Chapter 3: WRF Preprocessing System (WPS)

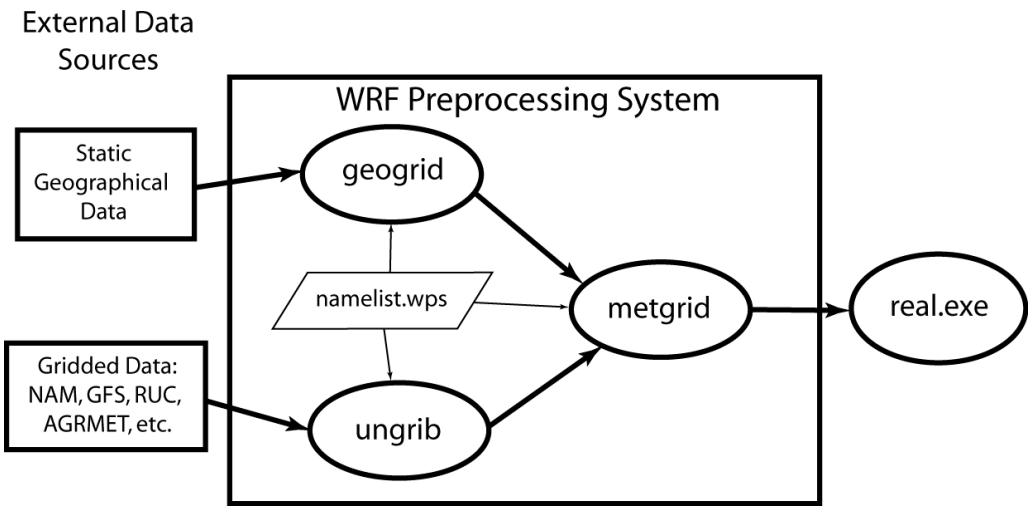
Table of Contents

- Introduction
- Function of Each WPS Program
- Installing the WPS
- Running the WPS
- Creating Nested Domains with the WPS
- Selecting Between USGS and MODIS-based Land Use Classifications
- Selecting Static Data for the Gravity Wave Drag Scheme
- Using Multiple Meteorological Data Sources
- Using Non-isobaric Meteorological Datasets
- Alternative Initialization of Lake SSTs
- Parallelism in the WPS
- Checking WPS Output
- WPS Utility Programs
- Writing Meteorological Data to the Intermediate Format
- Required Meteorological Fields for Running WRF
- Using MPAS Output for WRF Initial and Laterial Boundary Conditions
- Creating and Editing Vtables
- Writing Static Data to the Geogrid Binary Format
- Creating an Urban Fraction Field from NLCD Data
- Description of Namelist Variables
- Description of GEOGRID.TBL Options
- Description of index Options
- Description of METGRID.TBL Options
- Available Interpolation Options in Geogrid and Metgrid
- Land Use and Soil Categories in the Static Data
- WPS Output Fields

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 program 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' operations, 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, though for now, the user need not be concerned with them.

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 MPI 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.

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 to 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 Geographical Static Data Downloads page (https://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html), 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 as a "full-resolution" download and as a "low-resolution" download. Generally, the "low-resolution" sources for static fields are suitable only for code testing and educational purposes, and any applications where model accuracy are of concern should use the "full-resolution" geographical datasets.

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, NCL, and RIP4.

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 on 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, including the new version of RIP4.

Installing the WPS

The WRF Preprocessing System uses a build mechanism similar to that used by the WRF model. External libraries for geogrid and metgrid are limited to those required by the WRF model, since the WPS uses the WRF model's implementations of the WRF I/O API; consequently, *WRF must be compiled prior to installation of the WPS* so that the I/O API libraries in the WRF external directory will be available to WPS programs. Additionally, the ungrib program requires three compression libraries for GRIB Edition 2 support; however, if support for GRIB2 data is not needed, ungrib can be compiled without these compression libraries. For environment and library compatibility tests, as well as step-by-step instructions for building libraries and the WPS programs, see the How to Compile WRF (https://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation_tutorial.php) page.

Required Libraries

The only library required to build the WRF model is NetCDF. Usera can find the source code, precompiled binaries, and documentation at the UNIDATA home page (http://www.unidata.ucar.edu/software/netcdf/). Most users will select the NetCDF I/O option for WPS due to the easy access to utility programs that support the NetCDF data format, and before configuring the WPS, users should ensure that the environment variable NETCDF is set to the path of the NetCDF installation.

Where WRF adds a software layer between the model and the communications package, the WPS programs geogrid and metgrid make MPI calls directly. Most multi-processor machines come preconfigured with a version of MPI, so it is unlikely that users will need to install this package by themselves.

Three libraries are required by the ungrib program for GRIB Edition 2 compression support. Users are encouraged to engage their system administrators for the installation of these packages so that traditional library paths and include paths are maintained. Paths to user-installed compression libraries are handled in the `configure.wps` file by the `COMPRESSION_LIBS` and `COMPRESSION_INC` variables. As an alternative to manually editing the `COMPRESSION_LIBS` and `COMPRESSION_INC` variables in the `configure.wps` file, users may set the environment variables `JASPERLIB` and `JASPERINC` to the directories holding the JasPer library and include files *before configuring the WPS*; for example, if the JasPer libraries were installed in `/usr/local/jasper-1.900.1`, one might use the following commands (in csh or tcsh):

```
> setenv JASPERLIB /usr/local/jasper-1.900.1/lib
> setenv JASPERINC /usr/local/jasper-1.900.1/include
```

If the zlib and PNG libraries are not in a standard path that will be checked automatically by the compiler, the paths to these libraries can be added on to the JasPer environment

variables; for example, if the PNG libraries were installed in `/usr/local/libpng-1.2.29` and the zlib libraries were installed in `/usr/local/zlib-1.2.3`, one might use

```
      > setenv JASPERLIB "${JASPERLIB} -L/usr/local/libpng-1.2.29/lib -
L/usr/local/zlib-1.2.3/lib"
      > setenv JASPERINC "${JASPERINC} -I/usr/local/libpng-
1.2.29/include -I/usr/local/zlib-1.2.3/include"
```

after having previously set `JASPERLIB` and `JASPERINC`.

1) JasPer (an implementation of the JPEG2000 standard for "lossy" compression)
http://www.ece.uvic.ca/~mdadams/jasper/
Go down to "JasPer software", one of the "click here" parts is the source.

```
      > ./configure
      > make
      > make install
```

Note: The GRIB2 libraries expect to find include files in "jasper/jasper.h", so it may be necessary to manually create a "jasper" subdirectory in the "include" directory created by the JasPer installation, and manually link header files there.

2) PNG (compression library for "lossless" compression)
http://www.libpng.org/pub/png/libpng.html
Scroll down to "Source code" and choose a mirror site.

```
      > ./configure
      > make check
      > make install
```

3) zlib (a compression library used by the PNG library)
http://www.zlib.net/
Go to "The current release is publicly available here" section and download.

```
      > ./configure
      > make
      > make install
```

To get around portability issues, the NCEP GRIB libraries, w3 and g2, have been included in the WPS distribution. The original versions of these libraries are available for download from NCEP at http://www.nco.ncep.noaa.gov/pmb/codes/GRIB2/. The specific tar files to download are g2lib and w3lib. Because the ungrib program requires modules from these files, they are not suitable for usage with a traditional library option during the link stage of the build.

Required Compilers and Scripting Languages

The WPS requires the same Fortran and C compilers as were used to build the WRF model, since the WPS executables link to WRF's I/O API libraries. After executing the

`./configure` command in the WPS directory, a list of supported compilers on the current system architecture are presented.

WPS Installation Steps

- Download the `WPS.TAR.gz` file and unpack it at the same directory level as WRF, as shown below.

```
> ls
-rw-r--r--  1  563863 WPSV4.0.TAR.gz
drwxr-xr-x 18    4096 WRF

> gzip -d WPSV4.0.TAR.gz

> tar xf WPSV4.0.TAR

> ls
drwxr-xr-x  7     4096 WPS
-rw-r--r--  1  3491840 WPSV4.0.TAR
drwxr-xr-x 18     4096 WRF
```

- At this point, a listing of the current working directory should at least include the directories WRF and WPS. First, compile WRF (see the instructions for installing WRF in Chapter 2). Then, after the WRF executables are generated, change to the WPS directory and issue the configure command followed by the compile command as below.

```
> cd WPS

> ./configure
```

  - Choose one of the configure options

```
> ./compile >& compile.output
```

- After issuing the compile command, a listing of the current working directory should reveal symbolic links to executables for each of the three WPS programs: geogrid.exe, ungrib.exe, and metgrid.exe. If any of these links do not exist, check the compilation output in compile.output to see what went wrong.

```
> 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
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   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
```

Running the WPS

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.

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
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
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   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 = 'ARW',
 max_dom = 2,
 start_date = '2008-03-24_12:00:00','2008-03-24_12:00:00',
 end_date   = '2008-03-24_18:00:00','2008-03-24_12:00:00',
 interval_seconds = 21600,
 io_form_geogrid = 2
/

&geogrid
 parent_id         =   1,   1,
 parent_grid_ratio =   1,   3,
 i_parent_start    =   1,  31,
 j_parent_start    =   1,  17,
 e_we              =  74, 112,
 e_sn              =  61,  97,
 geog_data_res     = 'default','default',
 dx = 30000,
 dy = 30000,
 map_proj = 'lambert',
 ref_lat   = 34.83,
 ref_lon   = -81.03,
 truelat1  =  30.0,
 truelat2  =  60.0,
 stand_lon = -98.,
 geog_data_path = '/mmm/users/wrfhelp/WPS_GEOG/'
/
```

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'`. After selecting the dynamical core, the total number of domains (in the case of ARW) or nesting levels (in the case of NMM) must be chosen with `max_dom`. Since geogrid produces only time-independent data, the `start_date`, `end_date`, and `interval_seconds` variables are ignored by geogrid. Optionally, a location (if not the default, which is the current working directory) where domain files should be written to 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`.

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. Each of the four possible map projections in the ARW are shown graphically in the full-page figure below, and the namelist variables used to set the parameters of the projection are summarized in the following table.

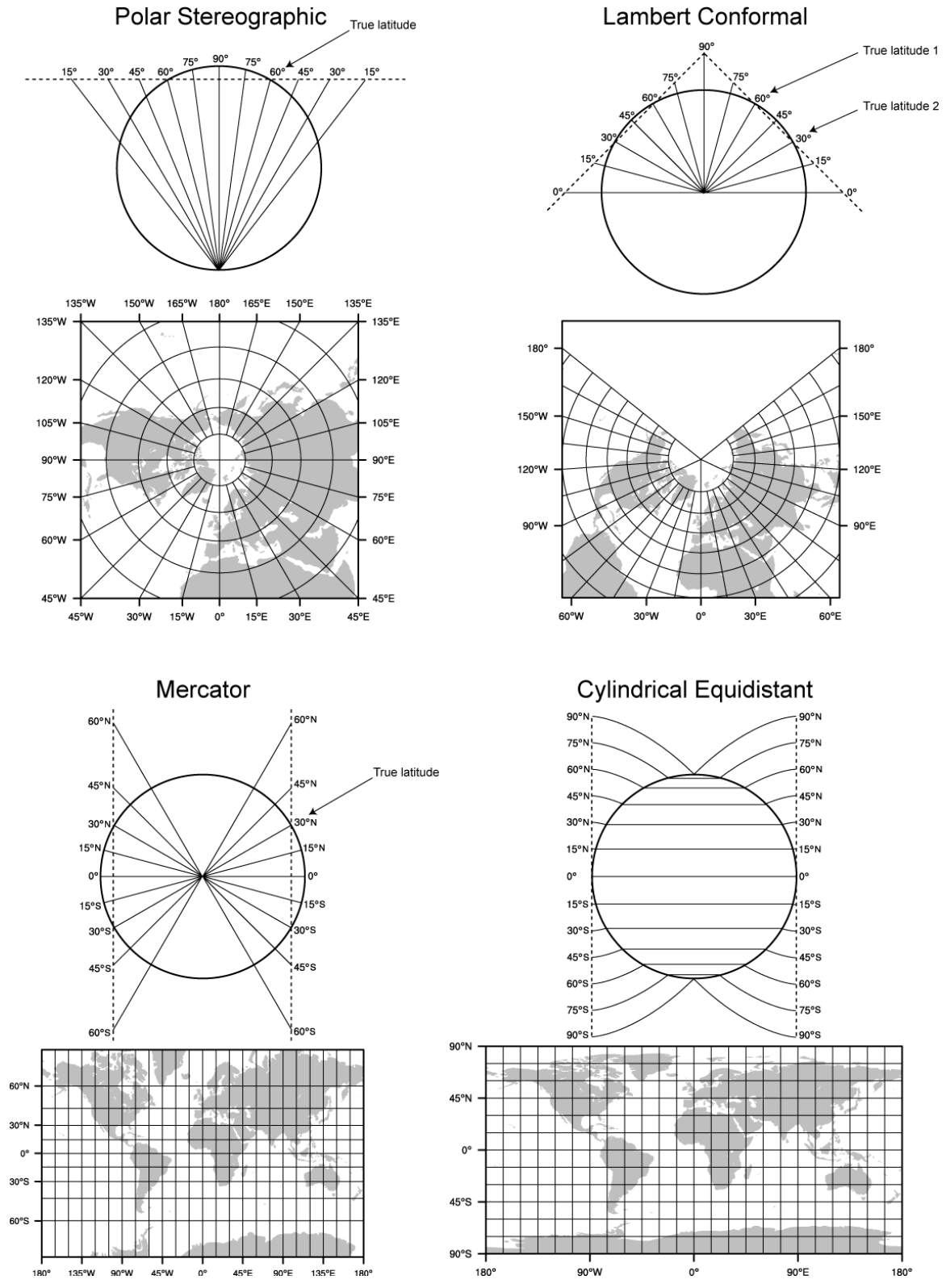
Map projection / value of `map_proj`	Projection parameters

`Lambert Conformal / 'lambert'`	`truelat1`
	`truelat2` (optional)
	`stand_lon`

Mercator / `'mercator'`	`truelat1`
Polar stereographic / `'polar'`	`truelat1` `stand_lon`

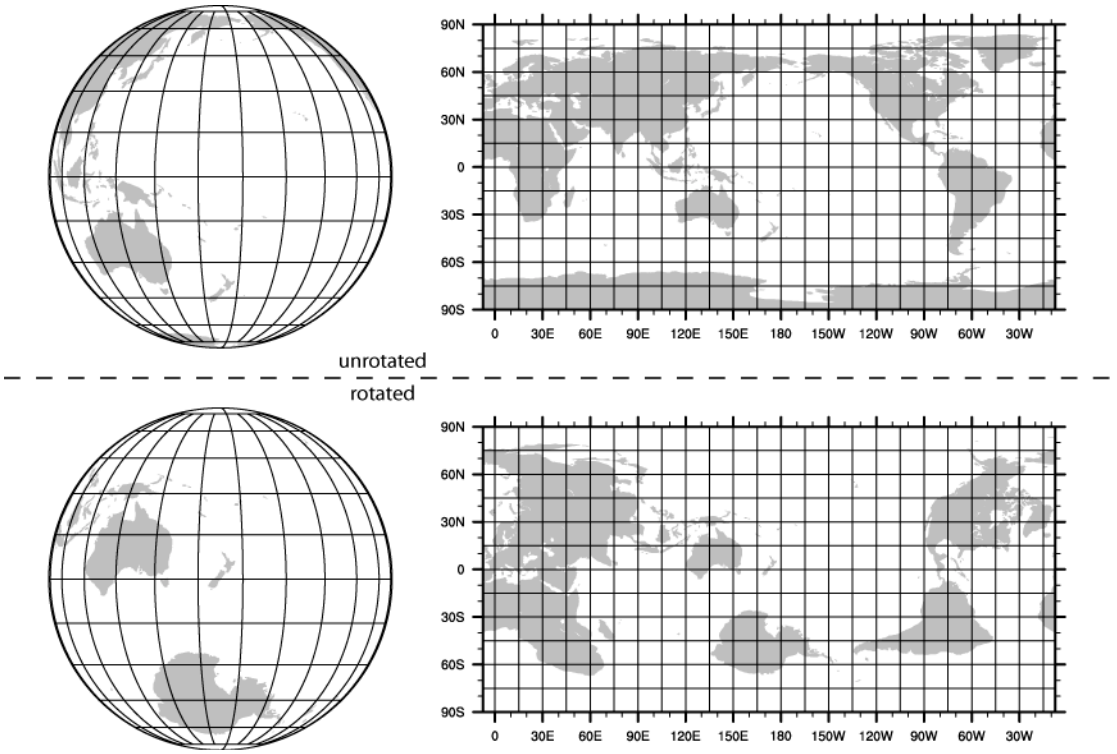
Regular latitude-longitude, or cylindrical equidistant / `'lat-lon'`	`pole_lat`
	`pole_lon`
	`stand_lon`

In the illustrations of the Lambert conformal, polar stereographic, and Mercator projections, it may be seen that the so-called true latitude (or true latitudes, in the case of the Lambert conformal), is the latitude at which the surface of projection intersects or is tangent to the surface of the earth. At this latitude, there is no distortion in the distances in the map projection, while at other latitudes, the distance on the surface of the earth is related to the distance on the surface of projection by a *map scale factor*. Ideally, the map projection and its accompanying parameters should be chosen to minimize the maximum distortion within the area covered by the model grids, since a high amount of distortion, evidenced by map scale factors significantly different from unity, can restrict the model time step more than necessary. As a general guideline, the polar stereographic projection is best suited for high-latitude WRF domains, the Lambert conformal projection is well-suited for mid-latitude domains, and the Mercator projection is good for low-latitude domains or domains with predominantly west-east extent. The cylindrical equidistant projection is required for global ARW simulations, although in its rotated aspect (i.e., when `pole_lat`, `pole_lon`, and `stand_lon` are changed from their default values) it can also be well-suited for regional domains anywhere on the earth’s surface.


Polar Stereographic
True latitude
15° 30° 45° 60° 75° 90° 75° 60° 45° 30° 15°
135°W 150°W 165°W 180° 165°E 150°E 135°E
135°W 120°W 105°W 90°W 75°W 60°W 45°W
135°E 120°E 105°E 90°E 75°E 60°E 45°E
45°W 30°W 15°W 0° 15°E 30°E 45°E
Lambert Conformal
True latitude 1
True latitude 2
90° 75° 60° 45° 30° 15° 0°
180° 150°W 120°W 90°W
180° 150°E 120°E 90°E
60°W 30°W 0° 30°E 60°E
Mercator
True latitude
60°N 45°N 30°N 15°N 0° 15°S 30°S 45°S 60°S
60°N 30°N 0° 30°S 60°S
180° 135°W 90°W 45°W 0° 45°E 90°E 135°E 180°
Cylindrical Equidistant
90°N 75°N 60°N 45°N 30°N 15°N 0° 15°S 30°S 45°S 60°S 75°S 90°S
90°N 45°N 0° 45°S 90°S
180° 90°W 0° 90°E 180°

When configuring a rotated latitude-longitude grid, the namelist parameters `pole_lat`, `pole_lon`, and `stand_lon` are changed from their default values. The parameters `pole_lat` and `pole_lon` specify the latitude and longitude of the geographic north pole within the model's *computational grid*, and `stand_lon` gives the rotation about the earth's axis. In the context of the ARW, the computational grid refers to the regular latitude-longitude grid on which model computation is done, and on whose latitude circles Fourier filters are applied at high latitudes; users interested in the details of this filtering are referred to the WRF Version 4 Technical Note (https://opensky.ucar.edu/islandora/object/opensky:2898), and here, it suffices to note that the computational latitude-longitude grid is always represented with computational latitude lines running parallel to the x-axis of the model grid and computational longitude lines running parallel to the y-axis of the grid.

If the earth's geographic latitude-longitude grid coincides with the computational grid, a global ARW domain shows the earth's surface as it is normally visualized on a regular latitude-longitude grid. If instead the geographic grid does not coincide with the model computational grid, geographical meridians and parallels appear as complex curves. The difference is most easily illustrated by way of example. In top half of the figure below, the earth is shown with the geographical latitude-longitude grid coinciding with the computational latitude-longitude grid. In the bottom half, the geographic grid (not shown) has been rotated so that the geographic poles of the earth are no longer located at the poles of the computational grid.



When WRF is to be run for a regional domain configuration, the location of the coarse domain is determined using the `ref_lat` and `ref_lon` variables, which specify the latitude and longitude, respectively, of the center of the coarse domain. If nested domains are to be processed, their locations with respect to the parent domain are specified with the `i_parent_start` and `j_parent_start` variables; further details of setting up nested domains are provided in the section on nested domains. Next, the dimensions of the coarse domain are determined by the variables `dx` and `dy`, which specify the nominal grid distance in the x-direction and y-direction, and `e_we` and `e_sn`, which give the number of velocity points (i.e., *u*-staggered or *v*-staggered points) in the x- and y-directions; for the `'lambert'`, `'mercator'`, and `'polar'` projections, `dx` and `dy` are given in meters, and for the `'lat-lon'` projection, `dx` and `dy` are given in degrees. For nested domains, only the variables `e_we` and `e_sn` are used to determine the dimensions of the grid, and `dx` and `dy` should not be specified for nests, since their values are determined recursively based on the values of the `parent_grid_ratio` and `parent_id` variables, which specify the ratio of a nest's parent grid distance to the nest's grid distance and the grid number of the nest's parent, respectively.

If the regular latitude-longitude projection will be used for a regional domain, care must be taken to ensure that the map scale factors in the region covered by the domain do not deviate significantly from unity. This can be accomplished by rotating the projection such that the area covered by the domain is located near the equator of the projection, since, for the regular latitude-longitude projection, the map scale factors in the x-direction are given by the cosine of the computational latitude. For example, in the figure above showing the unrotated and rotated earth, it can be seen that, in the rotated aspect, New Zealand is located along the computational equator, and thus, the rotation used there would be suitable for a domain covering New Zealand. As a general guideline for rotating the latitude-longitude projection for regional domains, the namelist parameters `pole_lat`, `pole_lon`, and `stand_lon` may be chosen according to the formulas in the following table.

	(`ref_lat`, `ref_lon`) in N.H.	(`ref_lat`, `ref_lon`) in S.H.
`pole_lat`	90.0 – `ref_lat`	90.0 + `ref_lat`
`pole_lon`	180.0	0.0
`stand_lon`	`-ref_lon`	`180.0 - ref_lon`

For global WRF simulations, the coverage of the coarse domain is, of course, global, so `ref_lat` and `ref_lon` do not apply, and `dx` *and* `dy` *should not be specified*, since the nominal grid distance is computed automatically based on the number of grid points. Also, it should be noted that the latitude-longitude, or cylindrical equidistant, projection (`map_proj = 'lat-lon'`) is the only projection in WRF that can support a global domain. *Nested domains within a global domain must not cover any area north of computational latitude +45 or south of computational latitude -45, since polar filters are applied poleward of these latitudes (although the cutoff latitude can be changed in the WRF namelist).*

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.

Depending on the value of the `wrf_core` namelist variable, the appropriate GEOGRID.TBL file must be used with geogrid, since the grid staggerings that WPS interpolates to 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. 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.ARW
```

For more details on the meaning and possible values for each variable, 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.

```
> 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_em.d01.nc
-rw-r--r-- 1  4745324 geo_em.d02.nc
drwxr-xr-x 4     4096 geogrid
lrwxrwxrwx 1       23 geogrid.exe -> geogrid/src/geogrid.exe
-rw-r--r-- 1    11169 geogrid.log
-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
```

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 = 'ARW',
 max_dom = 2,
 start_date = '2008-03-24_12:00:00','2008-03-24_12:00:00',
 end_date   = '2008-03-24_18:00:00','2008-03-24_12:00:00',
 interval_seconds = 21600,
 io_form_geogrid = 2
/

&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_em.d01.nc
-rw-r--r-- 1  4745324 geo_em.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_em.d01.nc
-rw-r--r-- 1    4745324 geo_em.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 = 'ARW',
 max_dom = 2,
 start_date = '2008-03-24_12:00:00','2008-03-24_12:00:00',
 end_date   = '2008-03-24_18:00:00','2008-03-24_12:00:00',
 interval_seconds = 21600,
 io_form_geogrid = 2
/

&metgrid
 fg_name                         = 'FILE',
 io_form_metgrid                 = 2,
/
```

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).

```
> ls metgrid/METGRID.TBL

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

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.

```
> 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_em.d01.nc
-rw-r--r-- 1    4745324 geo_em.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
-rw-r--r-- 1    5217648 met_em.d01.2008-03-24_12:00:00.nc
-rw-r--r-- 1    5217648 met_em.d01.2008-03-24_18:00:00.nc
-rw-r--r-- 1   12658200 met_em.d02.2008-03-24_12:00:00.nc
drwxr-xr-x 3       4096 metgrid
lrwxrwxrwx 1         23 metgrid.exe -> metgrid/src/metgrid.exe
-rw-r--r-- 1      65970 metgrid.log
-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
```

Creating Nested Domains with the WPS

To run the WPS for nested-domain simulations is essentially no more difficult than running for a single-domain case; the difference with nested-domain simulations is that the geogrid and metgrid programs process more than one grid when they are run, rather than a single grid for the simulation. In order to specify the size and location of nests, a number of variables in the namelist.wps file must be given lists of values, one value per nest.

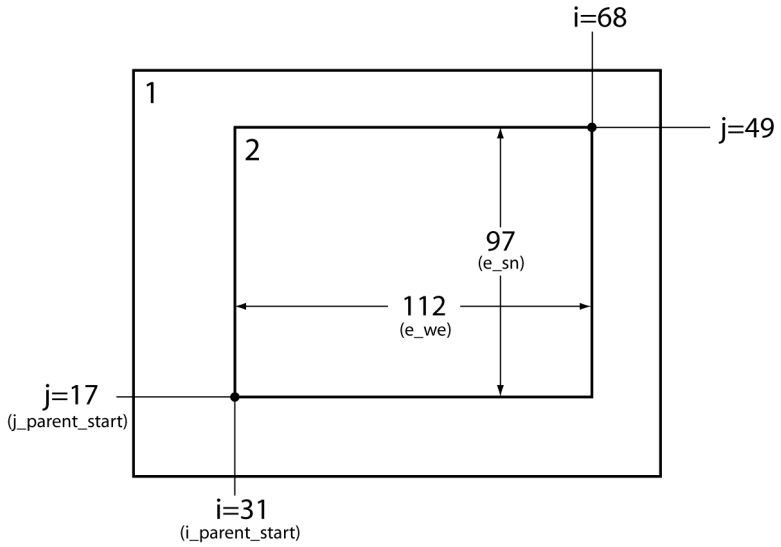
```
&share
 wrf_core = 'ARW',
 max_dom = 2,
 start_date = '2008-03-24_12:00:00','2008-03-24_12:00:00',
 end_date   = '2008-03-24_18:00:00','2008-03-24_12:00:00',
 interval_seconds = 21600,
 io_form_geogrid = 2
/

&geogrid
 parent_id         =   1,   1,
 parent_grid_ratio =   1,   3,
 i_parent_start    =   1,  31,
 j_parent_start    =   1,  17,
 e_we              =  74, 112,
 e_sn              =  61,  97,
 geog_data_res     = 'default','default',
 dx = 30000,
 dy = 30000,
 map_proj = 'lambert',
 ref_lat   = 34.83,
 ref_lon   = -81.03,
 truelat1  =  30.0,
 truelat2  =  60.0,
 stand_lon = -98.
 geog_data_path = '/mmm/users/wrfhelp/WPS_GEOG/'
/
```

The namelist variables that are affected by nests are shown in the (partial) namelist records above. The example shows namelist variables for a two-domain run (the coarse domain plus a single nest), and the effect on the namelist variables generalize to multiple nests in the obvious way: rather than specifying lists of two values, lists of *N* values must be specified, where *N* is the total number of model grids.

In the above example, 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 coarse 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 grid. 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, except when grid nudging, also called analysis nudging, is used in WRF. 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 its 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.



Next, the lower-left corner of a nest is specified as an (*i*, *j*) location in the nest's parent domain; this is done through the `i_parent_start` and `j_parent_start` variables, and the specified location is given with respect to the unstaggered 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. The nesting setup in our example namelist is illustrated in the figure above, where it may be seen how each of the above-mentioned variables is determined. 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; to ensure that the upper-

right corner of the nest's grid is coincident with an unstaggered grid point in the parent domain, both `e_we` and `e_sn` must be one greater than some integer multiple of the nesting ratio. Also, for each nest, the resolution (or list or resolutions; see the description of namelist variables) 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 MODIS IGBP 21-category data. However, the user may select an alternative set of land use categories based on the USGS land-cover classification. Although the MODIS-based data contain 21 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.

The 24-category USGS-based land use data may be selected instead of the MODIS 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 "`usgs_lakes+`". For example, in a two-domain configuration, where the `geog_data_res` variable would ordinarily be specified as

```
geog_data_res = 'default', 'default',
```

the user should instead specify

```
geog_data_res = 'usgs_lakes+default', 'usgs_lakes+default',
```

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 'usgs_lakes', 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 USGS-based land use data, which is identified with the string 'usgs_lakes', would be used instead of the 'default', resolutions (or source) of land-use data in the example above; for all other fields, the 'default' resolutions would be used for the first and second. As an aside, when none of the resolutions specified for a domain in `geog_data_res` is found in a GEOGRID.TBL entry, the resolution denoted by 'default' will be used.

When changing from the default 21-class MODIS land-use data, the user must also ensure that the `num_land_cat` namelist variable is set correctly in `&physics` namelist record in the WRF namelist.input file. For 24-class USGS data, `num_land_cat` should be set to 24.

Selecting Static Data for the Gravity Wave Drag Scheme

The gravity wave drag by orography (GWDO) scheme in the ARW requires ten 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 = 'default',
```

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+default',
```

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-minute 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. 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.exe. 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:

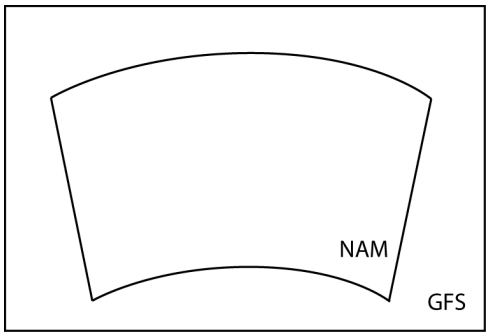
```
&metgrid
 constants_name = 'NARR_FIXED',
 fg_name = 'NARR_3D', 'NARR_SFC'
/
```

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.

Using Non-isobaric Meteorological Datasets

When using non-isobaric meteorological datasets to initialize a WRF simulation, it is important that such datasets are supplied to the metgrid.exe program with 3-d pressure and geopotential height fields on the same levels as other 3-d atmospheric variables, such as temperature and humidity. These fields are used by the WRF real.exe pre-processor for

vertical interpolation to WRF model levels, for surface pressure computation, and for other purposes.

For some data sources, namely ECMWF model-level data and UK Met Office model data, the 3-d pressure and/or geopotential height fields can be derived from the surface pressure and/or surface height fields using an array of coefficients, and the WPS provides utility programs for performing this derivation; see the section on WPS Utility Programs for more information on the calc_ecmwf_p.exe and height_ukmo.exe programs.

Other meteorological datasets explicitly provide 3-d pressure and geopotential height fields, and the user must only ensure that these fields exist in the set of intermediate files provided to the metgrid.exe program.

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.

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

1. Before running geogrid, ensure that the specification of `geog_data_res` in the `&geogrid` namelist record indicates either the USGS-based or the MODIS-based land use data with inland water bodies. The default behavior in the geogrid program is to use MODIS-based land use that contains a lake category. However, a USGS land use dataset that contains a lake category may also be used. For example, in a two-domain configuration, setting

   ```
   geog_data_res = 'usgs_lakes+default', 'usgs_lakes+default',
   ```

   would tell geogrid to use the USGS-based land use data for both domains.

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.

2. 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.

3. 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.

4. 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.

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 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, ncview, and new RIP4 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. Also, for users wishing to produce plots of fields suitable for use in publications, the new release of the RIP4 program may be of interest. The new RIP4 is capable of plotting horizontal contours, map backgrounds, and overlaying multiple fields within the same plot.

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 a 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.exe, 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.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 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 one of the following links:

http://www.ecmwf.int/en/forecasts/documentation-and-support/16-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/19-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/31-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/40-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/50-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/60-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/62-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/91-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/137-model-levels

This table should be written in plain text to a file, 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. height_ukmo.exe

The real program requires 3-d pressure and geopotential height fields to vertically interpolate the output of the metgrid program; however, data sets from the UKMO Unified Model contain a 3-d pressure field, but do not contain a geopotential height field. Accordingly, the height_ukmo.exe program may be used to compute a geopotential height field for data sets from the UKMO Unified Model. The height_ukmo.exe program requires no command-line arguments, but reads the `&metgrid` namelist record to get the prefix of the intermediate files created by ungrib.exe; the intermediate files indicated by the first prefix in the `fg_name` variable of the `&metgrid` namelist record are expected to contain a SOILHGT field, from which the height_ukmo.exe program computes, with the aid of an auxiliary table, the 3-d geopotential height field. The computed height field is written to a new intermediate file with the prefix HGT, and the prefix ‘HGT’ should then be added to the `fg_name` namelist variable in the `&metgrid` namelist record before running metgrid.exe. The name of the file containing the auxiliary table is currently hard-wired in the source code of the height_ukmo.exe program, and it is the responsibility of the user to change this file name in WPS/util/src/height_ukmo.F to the name of the table with the same number of levels as the GRIB data processed by ungrib.exe; tables for data with 38, 50, and 70 levels are provided in the WPS/util directory with file names vertical_grid_38_20m_G3.txt, vertical_grid_50_20m_63km.txt , and vertical_grid_70_20m_80km.txt, respectively.

E. plotgrids.ncl

The plotgrids.ncl 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 Graphics file in the chosen format (see inside the plotgrids.ncl script for making changes to the output format), 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.ncl, and determining a set of adjustments to the nest locations. To run this program, simply type 'ncl util/plotgrids.ncl' in the command line from inside the WPS/ directory. *Currently, this utility does not work for ARW domains that use the latitude-longitude projection* (i.e., when `map_proj = 'lat-lon'`).

F. 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.

G. 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.

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.F and metgrid/src/write_met_module.F, 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
```

Required Meteorological Fields for Running WRF

In order to successfully initialize a WRF simulation, the real.exe pre-processor requires a minimum set of meteorological and land-surface fields to be present in the output from the metgrid.exe program. Accordingly, these required fields must be available in the intermediate files processed by metgrid.exe. The set of required fields is described in the table, below.

Field name in intermediate file	Units	Description	Notes
TT	K	3-d air temperature	
RH	%	3-d relative humidity	Not needed if SPECHUMD is available
SPECHUMD	kg kg^{-1}	3-d specific humidity	Not needed if RH is available
UU	m s^{-1}	3-d wind u-component	
VV	m s^{-1}	3-d wind v-component	
GHT	m	3-d geopotential height	
PRESSURE	Pa	3-d pressure	Only needed for non-isobaric datasets
PSFC	Pa	Surface pressure	

PMSL	Pa	Mean sea-level pressure	
SKINTEMP	K	Skin temperature	
SOILHGT	m	Soil height	
TT	K	2-meter air temperature	
RH	%	2-meter relative humidity	Not needed if SPECHUMD is available
SPECHUMD	kg kg^{-1}	2-meter specific humidity	Not needed if RH is available
UU	m s^{-1}	10-meter wind u-component	
VV	m s^{-1}	10-meter wind v-component	
LANDSEA	fraction	Land-sea mask (0=water, 1=land)	
SM*tttbbb*	m^3 m^{-3}	Soil moisture	'*ttt*' is the layer top depth in cm, and '*bbb*' is the layer bottom depth in cm
ST*tttbbb*	K	Soil temperature	
SOILM*mmm*	kg m^{-3}	Soil moisture	'*mmm*' is the level depth in cm, not needed if SMtttbbb available
SOILT*mmm*	K	Soil temperature	

Using MPAS Output for WRF Initial and Lateral Boundary Conditions

The metgrid.exe program is capable of reading native, unstructured mesh output in netCDF format from the Model for Prediction Across Scales (MPAS; https://mpas-dev.github.io/); the metgrid.exe program can then horizontally interpolate the MPAS fields directly to any domain defined by the geogrid.exe program to produce output files that are usable by the WRF real.exe program in exatly the same way as metgrid output interpolated from intermediate files. In this way, output from MPAS may be used to provide initial and lateral boundary conditions for WRF.

When running an MPAS simulation, an output stream must be set up to contain the minimum set of fields necessary to initialize a WRF simulation. The following output stream should be sufficient with the MPAS v5.x and later code.

```
<stream name="wrf_ic_bc"
        type="output"
        filename_template="MPAS.$Y-$M-$D_$h.nc"
        output_interval="3:00:00" >

 <var name="xtime"/>
 <var_array name="scalars"/>
 <var name="pressure"/>
 <var name="zgrid"/>
 <var name="theta"/>
 <var name="uReconstructZonal"/>
 <var name="uReconstructMeridional"/>
 <var name="u10"/>
```

```
 <var name="v10"/>
 <var name="q2"/>
 <var name="t2m"/>
 <var name="skintemp"/>
 <var name="surface_pressure"/>
 <var name="mslp"/>
 <var name="tslb"/>
 <var name="smois"/>

</stream>
```

After having run MPAS with a suitable output stream defined, a set of netCDF files containing fields on the native MPAS mesh will have been produced. Because these files do not contain fields describing the locations, geometry, and connectivity of the MPAS grid cells, this information must be provided to the metgrid program with a "static" file from the MPAS simulation. Therefore, it is necessary to specify MPAS netCDF files (prefixed with `'mpas:'`) in the `&metgrid` namelist record with both the `constants_name` and `fg_name` variables, e.g.,

```
&metgrid
 constants_name = 'mpas:static.nc'
 fg_name = 'mpas:MPAS'
/
```

In the above example, the metgrid.exe program would first read the MPAS 'static.nc' file to read mesh information and compute remapping weights from the MPAS mesh to the WRF domain defined by the geogrid.exe program, then all time periods of the MPAS files with a prefix of 'MPAS' (and a suffix of `YYYY-MM-DD_HH.nc`) would be processed. The real.exe program can then be run as usual.

Data from intermediate files created by the ungrib.exe program can be combined with MPAS data by the metgrid program. This may be useful, e.g., to use SST, sea ice, or land-surface fields from another source. An example of combining MPAS data with ERA-Interim intermediate files with soil data (with the prefix 'ERAI_SOIL') is shown below.

```
&metgrid
  constants_name = 'mpas:static.nc'
  fg_name = 'mpas:MPAS', 'ERAI_SOIL'
/
```

Because the MPAS 'zgrid' field does not change in time, it can be omitted from the MPAS periodic output stream; in this case, however, the 'zgrid' field must be placed in its own netCDF file that must also define the dimension 'Time' as a netCDF unlimited dimension. Then, this file (say, 'zgrid.nc') can be supplied to the metgrid program using the `constants_name` namelist variable, e.g.,

```
&metgrid
 constants_name = 'mpas:static.nc', 'mpas:zgrid.nc'
 fg_name = 'mpas:MPAS'
```

```
/
```

Placing the ‘zgrid’ field in its own file can save considerable space when long MPAS simulations are run, or when the output stream to be used as WRF initial and boundary conditions is written out at high temporal frequency. The python script, below, may serve as an example of how to extract the ‘zgrid’ field to its own netCDF file.

```
from netCDF4 import Dataset

fin = Dataset('init.nc')
fout = Dataset('zgrid.nc','w',format='NETCDF3_64BIT')

nCells = fin.dimensions['nCells'].size
nVertLevelsP1 = fin.dimensions['nVertLevelsP1'].size

fout.createDimension(dimname='Time',size=None)
fout.createDimension(dimname='nCells',size=nCells)
fout.createDimension(dimname='nVertLevelsP1',size=nVertLevelsP1)
fout.createVariable(varname='zgrid',datatype='f',dimensions=('nCells',
'nVertLevelsP1'))
fout.variables['zgrid'][:] = fin.variables['zgrid'][:]
fout.close()
fin.close()
```

It is worth noting that the use of native MPAS output with metgrid.exe has not been thoroughly tested for parallel (i.e., “dmpar”) builds of the WPS; it is therefore recommended to run metgrid.exe in serial when processing MPAS datasets.

Also, in cases of large MPAS meshes, it may be necessary to increase the value of two constants in the metgrid code that are used to statically allocate several datastructures used in the computation of remapping weights from the MPAS mesh to the WRF domain. These two constants, shown below, are located in the WPS/src/metgrid/remapper.F file.

```
! should be at least (earth circumference / minimum grid distance)
integer, parameter :: max_queue_length    = 2700

! should be at least (nCells/32)
integer, parameter :: max_dictionary_size = 82000
```

After changing the value of these constants, metgrid must be recompiled.

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 "From Level2" fields are used to specify which levels a field may be found at. As with the "GRIB1 Param" field, the g1print.exe and g2print.exe programs may be used to find values for the level fields. The meanings of the level fields are dependent on the "Level Type" field and are summarized in the following table.

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*.

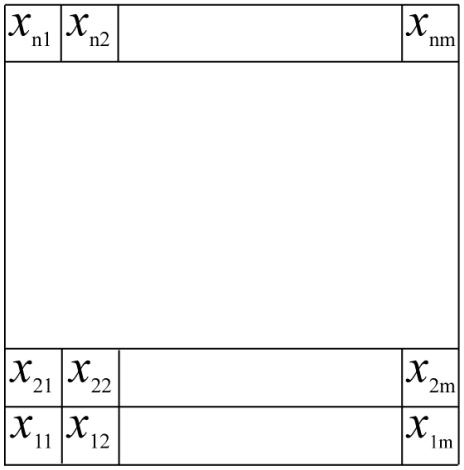
The final group of fields, which provide GRIB2-specific information, are found under the column headings below.

```
|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.

Creating an Urban Fraction Field from NLCD Data

Note: An urban fraction field that has been prepared based on 30-meter NLCD 2011 land cover is available for the continental United States; this dataset is available from the WPS geographical download page under the section for "optional fields". The details below may still be helpful for preparation of urban fraction fields for other regions.

In order to create a more inhomogeneous and detailed urban fraction field for use with NUDAPT, users may obtain high-resolution land cover information from the National Land Cover Database (NLCD) through the Multi-Resolution Land Characteristics Consortium. Generation of the urban fraction field, called FRC_URB2D in WRF, involves first downloading the NLCD data over the region covered by the WRF domain, converting the data into the binary format used by geogrid, and finally extracting only the urban categories to a new urban fraction field. The following steps can serve as a guide through this process.

1. Download NLCD data from http://gisdata.usgs.net/website/MRLC/viewer.php. Either of the 1992, 2001, or 2006 datasets may be used. After selecting an area to download, make sure to select GeoTIFF format in the "Request Summary Page" by clicking on "Modify Data Request". If available, data may instead be downloaded in BIL format, in which case the format conversion described in the next step can be skipped.

2. After downloading the data, unpacking the archive should yield a directory with a .tif file and a .tfw file, among others. In order for the information in the GeoTIFF file to be useful, the .tif image must be converted into the binary format used by the WPS. This conversion can be accomplished using the GDAL translation tool, gdal_translate, (http://gdal.org) by running the command

   ```
   > gdal_translate -of ENVI foo.tif data.bil
   ```

   where foo.tif is the name of the GeoTIFF image that was downloaded in Step 1. The output format "ENVI" is a simple binary raster format that matches the format used by geogrid. After converting the GeoTIFF to a binary file, the resulting data.bil file must be renamed to 00001-ncols.00001-nrows, where ncols is the number of columns (in i5.5 format) and nrows is the number of rows (also in i5.5 format) in the image; *these values should have been printed to the screen when the gdal_translate program was run.*

3. Use the converter program available from http://www2.mmm.ucar.edu/people/duda/uf/ to extract the urban categories from the binary tile and write a new tile of data containing urban fraction. The output file of this converter should be copied over the original land use tile, i.e., the urban fraction file should be renamed to 00001-ncols.00001-nrows, where ncols is the number of columns (in i5.5 format) and nrows is the number of rows (also in i5.5 format) in the tile, as in Step 2.

4. Create an index metadata file for the urban fraction data. In the directory created by unpacking the land use data, a .tfw file should also exist. The last two lines in this file give the location of the north-west corner of the data tile, which is used in the index file for variables known_lat and known_lon. If this location is given as (x,y) coordinates, in meters, then the coordinate converter utility available from http://www2.mmm.ucar.edu/people/duda/uf/ may be used to convert to (latitude, longitude), which is required by the index file. The basic index file should contain the following elements:

   ```
   type=continuous
   projection=albers_nad83
   dx=30.0
   dy=30.0
   known_x=1.0
   known_y=2351.0             # <- edit
   known_lat =   40.096571    # <- edit
   known_lon = -105.405615    # <- edit
   ```

```
truelat1=29.5
truelat2=45.5
stdlon=-96.0
wordsize=1
scale_factor=0.01
row_order=top_bottom
tile_x=2407               # <- edit
tile_y=2351               # <- edit
tile_z=1
units="unitless"
description="urban fraction"
```

5. Add the following entry to the GEOGRID.TBL file before re-running the geogrid.exe program:

```
==============================
name=FRC_URB2D
        priority=1
        dest_type=continuous
        fill_missing = 0.
        interp_option=default: average_gcell(1.0)+four_pt
        abs_path=default:/path/to/dataset/
==============================
```

   The path to the dataset and index files created in Step 3 and Step 4, respectively, should be substituted for '/path/to/dataset/' in the entry above.

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. 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.

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.

5. S_WE : A list of MAX_DOM integers which should all be set to 1. Default value is 1.

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_we` = *n**`parent_grid_ratio`+1 for some positive integer *n*). No default value.

7. S_SN : A list of MAX_DOM integers which should all be set to 1. Default value is 1.

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.

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.

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'.

3. ADD_LVLS : A logical that determines whether ungrib will attemp to vertically interpolate to an additional set of vertical levels specified using the NEW_PLVL and INTERP_TYPE namelist options. Default value is .FALSE..

4. INTERP_TYPE : An integer value specifying the method that ungrib will use when vertically interpolating to new levels. A value of 0 causes ungrib to interpolate linearly in pressure, and a value of 1 causes ungrib to interpolate linearly in log pressure. Default value is 0.

5. NEW_PLVL : An array of real values that specify the additional vertical levels, given in Pa, to which the ungrib program will attempt to interpolate when ADD_LVLS is true. The set of new levels can be specified explicitly, or, if the levels are evenly spaced in pressure, exactly three values can be specified: the starting pressure, the ending pressure, and the pressure increment. When a starting pressure, ending pressure, and increment are specified, the pressure increment must be a negative number to signal to the ungrib program that this value is not a target pressure level, but rather, an increment to be used between the first and second values. No default value.

6. PMIN : A real value specifying the minimum pressure level, in Pa, to be processed from GRIB data. This option applies only to isobaric data sets. Default value is 100.

D. METGRID section

This section defines variables used only by the metgrid program. Typically, the user will be interested primarily 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/'`.

5. PROCESS_ONLY_BDY: An integer specifying the number of boundary rows and columns to be processed by metgrid for time periods after the initial time; for the initial time, metgrid will always interpolate to every grid point. Setting this option to the intended value of spec_bdy_width in the WRF namelist.input will speed up processing in metgrid, but it should not be set if interpolated data are needed in the domain interior. If this option is set to zero, metgrid will horizontally interpolate meteorological data to every grid point in the model domains. *This option is only available for ARW.* Default value is 0.

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(`*r*`)`, and `average_gcell(`*r*`)`. For the search method (`search`), the optional argument *r* specifies the maximum search radius in units of grid points in the grid of the source data; the default search radius is 1200 points. 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; unless 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.

20. 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 geogrid output. Default value is null (i.e., no flag will be written for the field).

21. OPTIONAL : Either `yes` or `no`, indicating whether the dataset identified by the resolution specified in the `geog_data_res` namelist option is optional. If an entry in the GEOGRID.TBL file is optional and if the specified resolution of data cannot be read, geogrid will print an informational message indicating that the dataset was not interpolated and continue; otherwise, if the entry is not optional and the specified resolution of data cannot be read, geogrid will halt with an error. It is possible for different priority level entries for the same field to specify different values of the `optional` keyword, e.g., the `priority=2` entry for a field can be optional, while the `priority=1` entry can be non-optional (i.e., `optional=no`). Default value is `no`.

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"`.

34. FILENAME_DIGITS : An integer specifying the number of digits used in the names of data tiles. Possible values are 5 or 6. Default value is 5.

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 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(`*r*`)`, and `average_gcell(`*r*`)`. For the search method (`search`), the optional argument *r* specifies the maximum search radius in units of grid points in the grid of the source data; the default search radius is 1200 points. 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; unless 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.

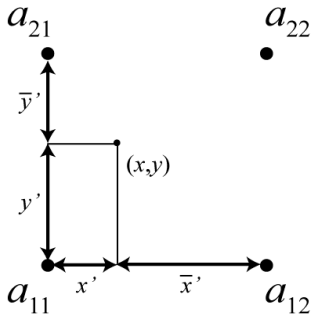
21. FLAG_IN_OUTPUT : A character string giving the name of a global attribute which will be assigned a value of 1 and written to the metgrid output if the interpolated field is to be output (output=yes). Default value is null (i.e., no flag will be written for the field).

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

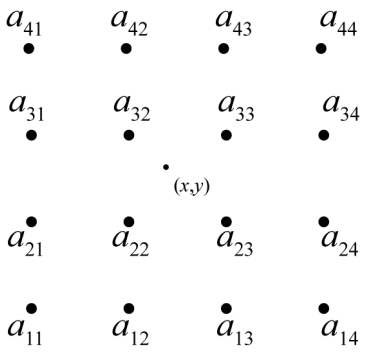
each method, the user is referred to the source code in the file WPS/geogrid/src/interp_options.F.

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 $1 \le i, j \le 4$.

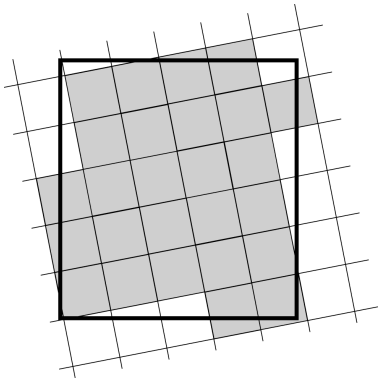
7. nearest_neighbor : Nearest neighbor interpolation

When used for continuous datasets (i.e., datasets that have `type=continuous` in their index files), 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. For categorical datasets (i.e., datasets that have `type=categorical` in their index files), this option actually causes the geogrid program to consider all source pixels that lie within each WRF grid cell, and to find the fraction of the WRF grid cell that is comprised of each category in the source data.

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)

Table 4: NLCD 40-category Landuse 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	Permanent Snow and Ice
16	Barren or Sparsely Vegetated
17	IGBP Water
18	Unclassified
19	Fill Value
20	Unclassified
21	Open Water
22	Perennial Ice/Snow
23	Developed Open Space
24	Developed Low Intensity
25	Developed Medium Intensity
26	Developed High Intensity
27	Barren Land (Rock/Sand/Clay)
28	Deciduous Forest
29	Evergreen Forest
30	Mixed Forest
31	Dwarf Scrub
32	Shrub/Scrub
33	Grassland/Herbaceous
34	Sedge/Herbaceous
35	Lichens
36	Moss
37	Pasture/Hay
38	Cultivated Crops
39	Woody Wetlands
40	Emergent Herbaceous Wetlands

WPS Output Fields

Below, a listing of the global attributes and fields that are written to the geogrid program's output files is given. This listing is an abridged version of the output from the ncdump program when run on a typical geo_em.d01.nc file.

```
netcdf geo_em.d01 {
dimensions:
      Time = UNLIMITED ; // (1 currently)
      DateStrLen = 19 ;
      west_east = 73 ;
      south_north = 60 ;
      south_north_stag = 61 ;
```

```
     west_east_stag = 74 ;
     land_cat = 21 ;
     soil_cat = 16 ;
     month = 12 ;
     num_urb_params = 132 ;
variables:
     char Times(Time, DateStrLen) ;
     float XLAT_M(Time, south_north, west_east) ;
          XLAT_M:units = "degrees latitude" ;
          XLAT_M:description = "Latitude on mass grid" ;
     float XLONG_M(Time, south_north, west_east) ;
          XLONG_M:units = "degrees longitude" ;
          XLONG_M:description = "Longitude on mass grid" ;
     float XLAT_V(Time, south_north_stag, west_east) ;
          XLAT_V:units = "degrees latitude" ;
          XLAT_V:description = "Latitude on V grid" ;
     float XLONG_V(Time, south_north_stag, west_east) ;
          XLONG_V:units = "degrees longitude" ;
          XLONG_V:description = "Longitude on V grid" ;
     float XLAT_U(Time, south_north, west_east_stag) ;
          XLAT_U:units = "degrees latitude" ;
          XLAT_U:description = "Latitude on U grid" ;
     float XLONG_U(Time, south_north, west_east_stag) ;
          XLONG_U:units = "degrees longitude" ;
          XLONG_U:description = "Longitude on U grid" ;
     float CLAT(Time, south_north, west_east) ;
          CLAT:units = "degrees latitude" ;
          CLAT:description = "Computational latitude on mass grid" ;
     float CLONG(Time, south_north, west_east) ;
          CLONG:units = "degrees longitude" ;
          CLONG:description = "Computational longitude on mass grid" ;
     float MAPFAC_M(Time, south_north, west_east) ;
          MAPFAC_M:units = "none" ;
          MAPFAC_M:description = "Mapfactor on mass grid" ;
     float MAPFAC_V(Time, south_north_stag, west_east) ;
          MAPFAC_V:units = "none" ;
          MAPFAC_V:description = "Mapfactor on V grid" ;
     float MAPFAC_U(Time, south_north, west_east_stag) ;
          MAPFAC_U:units = "none" ;
          MAPFAC_U:description = "Mapfactor on U grid" ;
     float MAPFAC_MX(Time, south_north, west_east) ;
          MAPFAC_MX:units = "none" ;
          MAPFAC_MX:description = "Mapfactor (x-dir) on mass grid" ;
     float MAPFAC_VX(Time, south_north_stag, west_east) ;
          MAPFAC_VX:units = "none" ;
          MAPFAC_VX:description = "Mapfactor (x-dir) on V grid" ;
     float MAPFAC_UX(Time, south_north, west_east_stag) ;
          MAPFAC_UX:units = "none" ;
          MAPFAC_UX:description = "Mapfactor (x-dir) on U grid" ;
     float MAPFAC_MY(Time, south_north, west_east) ;
          MAPFAC_MY:units = "none" ;
          MAPFAC_MY:description = "Mapfactor (y-dir) on mass grid" ;
     float MAPFAC_VY(Time, south_north_stag, west_east) ;
          MAPFAC_VY:units = "none" ;
          MAPFAC_VY:description = "Mapfactor (y-dir) on V grid" ;
     float MAPFAC_UY(Time, south_north, west_east_stag) ;
          MAPFAC_UY:units = "none" ;
          MAPFAC_UY:description = "Mapfactor (y-dir) on U grid" ;
     float E(Time, south_north, west_east) ;
          E:units = "-" ;
          E:description = "Coriolis E parameter" ;
     float F(Time, south_north, west_east) ;
          F:units = "-" ;
```

```
          F:description = "Coriolis F parameter" ;
     float SINALPHA(Time, south_north, west_east) ;
          SINALPHA:units = "none" ;
          SINALPHA:description = "Sine of rotation angle" ;
     float COSALPHA(Time, south_north, west_east) ;
          COSALPHA:units = "none" ;
          COSALPHA:description = "Cosine of rotation angle" ;
     float LANDMASK(Time, south_north, west_east) ;
          LANDMASK:units = "none" ;
          LANDMASK:description = "Landmask : 1=land, 0=water" ;
     float XLAT_C(Time, south_north_stag, west_east_stag) ;
          XLAT_C:units = "degrees latitude" ;
          XLAT_C:description = "Latitude at grid cell corners" ;
     float XLONG_C(Time, south_north_stag, west_east_stag) ;
          XLONG_C:units = "degrees longitude" ;
          XLONG_C:description = "Longitude at grid cell corners" ;
     float LANDUSEF(Time, land_cat, south_north, west_east) ;
          LANDUSEF:units = "category" ;
          LANDUSEF:description = "Noah-modified 21-category IGBP-MODIS landuse"
;
     float LU_INDEX(Time, south_north, west_east) ;
          LU_INDEX:units = "category" ;
          LU_INDEX:description = "Dominant category" ;
     float HGT_M(Time, south_north, west_east) ;
          HGT_M:units = "meters MSL" ;
          HGT_M:description = "GMTED2010 30-arc-second topography height" ;
     float SOILTEMP(Time, south_north, west_east) ;
          SOILTEMP:units = "Kelvin" ;
          SOILTEMP:description = "Annual mean deep soil temperature" ;
     float SOILCTOP(Time, soil_cat, south_north, west_east) ;
          SOILCTOP:units = "category" ;
          SOILCTOP:description = "16-category top-layer soil type" ;
     float SCT_DOM(Time, south_north, west_east) ;
          SCT_DOM:units = "category" ;
          SCT_DOM:description = "Dominant category" ;
     float SOILCBOT(Time, soil_cat, south_north, west_east) ;
          SOILCBOT:units = "category" ;
          SOILCBOT:description = "16-category top-layer soil type" ;
     float SCB_DOM(Time, south_north, west_east) ;
          SCB_DOM:units = "category" ;
          SCB_DOM:description = "Dominant category" ;
     float ALBEDO12M(Time, month, south_north, west_east) ;
          ALBEDO12M:units = "percent" ;
          ALBEDO12M:description = "Monthly surface albedo" ;
     float GREENFRAC(Time, month, south_north, west_east) ;
          GREENFRAC:units = "fraction" ;
          GREENFRAC:description = "MODIS FPAR" ;
     float LAI12M(Time, month, south_north, west_east) ;
          LAI12M:units = "m^2/m^2" ;
          LAI12M:description = "MODIS LAI" ;
     float SNOALB(Time, south_north, west_east) ;
          SNOALB:units = "percent" ;
          SNOALB:description = "Maximum snow albedo" ;
     float SLOPECAT(Time, south_north, west_east) ;
          SLOPECAT:units = "category" ;
          SLOPECAT:description = "Dominant category" ;
     float CON(Time, south_north, west_east) ;
          CON:units = "" ;
          CON:description = "Subgrid-scale orographic convexity" ;
     float VAR(Time, south_north, west_east) ;
          VAR:units = "" ;
          VAR:description = "Subgrid-scale orographic variance" ;
     float OA1(Time, south_north, west_east) ;
```

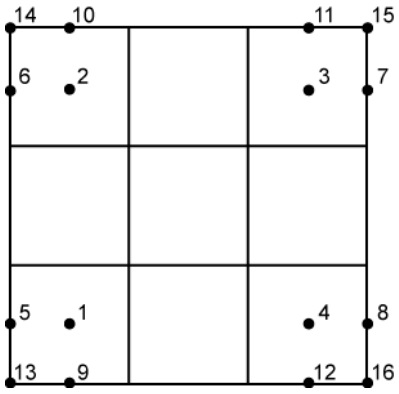
```
          OA1:units = "" ;
          OA1:description = "Subgrid-scale orographic asymmetry" ;
     float OA2(Time, south_north, west_east) ;
          OA2:units = "" ;
          OA2:description = "Subgrid-scale orographic asymmetry" ;
     float OA3(Time, south_north, west_east) ;
          OA3:units = "" ;
          OA3:description = "Subgrid-scale orographic asymmetry" ;
     float OA4(Time, south_north, west_east) ;
          OA4:units = "" ;
          OA4:description = "Subgrid-scale orographic asymmetry" ;
     float OL1(Time, south_north, west_east) ;
          OL1:units = "" ;
          OL1:description = "Subgrid-scale effective orographic length scale" ;
     float OL2(Time, south_north, west_east) ;
          OL2:units = "" ;
          OL2:description = "Subgrid-scale effective orographic length scale" ;
     float OL3(Time, south_north, west_east) ;
          OL3:units = "" ;
          OL3:description = "Subgrid-scale effective orographic length scale" ;
     float OL4(Time, south_north, west_east) ;
          OL4:units = "" ;
          OL4:description = "Subgrid-scale effective orographic length scale" ;
     float VAR_SSO(Time, south_north, west_east) ;
          VAR_SSO:units = "meters2 MSL" ;
          VAR_SSO:description = "Variance of Subgrid Scale Orography" ;
     float LAKE_DEPTH(Time, south_north, west_east) ;
          LAKE_DEPTH:units = "meters MSL" ;
          LAKE_DEPTH:description = "Topography height" ;
     float URB_PARAM(Time, num_urb_params, south_north, west_east) ;
          URB_PARAM:units = "dimensionless" ;
          URB_PARAM:description = "Urban_Parameters" ;

// global attributes:
          :TITLE = "OUTPUT FROM GEOGRID V4.0" ;
          :SIMULATION_START_DATE = "0000-00-00_00:00:00" ;
          :WEST-EAST_GRID_DIMENSION = 74 ;
          :SOUTH-NORTH_GRID_DIMENSION = 61 ;
          :BOTTOM-TOP_GRID_DIMENSION = 0 ;
          :WEST-EAST_PATCH_START_UNSTAG = 1 ;
          :WEST-EAST_PATCH_END_UNSTAG = 73 ;
          :WEST-EAST_PATCH_START_STAG = 1 ;
          :WEST-EAST_PATCH_END_STAG = 74 ;
          :SOUTH-NORTH_PATCH_START_UNSTAG = 1 ;
          :SOUTH-NORTH_PATCH_END_UNSTAG = 60 ;
          :SOUTH-NORTH_PATCH_START_STAG = 1 ;
          :SOUTH-NORTH_PATCH_END_STAG = 61 ;
          :GRIDTYPE = "C" ;
          :DX = 30000.f ;
          :DY = 30000.f ;
          :DYN_OPT = 2 ;
          :CEN_LAT = 34.83001f ;
          :CEN_LON = -81.03f ;
          :TRUELAT1 = 30.f ;
          :TRUELAT2 = 60.f ;
          :MOAD_CEN_LAT = 34.83001f ;
          :STAND_LON = -98.f ;
          :POLE_LAT = 90.f ;
          :POLE_LON = 0.f ;
          :corner_lats = 28.17127f, 44.36657f, 39.63231f, 24.61906f, 28.17842f,
44.37617f, 39.57812f, 24.57806f, 28.03771f, 44.50592f, 39.76032f, 24.49431f,
28.04485f, 44.51553f, 39.70599f, 24.45341f ;
          :corner_lons = -93.64893f, -92.39661f, -66.00165f, -72.64047f, -
```

```
93.80048f, -92.59155f, -65.83557f, -72.5033f, -93.65717f, -92.3829f, -65.9313f,
-72.68539f, -93.80841f, -92.57831f, -65.76495f, -72.54843f ;
           :MAP_PROJ = 1 ;
           :MMINLU = "MODIFIED_IGBP_MODIS_NOAH" ;
           :NUM_LAND_CAT = 21 ;
           :ISWATER = 17 ;
           :ISLAKE = 21 ;
           :ISICE = 15 ;
           :ISURBAN = 13 ;
           :ISOILWATER = 14 ;
           :grid_id = 1 ;
           :parent_id = 1 ;
           :i_parent_start = 1 ;
           :j_parent_start = 1 ;
           :i_parent_end = 74 ;
           :j_parent_end = 61 ;
           :parent_grid_ratio = 1 ;
           :FLAG_MF_XY = 1 ;
           :FLAG_LAI12M = 1 ;
           :FLAG_LAKE_DEPTH = 1 ;
}
```

The global attributes `corner_lats` and `corner_lons` contain the lat-lon location of the corners of the domain with respect to different grid staggerings (mass, *u*, *v*, and unstaggered). The locations referred to by each element of the `corner_lats` and `corner_lons` arrays are summarized in the table and figure below.

Array index	**Staggering**	**Corner**
1	Mass	Lower-left
2		Upper-left
3		Upper-right
4		Lower-right
5	U	Lower-left
6		Upper-left
7		Upper-right
8		Lower-right
9	V	Lower-left
10		Upper-left
11		Upper-right
12		Lower-right
13	Unstaggered	Lower-left
14		Upper-left
15		Upper-right
16		Lower-right



In addition to the fields in a geogrid output file (e.g., geo_em.d01.nc), the following fields and global attributes will also be present in a typical output file from the metgrid program, run with the default METGRID.TBL file and meteorological data from NCEP's GFS model.

```
netcdf met_em.d01.2016-04-07_00\:00\:00 {
dimensions:
      Time = UNLIMITED ; // (1 currently)
      DateStrLen = 19 ;
      west_east = 73 ;
      south_north = 60 ;
      num_metgrid_levels = 27 ;
      num_st_layers = 4 ;
      num_sm_layers = 4 ;
      south_north_stag = 61 ;
      west_east_stag = 74 ;
      z-dimension0132 = 132 ;
      z-dimension0012 = 12 ;
      z-dimension0016 = 16 ;
      z-dimension0021 = 21 ;
variables:
      char Times(Time, DateStrLen) ;
      float PRES(Time, num_metgrid_levels, south_north, west_east) ;
            PRES:units = "" ;
            PRES:description = "" ;
      float SOIL_LAYERS(Time, num_st_layers, south_north, west_east) ;
            SOIL_LAYERS:units = "" ;
            SOIL_LAYERS:description = "" ;
      float SM(Time, num_sm_layers, south_north, west_east) ;
            SM:units = "" ;
            SM:description = "" ;
      float ST(Time, num_st_layers, south_north, west_east) ;
            ST:units = "" ;
            ST:description = "" ;
      float GHT(Time, num_metgrid_levels, south_north, west_east) ;
            GHT:units = "m" ;
            GHT:description = "Height" ;
      float HGTTROP(Time, south_north, west_east) ;
            HGTTROP:units = "m" ;
            HGTTROP:description = "Height of tropopause" ;
      float TTROP(Time, south_north, west_east) ;
            TTROP:units = "K" ;
            TTROP:description = "Temperature at tropopause" ;
```

```
float PTROPNN(Time, south_north, west_east) ;
     PTROPNN:units = "Pa" ;
     PTROPNN:description = "PTROP, used for nearest neighbor interp" ;
float PTROP(Time, south_north, west_east) ;
     PTROP:units = "Pa" ;
     PTROP:description = "Pressure of tropopause" ;
float VTROP(Time, south_north_stag, west_east) ;
     VTROP:units = "m s-1" ;
     VTROP:description = "V                 at tropopause" ;
float UTROP(Time, south_north, west_east_stag) ;
     UTROP:units = "m s-1" ;
     UTROP:description = "U                 at tropopause" ;
float HGTMAXW(Time, south_north, west_east) ;
     HGTMAXW:units = "m" ;
     HGTMAXW:description = "Height of max wind level" ;
float TMAXW(Time, south_north, west_east) ;
     TMAXW:units = "K" ;
     TMAXW:description = "Temperature at max wind level" ;
float PMAXWNN(Time, south_north, west_east) ;
     PMAXWNN:units = "Pa" ;
     PMAXWNN:description = "PMAXW, used for nearest neighbor interp" ;
float PMAXW(Time, south_north, west_east) ;
     PMAXW:units = "Pa" ;
     PMAXW:description = "Pressure of max wind level" ;
float VMAXW(Time, south_north_stag, west_east) ;
     VMAXW:units = "m s-1" ;
     VMAXW:description = "V                 at max wind" ;
float UMAXW(Time, south_north, west_east_stag) ;
     UMAXW:units = "m s-1" ;
     UMAXW:description = "U                 at max wind" ;
float SNOWH(Time, south_north, west_east) ;
     SNOWH:units = "m" ;
     SNOWH:description = "Physical Snow Depth" ;
float SNOW(Time, south_north, west_east) ;
     SNOW:units = "kg m-2" ;
     SNOW:description = "Water equivalent snow depth" ;
float SKINTEMP(Time, south_north, west_east) ;
     SKINTEMP:units = "K" ;
     SKINTEMP:description = "Skin temperature" ;
float SOILHGT(Time, south_north, west_east) ;
     SOILHGT:units = "m" ;
     SOILHGT:description = "Terrain field of source analysis" ;
float LANDSEA(Time, south_north, west_east) ;
     LANDSEA:units = "proprtn" ;
     LANDSEA:description = "Land/Sea flag (1=land, 0 or 2=sea)" ;
float SEAICE(Time, south_north, west_east) ;
     SEAICE:units = "proprtn" ;
     SEAICE:description = "Ice flag" ;
float ST100200(Time, south_north, west_east) ;
     ST100200:units = "K" ;
     ST100200:description = "T 100-200 cm below ground layer (Bottom)" ;
float ST040100(Time, south_north, west_east) ;
     ST040100:units = "K" ;
     ST040100:description = "T 40-100 cm below ground layer (Upper)" ;
float ST010040(Time, south_north, west_east) ;
     ST010040:units = "K" ;
     ST010040:description = "T 10-40 cm below ground layer (Upper)" ;
float ST000010(Time, south_north, west_east) ;
     ST000010:units = "K" ;
     ST000010:description = "T 0-10 cm below ground layer (Upper)" ;
float SM100200(Time, south_north, west_east) ;
     SM100200:units = "fraction" ;
     SM100200:description = "Soil Moist 100-200 cm below gr layer" ;
```

```
     float SM040100(Time, south_north, west_east) ;
          SM040100:units = "fraction" ;
          SM040100:description = "Soil Moist 40-100 cm below grn layer" ;
     float SM010040(Time, south_north, west_east) ;
          SM010040:units = "fraction" ;
          SM010040:description = "Soil Moist 10-40 cm below grn layer" ;
     float SM000010(Time, south_north, west_east) ;
          SM000010:units = "fraction" ;
          SM000010:description = "Soil Moist 0-10 cm below grn layer (Up)" ;
     float PSFC(Time, south_north, west_east) ;
          PSFC:units = "Pa" ;
          PSFC:description = "Surface Pressure" ;
     float RH(Time, num_metgrid_levels, south_north, west_east) ;
          RH:units = "%" ;
          RH:description = "Relative Humidity" ;
     float VV(Time, num_metgrid_levels, south_north_stag, west_east) ;
          VV:units = "m s-1" ;
          VV:description = "V" ;
     float UU(Time, num_metgrid_levels, south_north, west_east_stag) ;
          UU:units = "m s-1" ;
          UU:description = "U" ;
     float TT(Time, num_metgrid_levels, south_north, west_east) ;
          TT:units = "K" ;
          TT:description = "Temperature" ;
     float PMSL(Time, south_north, west_east) ;
          PMSL:units = "Pa" ;
          PMSL:description = "Sea-level Pressure" ;

// global attributes:
          :TITLE = "OUTPUT FROM METGRID V4.0" ;
          :SIMULATION_START_DATE = "2016-04-07_00:00:00" ;
          :WEST-EAST_GRID_DIMENSION = 74 ;
          :SOUTH-NORTH_GRID_DIMENSION = 61 ;
          :BOTTOM-TOP_GRID_DIMENSION = 27 ;
          :WEST-EAST_PATCH_START_UNSTAG = 1 ;
          :WEST-EAST_PATCH_END_UNSTAG = 73 ;
          :WEST-EAST_PATCH_START_STAG = 1 ;
          :WEST-EAST_PATCH_END_STAG = 74 ;
          :SOUTH-NORTH_PATCH_START_UNSTAG = 1 ;
          :SOUTH-NORTH_PATCH_END_UNSTAG = 60 ;
          :SOUTH-NORTH_PATCH_START_STAG = 1 ;
          :SOUTH-NORTH_PATCH_END_STAG = 61 ;
          :GRIDTYPE = "C" ;
          :DX = 30000.f ;
          :DY = 30000.f ;
          :DYN_OPT = 2 ;
          :CEN_LAT = 34.83001f ;
          :CEN_LON = -81.03f ;
          :TRUELAT1 = 30.f ;
          :TRUELAT2 = 60.f ;
          :MOAD_CEN_LAT = 34.83001f ;
          :STAND_LON = -98.f ;
          :POLE_LAT = 90.f ;
          :POLE_LON = 0.f ;
          :corner_lats = 28.17127f, 44.36657f, 39.63231f, 24.61906f, 28.17842f,
44.37617f, 39.57812f, 24.57806f, 28.03771f, 44.50592f, 39.76032f, 24.49431f,
28.04485f, 44.51553f, 39.70599f, 24.45341f ;
          :corner_lons = -93.64893f, -92.39661f, -66.00165f, -72.64047f, -
93.80048f, -92.59155f, -65.83557f, -72.5033f, -93.65717f, -92.3829f, -65.9313f,
-72.68539f, -93.80841f, -92.57831f, -65.76495f, -72.54843f ;
          :MAP_PROJ = 1 ;
          :MMINLU = "MODIFIED_IGBP_MODIS_NOAH" ;
          :NUM_LAND_CAT = 21 ;
```

```
            :ISWATER = 17 ;
            :ISLAKE = 21 ;
            :ISICE = 15 ;
            :ISURBAN = 13 ;
            :ISOILWATER = 14 ;
            :grid_id = 1 ;
            :parent_id = 1 ;
            :i_parent_start = 1 ;
            :j_parent_start = 1 ;
            :i_parent_end = 74 ;
            :j_parent_end = 61 ;
            :parent_grid_ratio = 1 ;
            :NUM_METGRID_SOIL_LEVELS = 4 ;
            :FLAG_METGRID = 1 ;
            :FLAG_EXCLUDED_MIDDLE = 0 ;
            :FLAG_SOIL_LAYERS = 1 ;
            :FLAG_SNOW = 1 ;
            :FLAG_PSFC = 1 ;
            :FLAG_SM000010 = 1 ;
            :FLAG_SM010040 = 1 ;
            :FLAG_SM040100 = 1 ;
            :FLAG_SM100200 = 1 ;
            :FLAG_ST000010 = 1 ;
            :FLAG_ST010040 = 1 ;
            :FLAG_ST040100 = 1 ;
            :FLAG_ST100200 = 1 ;
            :FLAG_SLP = 1 ;
            :FLAG_SNOWH = 1 ;
            :FLAG_SOILHGT = 1 ;
            :FLAG_UTROP = 1 ;
            :FLAG_VTROP = 1 ;
            :FLAG_TTROP = 1 ;
            :FLAG_PTROP = 1 ;
            :FLAG_PTROPNN = 1 ;
            :FLAG_HGTTROP = 1 ;
            :FLAG_UMAXW = 1 ;
            :FLAG_VMAXW = 1 ;
            :FLAG_TMAXW = 1 ;
            :FLAG_PMAXW = 1 ;
            :FLAG_PMAXWNN = 1 ;
            :FLAG_HGTMAXW = 1 ;
            :FLAG_MF_XY = 1 ;
            :FLAG_LAI12M = 1 ;
            :FLAG_LAKE_DEPTH = 1 ;
}
```