolated in the vertical from the parent hybrid surfaces onto the nest hybrid surfaces using cubic splines.

The boundary conditions for the nest are updated at every time step of the parent domain. The outermost rows/columns of the nest are forced to be identical to the parent domain interpolated to the nest grid points. The third rows/columns are not directly altered by the parent domain, that is, their values are obtained from internal computations within the nest. The second rows/columns are a blend of the first and third rows/columns. This procedure is analogous to what is used to update the boundaries of the coarsest domain with the external data source. To obtain the values of the mass and momentum fields in the outermost row/column of the nest, interpolations from the parent grid to the nest are carried in the same manner as for nest initialization.

Most of options to start a nest run are handled through the namelist. **Note:** All variables in the ***namelist.input*** file that have multiple columns of entries need to be edited with caution.

The following are the key namelist variables to modify:

- ***start_*** and ***end_year/month/day/minute/second***: These control the nest start and end times
- ***history_interval***: History output file in minutes (integer only)
- ***frames_per_outfile***: Number of output times per history output file, used to split output files into smaller pieces
- ***max_dom***: Setting this to a number greater than 1 will invoke nesting. For example, if you want to have one coarse domain and one nest, set this variable to 2.
- ***e_we/e_sn***: Number of grid points in the east-west and north-south direction of the nest. In WPS, ***e_sw***. ***e_sn*** for the nest are specified to cover the entire domain of the coarse grid while, in file ***namelist.input***, ***e_we*** and ***e_sn*** for the nest are specified to cover the domain of the nest.
- ***e_vert***: Number of grid points in the vertical. No nesting is done in the vertical, therefore the nest must have the same number of levels as its parent.

- ***dx/dy***: grid spacing in **degrees**. The nest grid spacing must be 1/3 of its parent.
- ***grid_id***: The domain identifier will be used in the ***wrfout*** naming convention. The coarser grid must have ***grid_id*** = 1.
- ***parent_id***: Specifies the parent grid of each nest. The parents should be identified by their ***grid_id***.
- ***i_parent_start/j_parent_start***: Lower-left corner starting indices of the nest domain in its parent domain. The coarser grid should have ***parent_id*** = 1.
- ***parent_grid_ratio***: Integer parent-to-nest domain grid size ratio. **Note:** Must be 3 for the NMM.
- ***parent_time_step_ratio***: Integer parent-to-nest domain timestep ratio. **Note:** Must be 3 for the NMM. Since the timestep for the nest is determined using this variable, namelist variable ***time_step*** only assumes a value for the coarsest grid.
- ***feedback***: If feedback = 1, values of prognostic variables in the nest are fedback and overwrite the values in the coarse domain at the coincident points. 0 = no feedback.

In addition to the variables listed above, the following variables are used to specify physics options and need to have values for all domains (as many columns as domains: ***mp_physics, ra_lw_physics, ra_sw_physics, nrads, nradl, sf_sfclay_physics, sf_surface_physics, bl_pbl_physics, nphs, cu_physics, ncnvc***.

Note: It is recommended to run all domains with the same physics, the exception being the possibility of running cumulus parameterization in the coarser domain(s) but excluding it from the finer domain(s).

In case of doubt about whether a given variable accepts values for nested grids, search for that variable in the file ***WRFV3/Registry/Registry*** and check to see if the string ***max_doms*** is present in that line.

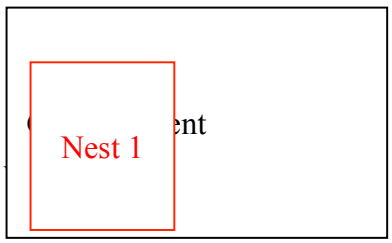
Before starting the WRF model, make sure to place the nest's time-invariant land-describing file in the proper directory.

For example, when using WPS, place the file ***geo_nmm_nest.l01.nc*** in the working directory where the model will be run. If more than one level of nest will be run, place additional files ***geo_nmm_nest.l02.nc***, ***geo_nmm_nest.l03.nc*** etc. in the working directory.

Examples:

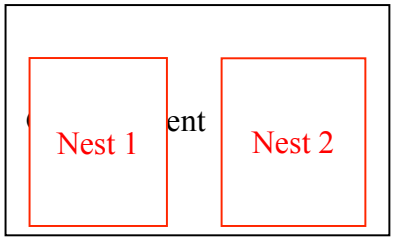
1. One nest and one level of nesting

WPS: requires file ***geo_nmm_nest.l01.nc***



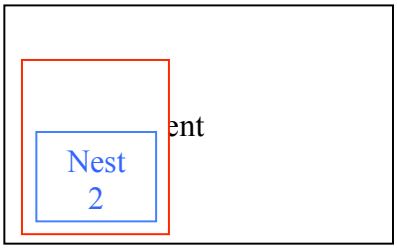
2. Two nests and one level of nesting

WPS: requires file ***geo_nmm_nest.l01.nc***



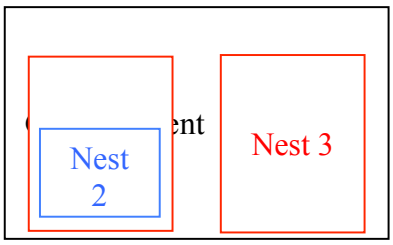
3. Two nests and two level of nesting

WPS: requires file ***geo_nmm_nest.l01.nc*** and ***geo_nmm_nest.l02.nc***

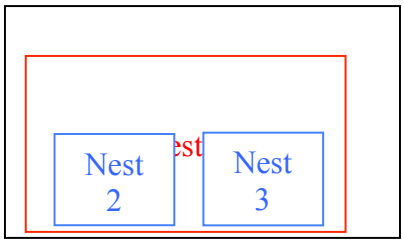


4. Three nests and two level of nesting

WPS: requires file ***geo_nmm_nest.l01.nc*** and ***geo_nmm_nest.l02.nc***



OR



After configuring the file ***namelist.input*** and placing the appropriate ***geo_nmm_nest**** file(s) in the proper directory(s), the WRF model can be run identically to the single domain runs described in Running wrf.exe.

Using Digital Filter Initialization

Digital filter initialization (DFI) is a new option in V3.2. It is a way to remove initial model imbalance as, for example, measured by the surface pressure tendency. This might be important when interested in the 0 – 6 hour simulation/forecast results. It runs a digital filter for a short model integration, backward and forward, and then starts the forecast. In the WRF implementation, this is all done in a single job. In the current release, DFI can only be used in a single domain run.

No special requirements are needed for data preparation. For a typical application, the following options are used in the ***namelist.input*** file:

dfi_opt = 3
dfi_nfilter = 7 (filter option: Dolph)
dfi_cutoff_seconds = 3600 (should not be longer than the filter window)

For time specification, it typically needs to integrate backward for 0.5 to 1 hour, and integrate forward for half of the time.

If option ***dfi_write_filtered_input*** is set to true, a filtered ***wrfinput*** file, ***wrfinput_initialized_d01***, will be produced.

If a different time step is used for DFI, one may use ***time_step_dfi*** to set it.

This works for NMM single domain. This capability is not tested for HWRF.

Using sst_update option

The WRF model physics does not predict sea-surface temperature, vegetation fraction, albedo and sea ice. For long simulations, the model provides an alternative to read in the time-varying data and update these fields. In order to use this option, one must have access to time-varying SST and sea ice fields. Twelve monthly values vegetation fraction and albedo are available from the ***geogrid*** program. Once these fields are processed via WPS, one may activate the following options in namelist record ***&time_control*** before running program ***real_nmm.exe*** and ***wrf.exe***:

sst_update = 1 in ***&physics***
io_form_auxinput4 = 2

auxinput4_inname = *"wrflowinp_d<domain>"* (created by ***real_nmm.exe***)
auxinput4_interval = 720,

Using IO Quilting

This option allows a few processors to be set alone to do output only. It can be useful and performance-friendly if the domain sizes are large, and/or the time taken to write a output time is getting significant when compared to the time taken to integrate the model in between the output times. There are two variables for setting the option:

nio_tasks_per_group: How many processors to use per IO group for IO quilting. Typically 1 or 2 processors should be sufficient for this purpose.

nio_groups: How many IO groups for IO. Default is 1.

Extended Reference List for WRF-NMM Dynamics and Physics

Aligo, E., Ferrier, B., J. Carley, E. Rodgers, M. Pyle, S. J. Weiss, and I. L. Jirak, 2014: Modified microphysics for use in high resolution NAM forecasts. 27 AMS Conference on Severe Local Storms. 3-7 November, Madison, WI.

Arakawa, A., and W. H. Schubert, 1974: Interaction of a cumulus cloud ensemble with the large scale environment. Part I. *J. Atmos. Sci*., **31**, 674-701.

Biswas M. K., Ligia Bernardet, Isaac Ginis, Young Kwon, Bin Liu, Qingfu Liu, Tim Marchok, Avichal Mehra, Kathryn Newman, Dmitry Sheinin, Subashini Subramanian, Vijay Tallapragada, Biju Thomas, Mingjing Tong, Samuel Trahan, Weiguo Wang, Richard Yablonsky and Xuejin Zhang, 2016: Hurricane Weather Research and Forecasting (HWRF) Model: 2016 Scientific Documentation.

Chen, F., Z. Janjic and K. Mitchell, 1997: Impact of atmospheric surface-layer parameterization in the new land-surface scheme of the NCEP mesoscale Eta model. *Boundary-Layer Meteorology*, **48**

Chen, S.-H., and W.-Y. Sun, 2002: A one-dimensional time dependent cloud model. *J. Meteor. Soc. Japan*, **80**, 99–118.

Chen, F., and J. Dudhia, 2001: Coupling an advanced land-surface/ hydrology model with the Penn State/ NCAR MM5 modeling system. Part I: Model description and implementation. *Mon. Wea. Rev*., **129**, 569–585.

Chou M.-D., and M. J. Suarez, 1994: An efficient thermal infrared radiation parameterization for use in general circulation models. NASA Tech. Memo. 104606, 3, 85pp.

Dudhia, J., 1989: Numerical study of convection observed during the winter monsoon experiment using a mesoscale two-dimensional model, *J. Atmos. Sci*., **46**, 3077–3107.

Ek, M. B., K. E. Mitchell, Y. Lin, E. Rogers, P. Grunmann, V. Koren, G. Gayno, and J. D. Tarpley, 2003: Implementation of NOAH land surface model advances in the NCEP operational mesoscale Eta model. *J. Geophys. Res*., **108**, No. D22, 8851, do1:10.1029/2002JD003296.

Fels, S. B., and M. D. Schwarzkopf, 1975: The simplified exchange approximation: A new method for radiative transfer calculations. *J. Atmos. Sci*., **32**, 1475-1488.

Ferrier, B. S., Y. Lin, T. Black, E. Rogers, and G. DiMego, 2002: Implementation of a new grid-scale cloud and precipitation scheme in the NCEP Eta model. Preprints, 15th Conference on Numerical Weather Prediction, San Antonio, TX, *Amer. Meteor. Soc*., 280-283.

Gopalakrishnan, S. G., N. Surgi, R. Tuleya and Z. Janjic, 2006. NCEP's Two-way-Interactive-Moving-Nest NMM-WRF modeling system for Hurricane Forecasting. *27th Conf. On Hurric. Trop. Meteor.* Available online at http://ams.confex.com/ams/27Hurricanes/techprogram/paper_107899.htm.

Grell, G. A., 1993: Prognostic Evaluation of Assumptions Used by Cumulus Parameterizations. *Mon. Wea. Rev*., **121**, 764-787.

Grell, G. A., and D. Devenyi, 2002: A generalized approach to parameterizing convection combining ensemble and data assimilation techniques. *Geophys. Res. Lett*., **29**(14), Article 1693.

Hong, S.-Y., J. Dudhia, and S.-H. Chen, 2004: A Revised Approach to Ice Microphysical Processes for the Bulk Parameterization of Clouds and Precipitation, *Mon. Wea. Rev*., **132**, 103–120.

Hong, S.-Y., H.-M. H. Juang, and Q. Zhao, 1998: Implementation of prognostic cloud scheme for a regional spectral model, *Mon. Wea. Rev*., **126**, 2621–2639.

Hong, S.-Y., and H.-L. Pan, 1996: Nonlocal boundary layer vertical diffusion in a medium-range forecast model, *Mon. Wea. Rev*., **124**, 2322–2339.

Janjic, Z. I., 1979: Forward-backward scheme modified to prevent two-grid-interval noise and its application in sigma coordinate models. *Contributions to Atmospheric Physics*, **52**, 69-84.

Janjic, Z. I., 1984: Non–linear advection schemes and energy cascade on semi–staggered grids. *Mon. Wea. Rev*, **112**, 1234–1245.

Janjic, Z. I., 1990: The step–mountain coordinates: physical package. *Mon. Wea. Rev*, **118**, 1429–1443.

Janjic, Z. I., 1994: The step–mountain eta coordinate model: further developments of the convection, viscous sublayer and turbulence closure schemes. *Mon. Wea. Rev*, **122**, 927–945.

Janjic, Z. I., 1996a: The Mellor-Yamada level 2.5 scheme in the NCEP Eta Model. 11th Conference on Numerical Weather Prediction, Norfolk, VA, 19-23 August 1996; *American Meteorological Society*, Boston, MA, 333-334.

Janjic, Z. I., 1996b: The Surface Layer in the NCEP Eta Model. 11th Conf. on NWP, Norfolk, VA, *American Meteorological Society*, 354–355.

Janjic, Z. I., 1997: Advection Scheme for Passive Substances in the NCEP Eta Model. Research Activities in Atmospheric and Oceanic Modeling, WMO, Geneva, CAS/JSC WGNE, 3.14.

Janjic, Z. I., 2000: Comments on "Development and Evaluation of a Convection Scheme for Use in Climate Models. *J. Atmos. Sci.*, **57**, p. 3686

Janjic, Z. I., 2001: Nonsingular Implementation of the Mellor-Yamada Level 2.5 Scheme in the NCEP Meso model. NCEP Office Note No. 437, 61 pp.

Janjic, Z. I., 2002a: A Nonhydrostatic Model Based on a New Approach. EGS XVIII, Nice France, 21-26 April 2002.

Janjic, Z. I., 2002b: Nonsingular Implementation of the Mellor–Yamada Level 2.5 Scheme in the NCEP Meso model, NCEP Office Note, No. 437, 61 pp.

Janjic, Z. I., 2003a: A Nonhydrostatic Model Based on a New Approach. *Meteorology and Atmospheric Physics*, **82**, 271-285. (Online: http://dx.doi.org/10.1007/s00703-001-0587-6).

Janjic, Z. I., 2003b: The NCEP WRF Core and Further Development of Its Physical Package. 5th International SRNWP Workshop on Non-Hydrostatic Modeling, Bad Orb, Germany, 27-29 October.

Janjic, Z. I., 2004: The NCEP WRF Core. 12.7, Extended Abstract, 20th Conference on Weather Analysis and Forecasting/16th Conference on Numerical Weather Prediction, Seattle, WA, American Meteorological Society.

Janjic, Z. I., J. P. Gerrity, Jr. and S. Nickovic, 2001: An Alternative Approach to Nonhydrostatic Modeling. *Mon. Wea. Rev.*, **129**, 1164-1178.

Janjic, Z. I., T. L. Black, E. Rogers, H. Chuang and G. DiMego, 2003: The NCEP Nonhydrostatic Meso Model (NMM) and First Experiences with Its Applications. EGS/EGU/AGU Joint Assembly, Nice, France, 6-11 April.

Janjic, Z. I, T. L. Black, E. Rogers, H. Chuang and G. DiMego, 2003: The NCEP Nonhydrostatic Mesoscale Forecasting Model. 12.1, Extended Abstract, 10th Conference on Mesoscale Processes, Portland, OR, American Meteorological Society. (Available Online).

Kain J. S. and J. M. Fritsch, 1990: A One-Dimensional Entraining/Detraining Plume Model and Its Application in Convective Parameterization. *J. Atmos. Sci*., **47**, No. 23, pp. 2784–2802.

Kain, J. S., and J. M. Fritsch, 1993: Convective parameterization for mesoscale models: The Kain-Fritcsh scheme, the representation of cumulus convection in numerical models, K. A. Emanuel and D.J. Raymond, Eds., *Amer. Meteor. Soc*., 246 pp.

Kain J. S., 2004: The Kain–Fritsch Convective Parameterization: An Update. *Journal of Applied Meteorology*, **43**, No. 1, pp. 170–181.

Kessler, E., 1969: On the distribution and continuity of water substance in atmospheric circulation, Meteor. Monogr., 32, Amer. Meteor. Soc., 84 pp.

Lacis, A. A., and J. E. Hansen, 1974: A parameterization for the absorption of solar radiation in the earth's atmosphere. J. Atmos. Sci., 31, 118–133.

Lin, Y.-L., R. D. Farley, and H. D. Orville, 1983: Bulk parameterization of the snow field in a cloud model. J. Climate Appl. Meteor., 22, 1065–1092.

Miyakoda, K., and J. Sirutis, 1986: Manual of the E-physics. [Available from Geophysical Fluid Dynamics Laboratory, Princeton University, P.O. Box 308, Princeton, NJ 08542]

Mlawer, E. J., S. J. Taubman, P. D. Brown, M. J. Iacono, and S. A. Clough, 1997: Radiative transfer for inhomogeneous atmosphere: RRTM, a validated correlated-k model for the longwave. J. Geophys. Res., 102 (D14), 16663–16682.

Pan, H.-L. and W.-S. Wu, 1995: Implementing a Mass Flux Convection Parameterization Package for the NMC Medium-Range Forecast Model. NMC Office Note, No. 409, 40pp. [Available from NCEP/EMC, W/NP2 Room 207, WWB, 5200 Auth Road, Washington, DC 20746-4304]

Pan, H-L. and L. Mahrt, 1987: Interaction between soil hydrology and boundary layer developments. *Boundary Layer Meteor*., **38**, 185-202.

Rutledge, S. A., and P. V. Hobbs, 1984: The mesoscale and microscale structure and organization of clouds and precipitation in midlatitude cyclones. XII: A diagnostic modeling study of precipitation development in narrow cloud-frontal rainbands. *J. Atmos. Sci*., **20**, 2949–2972.

Rogers, E., T. Black, B. Ferrier, Y. Lin, D. Parrish, and G. DiMego, 2001: Changes to the NCEP Meso Eta Analysis and Forecast System: Increase in resolution, new cloud microphysics, modified precipitation assimilation, modified 3DVAR analysis. Technical Procedures Bulletin. http://www.emc.ncep.noaa.gov/mmb/mmbpll/eta12tpb/.

Sadourny. R., 1975: The Dynamics of Finite-Difference Models of the Shallow-Water Equations. *J. Atmos. Sci.*, **32**, No. 4, pp. 680–689.

Schwarzkopf, M. D., and S. B. Fels, 1985: Improvements to the algorithm for computing CO2 transmissivities and cooling rates. *J. Geophys. Res*., **90**, 541-550.

Schwarzkopf, M. D., and S. B. Fels, 1991: The simplified exchange method revisited: An accurate, rapid method for computations of infrared cooling rates and fluxes. *J. Geophys. Res*., **96**, 9075-9096.

Skamarock, W. C., J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang and J. G. Powers, 2005: A Description of the Advanced Research WRF Version 2, NCAR Tech Note, NCAR/TN–468+STR, 88 pp. [Available from UCAR Communications, P.O. Box 3000, Boulder, CO, 80307]. Available on-line at: http://box.mmm.ucar.edu/wrf/users/docs/arw_v2.pdf)

Smirnova, T. G., J. M. Brown, and S. G. Benjamin, 1997: Performance of different soil model configurations in simulating ground surface temperature and surface fluxes. *Mon. Wea. Rev*., **125**, 1870–1884.

Smirnova, T. G., J. M. Brown, S. G. Benjamin, and D. Kim, 2000: Parameterization of cold season processes in the MAPS land-surface scheme. **J. Geophys. Res**., **105** (D3), 4077-4086.

Tallapragada, V and Co-authors, 2014: Hurricane Weather Research and Forecasting Model (HWRF): 2014 Scientific Documentation. http://www.dtcenter.org/HurrWRF/users/docs/scientific_documents/HWRFv3.6a_ScientificDoc.pdf

Tao, W.-K., J. Simpson, and M. McCumber 1989: An ice-water saturation adjustment, *Mon. Wea. Rev*., **117**, 231–235.

Troen, I. and L. Mahrt, 1986: A simple model of the atmospheric boundary layer: Sensitivity to surface evaporation. *Boundary Layer Meteor*., **37**, 129-148.

Thompson, G., R. M. Rasmussen, and K. Manning, 2004: Explicit forecasts of winter precipitation using an improved bulk microphysics scheme. Part I: Description and sensitivity analysis. *Mon. Wea. Rev*., **132**, 519–542.

Wicker, L. J., and R. B. Wilhelmson, 1995: Simulation and analysis of tornado development and decay within a three-dimensional supercell thunderstorm. *J. Atmos. Sci*., **52**, 2675–2703.

User's Guide for the NMM Core of the Weather Research and Forecast (WRF) Modeling System Version 3

Chapter 5: WRF Software

Table of Contents

WRF Build Mechanism

The WRF build mechanism provides a uniform apparatus for configuring and compiling the WRF model and pre-processors over a range of platforms with a variety of options. This section describes the components and functioning of the build mechanism. ***For information on building the HWRF code, please refer to HWRF Users' Guide.***

Required software

The WRF build relies on Perl (version 5 or later) and a number of UNIX utilities: csh and Bourne shell, make, M4, sed, awk, and the uname command. A C compiler is needed to compile programs and libraries in the tools and external directories. The WRF code itself is mostly standard Fortran (and uses a few 2003 capabilities). For distributed-memory processing, MPI and related tools and libraries should be installed.

Build Mechanism Components

Directory structure: The directory structure of WRF consists of the top-level directory plus directories containing files related to the WRF software framework (***frame***), the WRF model (***dyn_em***, ***dyn_nmm***, ***phys***, ***share***), WRF-Var (***da***), configuration files ***(arch, Registry***), helper and utility programs (***tools***), and packages that are distributed with the WRF code (***external***).

Scripts: The top-level directory contains three user-executable scripts: ***configure***, ***compile***, and ***clean***. The configure script relies on a Perl script in ***arch/Config_new.pl***.

Programs: A significant number of WRF lines of code are automatically generated at compile time. The program that does this is ***tools/registry*** and it is distributed as part of the source code with the WRF model.

Makefiles: The main ***makefile*** (input to the UNIX make utility) is in the top-level directory. There are also makefiles in most of the subdirectories that come with WRF. Make is called recursively over the directory structure. Make is not directly invoked by the user to compile WRF; the ***compile*** script is provided for this purpose. The WRF build has been structured to allow "parallel make". Before the compile command, the user sets an environment variable, J, to the number of processors to use. For example, to use two processors (in csh syntax):
setenv J "-j 2"
On some machines, this parallel **make** causes troubles (a typical symptom is a missing **mpif.h** file in the frame directory). The user can force that only a single processor to be used with the command:
setenv J "-j 1"

Configuration files: The ***configure.wrf*** contains compiler, linker, and other build settings, as well as rules and macro definitions used by the make utility. The ***configure.wrf*** file is included by the Makefiles in most of the WRF source distribution (Makefiles in ***tools*** and ***external*** directories do not include ***configure.wrf***). The ***configure.wrf*** file in the top-level directory is generated each time the configure script is invoked. It is also deleted by ***clean -a.*** Thus, ***configure.wrf*** is the place to make temporary changes, such as optimization levels and compiling with debugging, but permanent changes should be made in the file ***arch/configure_new.defaults***. The ***configure.wrf*** file is composed of three files: **arch/preamble_new** ***arch/postamble_new*** and ***arch_configure_new.defaults***.

The ***arch/configure_new.defaults*** file contains lists of compiler options for all the supported platforms and configurations. Changes made to this file will be permanent. This file is used by the configure script to generate a temporary ***configure.wrf*** file in the top-level directory. The ***arch*** directory also contains the files ***preamble_new*** and ***postamble_new***, which constitute the generic parts (non-architecture specific) of the ***configure.wrf*** file that is generated by the configure script.

The ***Registry*** directory contains files that control many compile-time aspects of the WRF code. The files are named ***Registry.****core*, where *core* is either ***EM*** (for builds using the Eulerian Mass (ARW) core), ***NMM*** (for builds using the NMM core). The configure script copies one of these to ***Registry/Registry***, which is the file that ***tools/registry*** will use as input. The choice of coredepends on settings to the **configure** script. Changes to ***Registry/Registry*** will be lost; permanent changes should be made to ***Registry.****core*. For the WRF NMM model, the file is typically **Registry.NMM**.

Environment variables: Certain aspects of the configuration and build are controlled by environment variables: the non-standard locations of NetCDF libraries or the Perl command, which dynamic core to compile, machine-specific features, and optional build libraries (such as Grib Edition 2, HDF, and parallel netCDF).

In addition to WRF-related environment settings, there may also be settings specific to particular compilers or libraries. For example, local installations may require setting a variable like ***MPICH_F90*** to make sure the correct instance of the Fortran 90 compiler is used by the ***mpif90*** command.

How the WRF build works

There are two steps in building WRF: configuration and compilation.

Configuration: The ***configure*** script configures the model for compilation on your system. The configuration first attempts to locate needed libraries such as NetCDF or HDF and tools such as Perl. It will check for these in normal places, or will use settings from the user's shell environment. The configuration file then calls the UNIX ***uname*** command to discover what platform you are compiling on. It then calls the Perl script ***arch/Config_new.pl***, which traverses the list of known machine configurations and displays a list of available options to the user. The selected set of options is then used to create the ***configure.wrf*** file in the top-level directory. This file may be edited but changes are temporary since the file will be deleted by ***clean –a*** or overwritten by the next invocation of the ***configure*** script. The only typical option that is included on the ***configure*** command is "***-d***" (for debug). The code builds relatively quickly and has the debugging switches enabled, but the model will run very slowly since all of the optimization has been deactivated. This script takes only a few seconds to run.

Compilation: The ***compile*** script is used to compile the WRF code after it has been configured using the ***configure*** script. This ***csh*** script performs a number of checks, constructs an argument list, copies to ***Registry/Registry*** the correct ***Registry.****core* file for the core being compiled, and invokes the UNIX make command in the top-level directory. The core to be compiled is determined from the user's environment. For example, to set it for WRF-NMM core the "***setenv WRF_NMM_CORE 1"*** and "***setenv WRF_NMM_NEST 1"*** commands should be issued. If no core is specified in the environment (by setting **WRF_*core*_CORE** to `1`) the default core is selected (currently the Eulerian Mass core for ARW). The ***makefile*** in the top-level directory directs the rest of the build, accomplished as a set of recursive invocations of make in the subdirectories of WRF. Most of these makefiles include the ***configure.wrf*** file from the top-level directory. The order of a complete build is as follows:

1. Make in ***external*** directory

    a. make in ***external/io_{grib1, grib_share, int, netcdf}*** for Grib Edition 1, binary, and NetCDF implementations of I/O API

b. make in ***RSL_LITE*** directory to build communications layer (DM_PARALLEL only)

c. make in ***external/esmf_time_f90*** directory to build ESMF time manager library

d. make in ***external/fftpack*** directory to build FFT library for the global filters

e. make in other external directories as specified by "external:" target in the ***configure.wrf*** file

2. Make in the ***tools*** directory to build the program that reads the ***Registry/Registry*** file and auto-generates files in the ***inc*** directory

3. Make in the ***frame*** directory to build the WRF framework specific modules

4. Make in the ***share*** directory to build the non-core-specific mediation layer routines, including WRF I/O modules that call the I/O API

5. Make in the ***phys*** directory to build the WRF model layer routines for physics (non core-specific)

6. Make in the ***dyn_"****core****"*** directory for core-specific mediation-layer and model-layer subroutines

7. Make in the ***main*** directory to build the main programs for WRF and symbolically link to create executable files (location depending on the build case that was selected as the argument to the compile script (e.g. compile ***em_real*** or compile ***nmm_real***)

Source files (.F and, in some of the external directories, .F90) are preprocessed to produce .f90 files, which are input to the compiler. As part of the preprocessing, Registry-generated files from the ***inc*** directory may be included. Compiling the .f90 files results in the creation of object (***.o***) files that are added to the library ***main/libwrflib.a***. Most of the ***external*** directories generate their own library file. The linking step produces the ***wrf.exe*** and ***real_nmm.exe*** executables.

The .o files and .f90 files from a compile are retained until the next invocation of the ***clean*** script. The .f90 files provide the true reference for tracking down run time errors that refer to line numbers or for sessions using interactive debugging tools such as dbx or gdb. ***For information on building the HWRF code, please refer to HWRF Users' Guide.***

Registry

Tools for automatic generation of application code from user-specified tables provide significant software productivity benefits in development and maintenance of large

applications such as WRF. Just for the WRF model, hundreds of thousands of lines of WRF code are automatically generated from a user-edited table, called the Registry. The Registry provides a high-level single-point-of-control over the fundamental structure of the model data, and thus provides considerable utility for developers and maintainers. It contains lists describing state data fields and their attributes: dimensionality, binding to particular solvers, association with WRF I/O streams, communication operations, and run time configuration options (namelist elements and their bindings to model control structures). Adding or modifying a state variable to WRF involves modifying a single line of a single file; this single change is then automatically propagated to scores of locations in the source code the next time the code is compiled.

The WRF Registry has two components: the Registry file (which the user may edit) and the Registry program.

The Registry file is located in the ***Registry*** directory and contains the entries that direct the auto-generation of WRF code by the Registry program. There is more than one Registry in this directory, with filenames such as ***Registry.EM*** (for builds using the Eulerian Mass/ARW core) and ***Registry.NMM*** (for builds using the NMM core). The WRF Build Mechanism copies one of these to the file ***Registry/Registry*** and this file is used to direct the Registry program. The syntax and semantics for entries in the Registry are described in detail in "WRF Tiger Team Documentation: The Registry" at: *http://www2.mmm.ucar.edu/wrf/WG2/Tigers/Registry/***.**
The Registry program is distributed as part of WRF in the ***tools*** directory. It is built automatically (if necessary) when WRF is compiled. The executable file is ***tools/registry***. This program reads the contents of the Registry file, ***Registry/Registry***, and generates files in the ***inc*** directory. These include files are inserted (with ***cpp #include*** commands) into WRF Fortran source files prior to compilation. Additional information on these is provided as an appendix to "WRF Tiger Team Documentation: The Registry". The Registry program itself is written in C. The source files and ***makefile*** are in the ***tools*** directory.

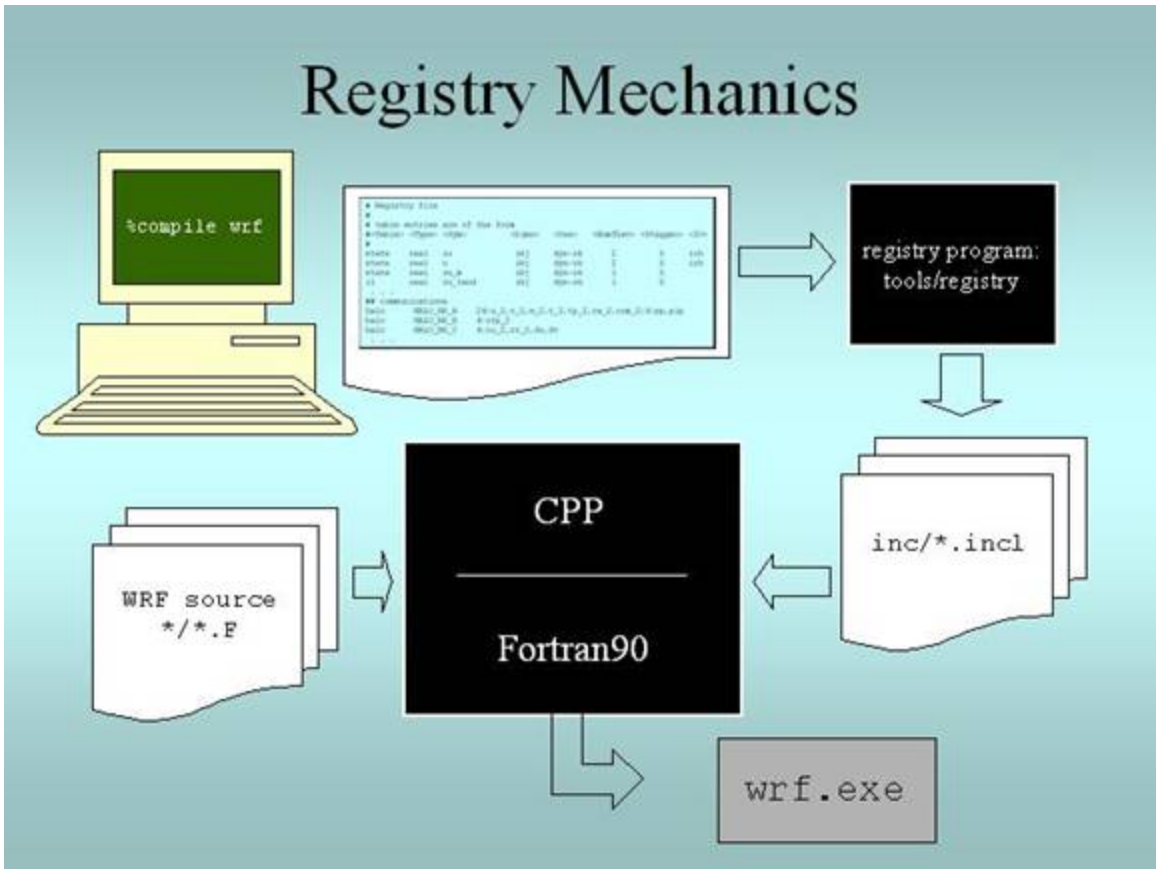


Figure 5.1: When the user compiles WRF, the Registry Program reads ***Registry/Registry***, producing auto-generated sections of code that are stored in files in the ***inc*** directory. These are included into WRF using the CPP preprocessor and the Fortran compiler.

In addition to the WRF model itself, the ***Registry/Registry*** file is used to build the accompanying preprocessors such as ***real_nmm.exe*** (for real data simulations).

Every variable that is an input or an output field is described in the Registry. Additionally, every variable that is required for parallel communication, specifically associated with a physics package, or needs to provide a tendency to multiple physics or dynamics routines is contained in the Registry. For each of these variables, the index ordering, horizontal and vertical staggering, feedback and nesting interpolation requirements, and the associated IO are defined. For most users, to add a variable into the model requires, regardless of dimensionality, only the addition of a single line to the Registry (make sure that changes are made to the correct **Registry.*core*** file, as changes to the ***Registry*** file itself are overwritten). Since the Registry modifies code for compile-time options, any change to the Registry REQUIRES that the code be returned to the original unbuilt status with the ***clean –a*** command.

The other very typical activity for users is to define new run-time options, which are handled via a Fortran namelist file ***namelist.input*** in WRF. As with the model state arrays and variables, the entire model configuration is described in the Registry. As with

the model arrays, adding a new namelist entry is as easy as adding a new line in the Registry.

While the model state and configuration are by far the most commonly used features in the Registry, the data dictionary has several other powerful uses. The Registry file provides input to generate all of the communications for the distributed memory processing (halo interchanges between patches, support for periodic lateral boundaries, and array transposes for FFTs to be run in the X, Y, or Z directions). The Registry associates various fields with particular physics packages, so that the memory footprint reflects the actual selection of the options, not a maximal value.

Together, these capabilities allow a large portion of the WRF code to be automatically generated. Any code that is automatically generated relieves the developer of the effort of coding and debugging that portion of software. Usually, the pieces of code that are suitable candidates for automation are precisely those that are fraught with "hard to detect" errors, such as communications, indexing, and IO which must be replicated for hundreds of variables.

Registry Syntax

Each entry in the Registry is for a specific variable, whether it is for a new dimension in the model, a new field, a new namelist value, or even a new communication. For readability, a single entry may be spread across several lines with the traditional "\" at the end of a line to denote that the entry is continuing. When adding to the Registry, most users find that it is helpful to copy an entry that is similar to the anticipated new entry, and then modify that Registry entry. The Registry is not sensitive to spatial formatting. White space separates identifiers in each entry.

Note: Do not simply remove an identifier and leave a supposed token blank, use the appropriate default value (currently a dash character "-").

Registry Entries

The WRF Registry has the following types of entries (not case dependent):

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
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

These ***keywords*** appear as the first word in a line of the file ***Registry*** to define which type of information is being provided. Following are examples of the more likely Registry types that users will need to understand

Registry Dimspec

The first set of entries in the Registry is the specifications of the dimensions for the fields to be defined. To keep the WRF system consistent between the dynamical cores and Chemistry, a unified ***registry.dimspec*** file is used (located in the ***Registry*** directory). This single file is included into each Registry file, with the keyword ***include***. In the example below, three dimensions are defined: i, j, and k. If you do an "***ncdump -h***" on a WRF file, you will notice that the three primary dimensions are named as "***west_east***", "***south_north***", and "***bottom_top***". That information is contained in this example (the example is broken across two lines, but interleaved).

#<Table> <Dim> <Order> <How defined>
dimspec i 1 standard_domain
dimspec j 3 standard_domain
dimspec k 2 standard_domain

<Coord-axis> <Dimname in Datasets>
x west_east
y south_north
z bottom_top

The WRF system has a notion of horizontal and vertical staggering, so the dimension names are extended with a **"*_stag***" suffix for the staggered sizes. The list of names in the <Dim> column may either be a single unique character (for release 3.0.1.1 and prior), or the <Dim> column may be a string with no embedded spaces (such as ***my_dim***). When this dimension is used later to dimension a **`state`** or **`i1`** variable, it must be surrounded by curly braces (such as **`{`*my_dim*`}`**). This <Dim> variable is not case specific, so for example "**i**" is the same as an entry for "**I**".

Registry State and I1

A ***state*** variable in WRF is a field that is eligible for IO and communications, and exists for the duration of the model forecast. The ***I1*** variables (intermediate level one) are typically thought of as tendency terms, computed during a single model time-step, and then discarded prior to the next time-step. The space allocation and de-allocation for these ***I1*** variables is automatic (on the stack for the model solver). In this example, for readability, the column titles and the entries are broken into multiple interleaved lines, with the user entries in a ***bold font***.

Some fields have simple entries in the ***Registry*** file. The following is a **state** variable that is a Fortran type ***real***. The name of the field inside the WRF model is ***u_gc***. It is a three dimension array (***igj***). It has a single time level, and is staggered in the ***X*** and ***Z***

directions. This field is input only to the real program (***i1***). On output, the netCDF name is ***UU***, with the accompanying description and units provided.

#<Table> <Type> <Sym> <Dims>
state real u_gc ijg

<Use> <NumTLev> <Stagger> <IO>
dyn_nmm 1 Z i1

<DNAME> <DESCRIP> <UNITS>
"UU" "x-wind component" "m s-1"

If a variable is not staggered, a "-" (dash) is inserted instead of leaving a blank space. The same dash character is required to fill in a location when a field has no IO specification. The variable description and units columns are used for post-processing purposes only; this information is not directly utilized by the model.

When adding new variables to the ***Registry*** file, users are warned to make sure that variable names are unique. The <Sym> refers to the variable name inside the WRF model, and it is not case sensitive. The <DNAME> is quoted, and appears exactly as typed. Do not use imbedded spaces. While it is not required that the <Sym> and <DNAME> use the same character string, it is highly recommended. The `<DESCRIP>` and the `<UNITS>` are optional, however they are a good way to supply self-documentation to the Registry. Since the `<DESCRIP>` value is used in the automatic code generation, restrict the variable description to 40 characters or less.

From this example, we can add new requirements for a variable. Suppose that the variable to be added is not specific to any dynamical core. We would change the <Use> column entry of ***dyn_nmm*** to ***misc*** (for miscellaneous). The ***misc*** entry is typical of fields used in physics packages. Only dynamics variables have more than a single time level, and this introductory material is not suitable for describing the impact of multiple time periods on the registry program. For the <Stagger> option, users may select any subset from ***{X, Y, Z}*** or ***{-}***, where the dash character "**-**" signifies "no staggering".

The <IO> column handles file input and output, and it handles the nesting specification for the field. The file input and output uses three letters: ***i*** (input), ***r*** (restart), and ***h*** (history). If the field is to be in the input file to the model, the restart file from the model, and the history file from the model, the entry would be ***irh***. To allow more flexibility, the input and history fields are associated with streams. The user may specify a digit after the ***i*** or the ***h*** token, stating that this variable is associated with a specified stream (***1*** *through* ***9***) instead of the default (***0***). A single variable may be associated with multiple streams. Once any digit is used with the ***i*** or ***h*** tokens, the default ***0*** stream must be explicitly stated. For example, <IO> entry ***i*** and <IO> entry ***i0*** are the same. However, <IO> entry ***h1*** outputs the field to the first auxiliary stream, but does not output the field to the default history stream. The <IO> entry ***h01*** outputs the field to both the default history stream and the first auxiliary stream.

Nesting support for the model is also handled by the <IO> column. The letters that are parsed for nesting are: ***u*** (*up* as in feedback up), ***d*** (*down*, as in downscale from coarse to fine grid), ***f*** (*forcing*, how the lateral boundaries are processed), and **s** (*smoothing*). As with other entries, the best coarse of action is to find a field nearly identical to the one that you are inserting into the ***Registry*** file, and copy that line. The user needs to make the determination whether or not it is reasonable to smooth the field in the area of the coarse grid, where the fine-grid feeds back to the coarse grid. Variables that are defined over land and water, non-masked, are usually smoothed. The lateral boundary forcing is primarily for dynamics variables, and is ignored in this overview. For non-masked fields (such as wind, temperature, pressure), the downward interpolation (controlled by ***d***) and the feedback (controlled by ***u***) use default routines. Variables that are land fields (such as soil temperature ***TSLB***) or water fields (such as sea ice ***XICE***) have special interpolators, as shown in the examples below (again, interleaved for readability):

#<Table> <Type> <Sym> <Dims>
state real TSLB ilj
state real XICE ij

<Use> <NumTLev> <Stagger>
misc 1 Z
misc 1 -

<IO>
i02rhd=(interp_mask_land_field:lu_index)u=(copy_fcnm) i0124rhd=(interp_mas k_water_field:lu_index)u=(copy_fcnm)

<DNAME> <DESCRIP> <UNITS>
"TSLB" "SOIL TEMPERATURE" "K"
"SEAICE" "SEA ICE FLAG" ""

Note that the ***d*** and ***u*** entries in the <IO> section are followed by an "=" then a parenthesis enclosed subroutine, and a colon separated list of additional variables to pass to the routine. It is recommended that users follow the existing pattern: ***du*** for non-masked variables, and the above syntax for the existing interpolators for masked variables.

Registry Rconfig

The ***Registry*** file is the location where the run-time options to configure the model are defined. Every variable in the WRF namelist is described by an entry in the ***Registry*** file. The default value for each of the namelist variables is as assigned in the Registry. The standard form for the entry for two namelist variables is given (broken across lines and interleaved):

#<Table> <Type> <Sym>
rconfig integer run_days
rconfig integer start_year

<How set> <Nentries> <Default>
namelist,time_control 1 0
namelist,time_control max_domains 1993

The keyword for this type of entry in the ***Registry*** file is ***rconfig*** (run-time configuration). As with the other model fields (such as ***state*** and ***i1***), the <Type> column assigns the Fortran kind of the variable: ***integer***, ***real***, or ***logical***. The name of the variable in the WRF namelist is given in the <Sym> column, and is part of the derived data type structure as are the ***state*** fields. There are a number of Fortran namelist records in the file ***namelist.input***. Each namelist variable is a member of one of the specific namelist records. The previous example shows that ***run_days*** and ***start_year*** are both members of the ***time_control*** record. The <Nentries> column refers to the dimensionality of the namelist variable (number of entries). For most variables, the <Nentries> column has two eligible values, either ***1*** (signifying that the scalar entry is valid for all domains) or ***max_domains*** (signifying that the variable is an array, with a value specified for each domain). Finally, a default value is given. This permits a namelist entry to be removed from the ***namelist.input*** file if the default value is acceptable.

The registry program constructs two subroutines for each namelist variable, one to retrieve the value of the namelist variable, and the other to set the value. For an integer variable named ***my_nml_var***, the following code snippet provides an example of the easy access to the namelist variables.

INTEGER :: my_nml_var, dom_id
CALL nl_get_my_nml_var (dom_id , my_nml_var)

The subroutine takes two arguments. The first is the input integer domain identifier (for example, ***1*** for the most coarse grid, **2** for the second domain), and the second argument is the returned value of the namelist variable. The associated subroutine to set the namelist variable, with the same argument list, is ***nl_set_my_nml_var***. For namelist variables that are scalars, the grid identifier should be set to ***1***.

The ***rconfig*** line may also be used to define variables that are convenient to pass around in the model, usually part of a derived configuration (such as the number of microphysics species associated with a physics package). In this case, the <How set> column entry is ***derived***. This variable does not appear in the namelist, but is accessible with the same generated ***nl_set*** and ***nl_get*** subroutines.

Registry Halo, Period, and Xpose

The distributed memory, inter-processor communications are fully described in the ***Registry*** file. An entry in the Registry constructs a code segment which is included (with ***cpp***) in the source code. Following is an example of a ***halo*** communication (split across two lines and interleaved for readability).

#<Table> <CommName> <Core>
halo HALO_NMM_K dyn_nmm

<Stencil:varlist>
8:q2;24:t,u,v,q,w,z

The keyword is ***halo***. The communication is named in the <CommName> column, so that it can be referenced in the source code. The entry in the <CommName> column is case sensitive (the convention is to start the name with ***HALO_NMM***). The selected dynamical core is defined in the <Core> column. There is no ambiguity, as every communication in each ***Registry*** file will have the exact same <Core> column option. The last set of information is the <Stencil:varlist>. The portion in front of the "**:**" is the stencil size, and the comma-separated list afterwards defines the variables that are communicated with that stencil size. Different stencil sizes are available, and are "**;**" separated in the same <Stencil:varlist> column. The stencil sizes ***8***, ***24***, ***48*** all refer to a square with an odd number of grid cells on a side, with the center grid cell removed (***8*** = 3x3-1, ***24*** = 5x5-1, ***48*** = 7x7-1). The special small stencil ***4*** is just a simple north, south, east, west communication pattern.

The convention in the WRF model is to provide a communication immediately after a variable has been updated. The communications are restricted to the mediation layer (an intermediate layer of the software that is placed between the framework level and the model level. The model level is where developers spend most of their time. The majority of users will insert communications into the ***dyn_nmm/solve_m,m.F*** subroutine. The ***HALO_NMM_K*** communication defined in the ***Registry*** file, in the example above, is activated by inserting a small section of code that includes an automatically generated code segment into the solve routine, via standard ***cpp*** directives.

#ifdef DM_PARALLEL
 # include "HALO_NMM_K.inc"
#endif

The parallel communications are only required when the WRF code is built for distributed-memory parallel processing, which accounts for the surrounding ***#ifdef***.

The ***period*** communications are required when periodic lateral boundary conditions are selected (ARW only). The Registry syntax is very similar for ***period*** and ***halo*** communications, but the stencil size refers to how many grid cells to communicate, in a direction that is normal to the periodic boundary.

#<Table> <CommName> <Core> <Stencil:varlist>

period PERIOD_EM_COUPLE_A dyn_em 2:mub,mu_1,mu_2

The **xpose** (a data transpose) entry is used when decomposed data is to be re-decomposed (ARW only). This is required when doing FFTs in the x-direction for polar filtering, for example. No stencil size is necessary.

#<Table> <CommName> <Core> <Varlist>
xpose XPOSE_POLAR_FILTER_T dyn_em t_2,t_xxx,dum_yyy

It is anticipated that many users will add to the the parallel communications portion of the Registry file (***halo*** and ***period***. It is unlikely that users will add ***xpose*** fields.

Registry Package

The ***package*** option in the ***Registry*** file associates fields with particular physics packages. Presently, it is mandatory that all 4-D arrays be assigned. Any 4-D array that is not associated with the selected physics option at run-time is either allocated, used for IO, nor communicated. All other 2-D and 3-D arrays are eligible for use with a ***package*** assignment, but that is not required.

The purpose of the ***package*** option is to allow users to reduce the memory used by the model, since only "necessary" fields are processed. An example for a microphysics scheme is given below.

#<Table> <PackageName> <NMLAssociated> <Variables>
package kesslerscheme mp_physics==1 - moist:qv,qc,qr

The entry keyword is ***package***, and is associated with the single physics option listed under <NMLAssociated>. The package is referenced in the code in Fortran ***IF*** and ***CASE*** statements by the name given in the <PackageName> column, instead of the more confusing and typical ***IF (mp_physics == 1)*** approach. The <Variables> column must start with a dash character and then a blank "**- "** (for historical reasons of backward compatibility). The syntax of the <Variables> column then is a 4-D array name, followed by a colon, and then a comma-separated list of the 3-D arrays constituting that 4-D amalgamation. In the example above, the 4-D array is ***moist***, and the selected 3-D arrays are ***qv***, ***qc***, and ***qr***. If more than one 4-D array is required, a "**;**" separates those sections from each other in the <Variables> column.

In addition to handling 4-D arrays and their underlying component 3-D arrays, the ***package*** entry is able to associate generic ***state*** variables, as shown in the example following. If the namelist variable ***use_wps_input*** is set to ***1***, then the variables ***u_gc*** and ***v_gc*** are available to be processed.

#<Table> <PackageName> <NMLAssociated> <Variables>
package realonly use_wps_input==1 - state:u_gc,v_gc

I/O Applications Program Interface (I/O API)

The software that implements WRF I/O, like the software that implements the model in general, is organized hierarchically, as a "software stack" **(http://www2.mmm.ucar.edu/wrf/WG2/Tigers/IOAPI/IOStack.html)**. From top (closest to the model code itself) to bottom (closest to the external package implementing the I/O), the I/O stack looks like this:

- Domain I/O (operations on an entire domain)
- Field I/O (operations on individual fields)
- Package-neutral I/O API
- Package-dependent I/O API (external package)

The lower-levels of the stack, associated with the interface between the model and the external packages, are described in the I/O and Model Coupling API specification document on http://www2.mmm.ucar.edu/wrf/WG2/Tigers/IOAPI/index.html.

Timekeeping

Starting times, stopping times, and time intervals in WRF are stored and manipulated as Earth System Modeling Framework (ESMF, *http://www.earthsystemmodeling.org/esmf_releases/non_public/ESMF_5_2_0p1/ESMF_refdoc/node6.html#SECTION06040000000000000000*) time manager objects. This allows exact representation of time instants and intervals as integer numbers of years, months, hours, days, minutes, seconds, and fractions of a second (numerator and denominator are specified separately as integers). All time computations involving these objects are performed exactly, by using integer arithmetic, with the result that there is no accumulated time step drift or rounding, even for fractions of a second.

The WRF implementation of the ESMF Time Manger is distributed with WRF in the ***external/esmf_time_f90*** directory. This implementation is entirely Fortran90 (as opposed to the ESMF implementation in C++) and it is conformant to the version of the ESMF Time Manager API that was available in 2009.

WRF source modules and subroutines that use the ESMF routines do so by use-association of the top-level ESMF Time Manager module, ***esmf_mod***:

USE esmf_mod

The code is linked to the library file ***libesmf_time.a*** in the ***external/esmf_time_f90*** directory.

ESMF timekeeping is set up on a domain-by-domain basis in the routine setup_timekeeping (***share/set_timekeeping.F***). Each domain keeps its own clocks and alarms. Since the time arithmetic is exact there is no problem with clocks on separate domains getting out of synchronization.

Software Documentation

Detailed and comprehensive documentation aimed at WRF software is available at *http://www2.mmm.ucar.edu/wrf/WG2/software_2.0*.

Performance

Benchmark information is available at http://www2.mmm.ucar.edu/wrf/WG2/benchv3/