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

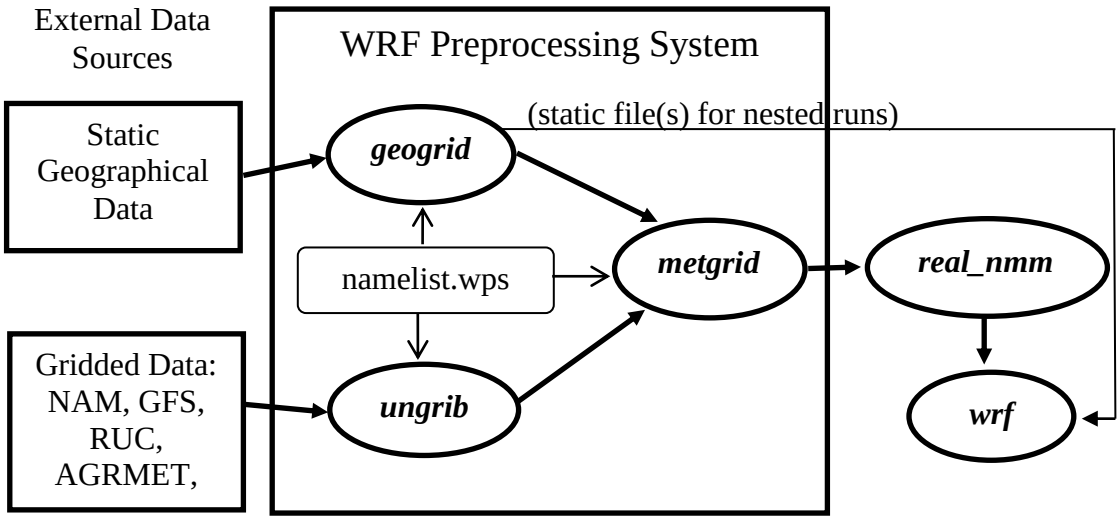
Chapter 3: WRF Preprocessing System (WPS)

Table of Contents

Introduction

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

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



The data flow between the programs of the WPS is shown in the figure above. Each of the WPS programs reads parameters from a common namelist file, as shown in the figure. This namelist file has separate namelist records for each of the programs and a shared namelist record, which defines parameters that are used by more than one WPS program. Not shown in the figure are additional table files that are used by individual programs. These tables provide additional control over the programs' operation, though they generally do not need to be changed by the user. The GEOGRID.TBL, METGRID.TBL, and Vtable files are explained later in this document, 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 MPICH libraries and suitable compilers are available, the metgrid and geogrid programs may be compiled for distributed memory execution, which allows large model domains to be processed in less time. The work performed by the ungrib program is not amenable to parallelization, so ungrib may only be run on a single processor.

Function of Each WPS Program

The WPS consists of three independent programs: *geogrid*, *ungrib*, and *metgrid*. Also included in the WPS are several utility programs, which are described in the section on utility programs. A brief description of each of the three main programs is given below, with further details presented in subsequent sections.

Program geogrid

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

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

Output from geogrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, geogrid can be made to write its output in NetCDF for easy visualization using external software packages, including ncview, 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 of the format). The GRIB files contain time-varying meteorological fields and are typically from another regional or global model, such as NCEP's NAM or GFS models. The ungrib program can read GRIB Edition 1 and, if compiled with a "GRIB2" option, GRIB Edition 2 files.

GRIB files typically contain more fields than are needed to initialize WRF. Both versions of the GRIB format use various codes to identify the variables and levels in the GRIB file. Ungrib uses tables of these codes – called Vtables, for "variable tables" – to define which fields to extract from the GRIB file and write to the intermediate format. Details about the codes can be found in the WMO GRIB documentation and in documentation from the originating center. Vtables for common GRIB model output files are provided with the ungrib software.

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

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

Program metgrid

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

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

Output from metgrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, metgrid can be made to write its output in NetCDF for easy visualization using external software packages, including the new version of RIP4.

Running the WPS

Note: For software requirements and how to compile the WRF Preprocessing System package, see Chapter 2.

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 = 'NMM',
 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     = '10m','2m',
 dx = 0.289153,
 dy = 0.287764,
 map_proj = 'rotated_ll',
 ref_lat   =  34.83,
 ref_lon   = -81.03,
 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 and must be set to `rotated_ll` for WRF-NMM.

Besides setting variables related to the projection, location, and coverage of model domains, the path to the static geographical data sets must be correctly specified with the `geog_data_path` variable. Also, the user may select which resolution of static data geogrid will interpolate from using the `geog_data_res` variable, whose value should match one of the resolutions of data in the GEOGRID.TBL. If the full set of static data are downloaded from the WRF download page, possible resolutions include `'30s'`, `'2m'`, `'5m'`, and `'10m'`, corresponding to 30-arc-second data, 2-, 5-, and 10-arc-minute data.

Depending on the value of the `wrf_core` namelist variable, the appropriate GEOGRID.TBL file must be used with geogrid, since the grid staggerings 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.NMM
```

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_nmm.d01.nc
-rw-r--r-- 1  4745324 geo_nmm.d02.nc
drwxr-xr-x 4     4096 geogrid
lrwxrwxrwx 1       23 geogrid.exe -> geogrid/src/geogrid.exe
-rw-r--r-- 1    11169 geogrid.log
-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 = 'NMM',
 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_nmm.d01.nc
-rw-r--r-- 1  4745324 geo_nmm.d02.nc
drwxr-xr-x 4     4096 geogrid
lrwxrwxrwx 1       23 geogrid.exe -> geogrid/src/geogrid.exe
-rw-r--r-- 1    11169 geogrid.log
lrwxrwxrwx 1       38 GRIBFILE.AAA -> /data/gfs/gfs_080324_12_00
lrwxrwxrwx 1       38 GRIBFILE.AAB -> /data/gfs/gfs_080324_12_06
-rwxr-xr-x 1     1328 link_grib.csh
drwxr-xr-x 3     4096 metgrid
lrwxrwxrwx 1       23 metgrid.exe -> metgrid/src/metgrid.exe
-rw-r--r-- 1     1094 namelist.wps
-rw-r--r-- 1     1987 namelist.wps.all_options
-rw-r--r-- 1     1075 namelist.wps.global
-rw-r--r-- 1      652 namelist.wps.nmm
-rw-r--r-- 1     4786 README
drwxr-xr-x 4     4096 ungrib
lrwxrwxrwx 1       21 ungrib.exe -> ungrib/src/ungrib.exe
drwxr-xr-x 3     4096 util
lrwxrwxrwx 1       33 Vtable -> ungrib/Variable_Tables/Vtable.GFS
```

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

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

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

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

will be written to the end of the ungrib.output file, and the intermediate files should appear in the current working directory. The intermediate files written by ungrib will have names of the form `FILE:`*`YYYY-MM-DD_HH`* (unless, of course, the `prefix` variable was set to a prefix other than `'FILE'`).

```
> ls
drwxr-xr-x 2       4096 arch
-rwxr-xr-x 1       1672 clean
-rwxr-xr-x 1       3510 compile
-rw-r--r-- 1      85973 compile.output
-rwxr-xr-x 1       4257 configure
-rw-r--r-- 1       2486 configure.wps
-rw-r--r-- 1  154946888 FILE:2008-03-24_12
-rw-r--r-- 1  154946888 FILE:2008-03-24_18
-rw-r--r-- 1    1957004 geo_nmm.d01.nc
-rw-r--r-- 1    4745324 geo_nmm.d02.nc
drwxr-xr-x 4       4096 geogrid
```

```
      lrwxrwxrwx 1         23 geogrid.exe -> geogrid/src/geogrid.exe
      -rw-r--r-- 1      11169 geogrid.log
      lrwxrwxrwx 1         38 GRIBFILE.AAA -> /data/gfs/gfs_080324_12_00
      lrwxrwxrwx 1         38 GRIBFILE.AAB -> /data/gfs/gfs_080324_12_06
      -rwxr-xr-x 1       1328 link_grib.csh
      drwxr-xr-x 3       4096 metgrid
      lrwxrwxrwx 1         23 metgrid.exe -> metgrid/src/metgrid.exe
      -rw-r--r-- 1       1094 namelist.wps
      -rw-r--r-- 1       1987 namelist.wps.all_options
      -rw-r--r-- 1       1075 namelist.wps.global
      -rw-r--r-- 1        652 namelist.wps.nmm
      -rw-r--r-- 1       4786 README
      drwxr-xr-x 4       4096 ungrib
      lrwxrwxrwx 1         21 ungrib.exe -> ungrib/src/ungrib.exe
      -rw-r--r-- 1       1418 ungrib.log
      -rw-r--r-- 1      27787 ungrib.output
      drwxr-xr-x 3       4096 util
      lrwxrwxrwx 1         33 Vtable ->
ungrib/Variable_Tables/Vtable.GFS
```

Step 3: Horizontally interpolating meteorological data with metgrid

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

```
&share
 wrf_core = 'NMM',
 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.NMM
```

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_nmm.d01.nc
-rw-r--r-- 1    4745324 geo_nmm.d02.nc
drwxr-xr-x 4       4096 geogrid
lrwxrwxrwx 1         23 geogrid.exe -> geogrid/src/geogrid.exe
-rw-r--r-- 1      11169 geogrid.log
lrwxrwxrwx 1         38 GRIBFILE.AAA -> /data/gfs/gfs_080324_12_00
lrwxrwxrwx 1         38 GRIBFILE.AAB -> /data/gfs/gfs_080324_12_06
-rwxr-xr-x 1       1328 link_grib.csh
-rw-r--r-- 1    5217648 met_nmm.d01.2008-03-24_12:00:00.nc
-rw-r--r-- 1    5217648 met_nmm.d01.2008-03-24_18:00:00.nc
-rw-r--r-- 1   12658200 met_nmm.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

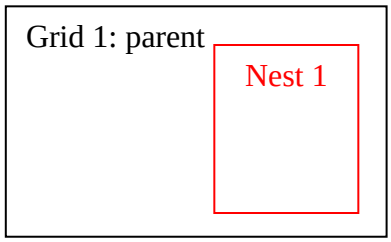
At this time, the WRF-NMM supports one-way and two-way stationary and moving (if running an HWRF configuration, see HWRF User's Guide) nests. Because the WRF-NMM nesting strategy was targeted towards the the capability of moving nests, time-invariant information, such as topography, soil type, albedo, etc. for a nest must be acquired over the entire domain of the coarsest grid even though, for a stationary nest, that information will only be used over the location where the nest is initialized.

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

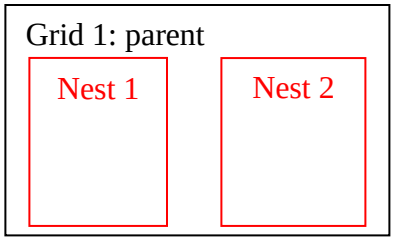
The number of grids is unlimited. Grids may be located side by side (i.e., two nests may be children of the same parent and located on the same nest level), or telescopically nested. The nesting ratio for the WRF-NMM is always 3. Hence, the grid spacing of a nest is always 1/3 of its parent.

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

For example:



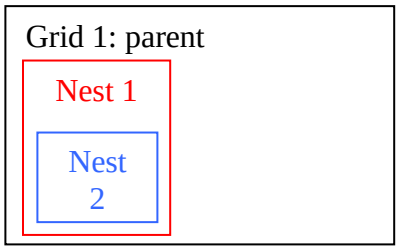
OR



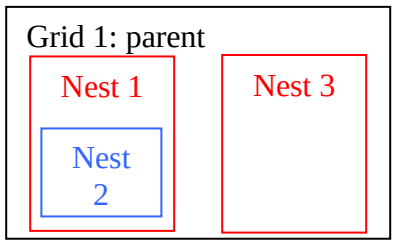
will create an output file for the parent domain: ***geo_nmm.d01.nc*** and one higher resolution output file for nest level one: ***geo_nmm_nest.l01.nc***

If, however, two telescopic nests are defined (nest 1 inside the parent and nest 2 inside nest 1), then two additional static files will be created. Even if an additional nest 3 was added at the same grid spacing as nest1, or at the same grid spacing as nest 2, there would still be only two additional static files created.

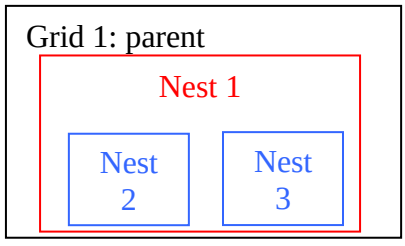
For example:



OR



OR



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

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

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

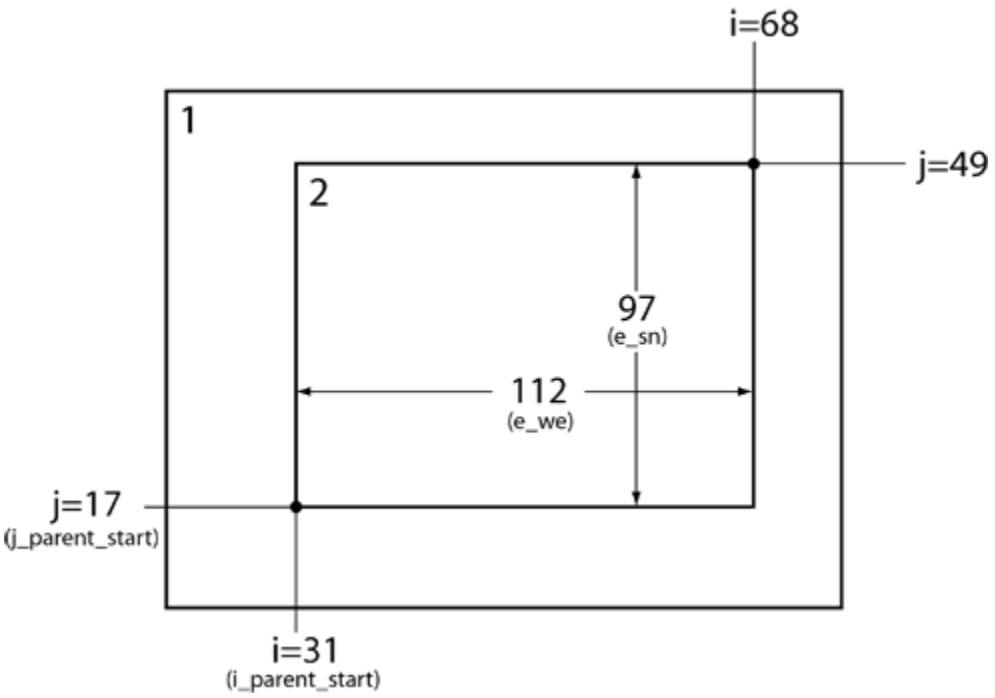
The remaining changes are to the "***geogrid***" namelist record. In this record, the parent of each nest must be specified with the ***parent_id*** variable. Every nest must be a child of exactly one other nest, with the coarse domain being its own parent. Related to the identity of a nest's parent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

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

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

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

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



Selecting Between USGS and MODIS-based Land Use Classifications

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

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

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

the user should instead specify

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

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

Selecting Static Data for the Gravity Wave Drag Scheme

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

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

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

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

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

```
geog_data_res = ‘10m+30s’,
```

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

Using Multiple Meteorological Data Sources

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

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

or

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

The second metgrid capability – that of interpolating data from multiple sources – may be useful in situations where two or more complementary data sets need to be combined to produce the full input data needed by `real`. To interpolate from multiple sources of time-varying, meteorological data, the `fg_name` variable in the `&metgrid` namelist record should be set to a list of prefixes of intermediate files, including path information when necessary. When multiple path-prefixes are given, and the same meteorological field is available from more than one of the sources, data from the last-specified source will take priority over all preceding sources. Thus, data sources may be prioritized by the order in which the sources are given.

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

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

To simplify the process of extracting fields from GRIB files, the `prefix` namelist variable in the `&ungrib` record may be employed. This variable allows the user to control the names of (and paths to) the intermediate files that are created by ungrib. The utility of

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

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

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

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

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

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

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

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

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

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

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

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

For the sake of clarity, the fixed file may be renamed to remove any date information, for example, by renaming it to simply `NARR_FIXED`, since the fields in the file are static. In this example, we note that the NARR fixed data are only available at a specific time,

1979 November 08 at 0000 UTC, and thus, the user would need to set the correct starting and ending time for the data in the `&share` namelist record before running ungrib on the NARR fixed file; of course, the times should be re-set before metgrid is run.

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

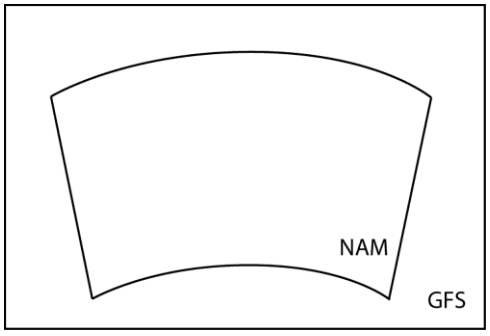
```
&metgrid
 constants_name = 'NARR_FIXED',
 fg_name = 'NARR_3D', 'NARR_SFC'
/
```

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

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

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

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



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

Alternative Initialization of Lake SSTs

The default treatment of sea-surface temperatures – both for oceans and lakes – in the metgrid program involves simply interpolating the SST field from the intermediate files to all water points in the WRF domain. However, if the lakes that are resolved in the WRF domain are not resolved in the GRIB data, and especially if those lakes are geographically distant from resolved water bodies, the SST field over lakes will most

likely be extrapolated from the nearest resolved water bodies in the GRIB data; this situation can lead to lake SST values that are either unrealistically warm or unrealistically cold.

Without a higher-resolution SST field for metgrid to use, one alternative to extrapolating SST values for lakes is to manufacture a "best guess" at the SST for lakes. In the metgrid and real programs, this can be done using a combination of a special land use data set that distinguishes between lakes and oceans, and a field to be used as a proxy for SST over lakes. A special land use data set is necessary, since WRF's real pre-processing program needs to know where the manufactured SST field should be used instead of the interpolated SST field from the GRIB data.

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

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

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

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

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

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

3. After running the ungrib program, use the avg_tsfc.exe utility program to create an intermediate file containing a daily-average surface air temperature field, which will

be substituted for the SST field only over lakes by the real program; for more information on the avg_tsfc.exe utility, see the section on WPS utility programs.

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

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

Parallelism in the WPS

If the dimensions of the domains to be processed by the WPS become too large to fit in the memory of a single CPU, it is possible to run the geogrid and metgrid programs in a distributed memory configuration. In order to compile geogrid and metgrid for distributed memory execution, the user must have MPI libraries installed on the target machine, and must have compiled WPS using one of the "DM parallel" configuration options. Upon successful compilation, the geogrid and metgrid programs may be run with the *mpirun* or *mpiexec* commands, or through a batch queuing system, depending on the machine.

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 new namelist record in the namelist.wps file:

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

Within the `&mod_levs` namelist record, the variable `press_pa` is used to specify a list of levels to keep; the specified levels should match values of `xlvl` in the intermediate format files (see the discussion of the WPS intermediate format for more information on

the fields of the intermediate files). The mod_levs program takes two command-line arguments as its input. The first argument is the name of the intermediate file to operate on, and the second argument is the name of the output file to be written.

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

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

C. calc_ecmwf_p.exe

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

The plotgrids.exe program is an NCAR Graphics-based utility whose purpose is to plot the locations of all nests defined in the namelist.wps file. The program operates on the namelist.wps file, and thus, may be run without having run any of the three main WPS programs. Upon successful completion, plotgrids produces an NCAR Graphics metafile, gmeta, which may be viewed using the idt command. The coarse domain is drawn to fill the plot frame, a map outline with political boundaries is drawn over the coarse domain, and any nested domains are drawn as rectangles outlining the extent of each nest. This utility may be useful particularly during initial placement of domains, at which time the user can iteratively adjust the locations of nests by editing the namelist.wps file, running plotgrids.exe, and determining a set of adjustments to the nest locations. *Currently, this utility does not work for ARW domains that use the latitude-longitude projection* (i.e., when `map_proj = 'lat-lon'`).

E. g1print.exe

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

F. g2print.exe

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

G. plotfmt.exe

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

H. rd_intermediate.exe

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

WRF Domain Wizard

WRF Domain Wizard is a graphical user interface (GUI) which interacts with the WPS and enables users to easily define domains, create a ***namelist.wps*** and run ***geogrid, ungrib,*** and ***metgrid***. A helpful online tutorial can be found at: *http://wrfportal.org/DomainWizard.html*.

Writing Meteorological Data to the Intermediate Format

The role of the ungrib program is to decode GRIB data sets into a simple intermediate format that is understood by metgrid. If meteorological data are not available in GRIB Edition 1 or GRIB Edition 2 formats, the user is responsible for writing such data into the intermediate file format. Fortunately, the intermediate format is relatively simple, consisting of a sequence of unformatted Fortran writes. It is important to note that these *unformatted writes use big-endian byte order*, which can typically be specified with compiler flags. Below, we describe the WPS intermediate format; users interested in the SI or MM5 intermediate formats can first gain familiarity with the WPS format, which is very similar, and later examine the Fortran subroutines that read and write all three intermediate formats (metgrid/src/read_met_module.F90 and metgrid/src/write_met_module.F90, respectively).

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

```
integer :: version              ! Format version (must =5 for WPS format)
integer :: nx, ny               ! x- and y-dimensions of 2-d array
integer :: iproj                ! Code for projection of data in array:
                                !       0 = cylindrical equidistant
                                !       1 = Mercator
                                !       3 = Lambert conformal conic
                                !       4 = Gaussian (global only!)
```

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


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

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

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

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

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

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

```
end if

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

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

Creating and Editing Vtables

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

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

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

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

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

Given a GRIB code, the "Level Type", "From Level1", and "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*.

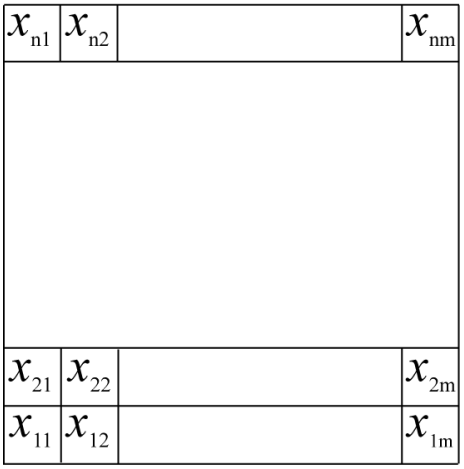
The final group of fields, which provide GRIB2-specific information, are found under the column headings below.

```
|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.

Description of the Namelist Variables

A. SHARE section

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

B. GEOGRID section

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

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

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

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

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

5. S_WE : A list of MAX_DOM integers which should all be set to 1. Default value is 1. **For WRF-NMM nests, see note on page 3-15.**

6. E_WE : A list of MAX_DOM integers specifying, for each nest, the nest's full west-east dimension. For nested domains, `e_we` must be one greater than an integer multiple of the nest's `parent_grid_ratio` (i.e., `e_ew` = *n**`parent_grid_ratio`+1 for some positive integer *n*). No default value. **For WRF-NMM nests, see note on page 3-15.**

7. S_SN : A list of MAX_DOM integers which should all be set to 1. Default value is 1. **For WRF-NMM nests, see note on page 3-15.**

8. E_SN : A list of MAX_DOM integers specifying, for each nest, the nest's full south-north dimension. For nested domains, `e_sn` must be one greater than an integer multiple of the nest's `parent_grid_ratio` (i.e., `e_sn` = *n**`parent_grid_ratio`+1 for some positive integer *n*). No default value. **For WRF-NMM nests, see note on page 3-15.**

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

H V H V H V H (V)
V H V H V H V (H)
H V H V H V H (V)
V H V H V H V (H)
H V H V H V H (V)

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

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

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

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

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

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

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

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

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

13. REF_LAT : A real value specifying the latitude part of a (latitude, longitude) location whose (i,j) location in the simulation domain is known. For ARW, `ref_lat` gives the

latitude of the center-point of the coarse domain by default (i.e., when `ref_x` and `ref_y` are not specified). For NMM, `ref_lat` always gives the latitude to which the origin is rotated. No default value.

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

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

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

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

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

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

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

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

22. GEOG_DATA_PATH : A character string giving the path, either relative or absolute, to the directory where the geographical data directories may be found. This path is the

one to which `rel_path` specifications in the GEOGRID.TBL file are given in relation to. No default value.

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

C. UNGRIB section

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

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

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

D. METGRID section

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

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

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

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

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

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

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

Description of GEOGRID.TBL Options

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

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

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

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

4. INTERP_OPTION : A sequence of one or more character strings, which are the names of interpolation methods to be used when horizontally interpolating the field. Available

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

18. DF_DY : When `df_dy` is assigned a character string value, the effect is to cause geogrid to compute the directional derivative of the field in the y-direction using a central

difference along the interior of the domain, or a one-sided difference at the boundary of the domain; the derivative field will be named according to the character string assigned to the keyword `df_dy`. Default value is null (i.e., no derivative field is computed).

19. Z_DIM_NAME : For 3-dimensional output fields, a character string giving the name of the vertical dimension, or z-dimension. A continuous field may have multiple levels, and thus be a 3-dimensional field, and a categorical field may take the form of a 3-dimensional field if it is written out as fractional fields for each category. No default value.

Description of index Options

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

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

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

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

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

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

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

7. DY : A real value giving the grid spacing in the y-direction of the data set. If `projection` is one of `lambert`, `polar`, `mercator`, `albers_nad83`, or `polar_wgs84`, `dy`

gives the grid spacing in meters; if `projection` is `regular_ll`, `dy` gives the grid spacing in degrees. No default value.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Description of METGRID.TBL Options

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

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

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

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

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

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

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

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

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

9. INTERP_OPTION : A sequence of one or more names of interpolation methods to be used when horizontally interpolating the field. Available interpolation methods are: `average_4pt`, `average_16pt`, `wt_average_4pt`, `wt_average_16pt`, `nearest_neighbor`, `four_pt`, `sixteen_pt`, `search`, `average_gcell(`*r*`)` ; for the grid cell average method (`average_gcell`), the optional argument *r* specifies the minimum ratio of source data resolution to simulation grid resolution at which the method will be applied; if a ratio is not specified, *r* = 0.0, and the option is used for any ratio. When a sequence of two or more methods are given, the methods should be separated by a + sign. Default value is `nearest_neighbor`.

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

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

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

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

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

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

16. FILL_LEV : The `fill_lev` keyword, which may be specified multiple times within a table section, specifies how a level of the field should be filled if that level does not already exist. A generic value for the keyword takes the form *DLEVEL*:*FIELD*(*SLEVEL*), where *DLEVEL* specifies the level in the field to be filled, *FIELD* specifies the source field from which to copy levels, and *SLEVEL* specifies the level within the source field to use. *DLEVEL* may either be an integer or the string `all`. *FIELD* may either be the name of another field, the string `const`, or the string `vertical_index`. If *FIELD* is specified as `const`, then *SLEVEL* is a constant value that will be used to fill with; if *FIELD* is specified as `vertical_index`, then (*SLEVEL*) must not be specified, and the value of the vertical index of the source field is used; if *DLEVEL* is 'all', then all levels from the field specified by the `level_template` keyword are used to fill the corresponding levels in the field, one at a time. No default value.

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

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

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

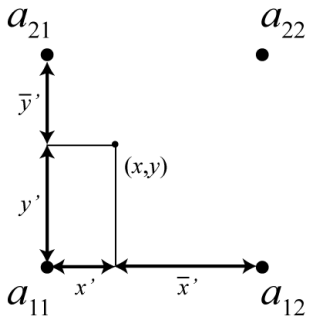
20. VERTICAL_INTERP_OPTION : A character string specifying the vertical interpolation method that should be used when vertically interpolating to missing points. Currently, this option is not implemented. No default value.

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

Available Interpolation Options in Geogrid and Metgrid

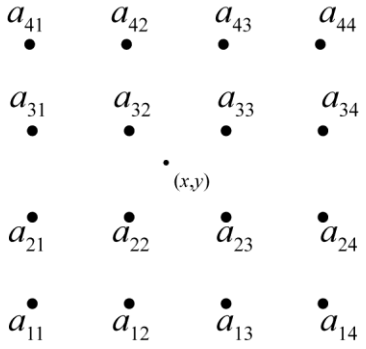
Through the GEOGRID.TBL and METGRID.TBL files, the user can control the method by which source data – either static fields in the case of geogrid or meteorological fields in the case of metgrid – are interpolated. In fact, a list of interpolation methods may be given, in which case, if it is not possible to employ the *i*-th method in the list, the (*i*+1)-st method will be employed, until either some method can be used or there are no methods left to try in the list. For example, to use a four-point bi-linear interpolation scheme for a field, we could specify `interp_option=four_pt`. However, if the field had areas of missing values, which could prevent the `four_pt` option from being used, we could request that a simple four-point average be tried if the `four_pt` method couldn't be used by specifying `interp_option=four_pt+average_4pt` instead. Below, each of the available interpolation options in the WPS are described conceptually; for the details of each method, the user is referred to the source code in the file WPS/geogrid/src/interp_options.F.

1. four_pt : Four-point bi-linear interpolation



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

2. sixteen_pt : Sixteen-point overlapping parabolic interpolation



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

3. average_4pt : Simple four-point average interpolation

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

4. wt_average_4pt : Weighted four-point average interpolation

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

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

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

5. average_16pt : Simple sixteen-point average interpolation

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

6. wt_average_16pt : Weighted sixteen-point average interpolation

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

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

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

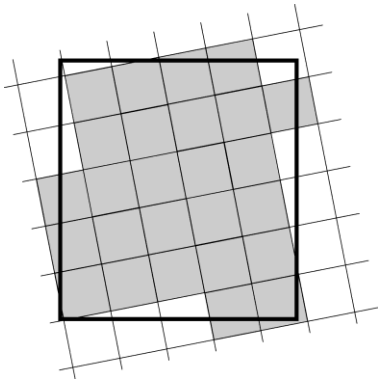
7. nearest_neighbor : Nearest neighbor interpolation

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

8. search : Breadth-first search interpolation

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

9. average_gcell : Model grid-cell average



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

Land Use and Soil Categories in the Static Data

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

Descriptions of the 24 land use categories and 16 soil categories are provided in the tables below.

Table 1: USGS 24-category Land Use Categories

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

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

Land Use Category	Land Use Description
1	Evergreen Needleleaf Forest
2	Evergreen Broadleaf Forest
3	Deciduous Needleleaf Forest
4	Deciduous Broadleaf Forest
5	Mixed Forests
6	Closed Shrublands
7	Open Shrublands
8	Woody Savannas
9	Savannas
10	Grasslands
11	Permanent Wetlands

12	Croplands
13	Urban and Built-Up
14	Cropland/Natural Vegetation Mosaic
15	Snow and Ice
16	Barren or Sparsely Vegetated
17	Water
18	Wooded Tundra
19	Mixed Tundra
20	Barren Tundra

Table 3: 16-category Soil Categories

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

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 of the output from the ncdump program when run on a typical geo_nmm.d01.nc file.

```
netcdf geo_nmm.d01 {
dimensions:
       Time = UNLIMITED ; // (1 currently)
       DateStrLen = 19 ;
       west_east = 19 ;
       south_north = 39 ;
       land_cat = 24 ;
       soil_cat = 16 ;
       month = 12 ;
variables:
       char Times(Time, DateStrLen) ;
       float XLAT_M(Time, south_north, west_east) ;
              XLAT_M:FieldType = 104 ;
```

```
        XLAT_M:MemoryOrder = "XY " ;
        XLAT_M:units = "degrees latitude" ;
        XLAT_M:description = "Latitude on mass grid" ;
        XLAT_M:stagger = "M" ;
        XLAT_M:sr_x = 1 ;
        XLAT_M:sr_y = 1 ;
    float XLONG_M(Time, south_north, west_east) ;
        XLONG_M:FieldType = 104 ;
        XLONG_M:MemoryOrder = "XY " ;
        XLONG_M:units = "degrees longitude" ;
        XLONG_M:description = "Longitude on mass grid" ;
        XLONG_M:stagger = "M" ;
        XLONG_M:sr_x = 1 ;
        XLONG_M:sr_y = 1 ;
    float XLAT_V(Time, south_north, west_east) ;
        XLAT_V:FieldType = 104 ;
        XLAT_V:MemoryOrder = "XY " ;
        XLAT_V:units = "degrees latitude" ;
        XLAT_V:description = "Latitude on velocity grid" ;
        XLAT_V:stagger = "V" ;
        XLAT_V:sr_x = 1 ;
        XLAT_V:sr_y = 1 ;
    float XLONG_V(Time, south_north, west_east) ;
        XLONG_V:FieldType = 104 ;
        XLONG_V:MemoryOrder = "XY " ;
        XLONG_V:units = "degrees longitude" ;
        XLONG_V:description = "Longitude on velocity grid" ;
        XLONG_V:stagger = "V" ;
        XLONG_V:sr_x = 1 ;
        XLONG_V:sr_y = 1 ;
    float E(Time, south_north, west_east) ;
        E:FieldType = 104 ;
        E:MemoryOrder = "XY " ;
        E:units = "-" ;
        E:description = "Coriolis E parameter" ;
        E:stagger = "M" ;
        E:sr_x = 1 ;
        E:sr_y = 1 ;
    float F(Time, south_north, west_east) ;
        F:FieldType = 104 ;
        F:MemoryOrder = "XY " ;
        F:units = "-" ;
        F:description = "Coriolis F parameter" ;
        F:stagger = "M" ;
        F:sr_x = 1 ;
        F:sr_y = 1 ;
    float LANDMASK(Time, south_north, west_east) ;
        LANDMASK:FieldType = 104 ;
        LANDMASK:MemoryOrder = "XY " ;
        LANDMASK:units = "none" ;
        LANDMASK:description = "Landmask : 1=land, 0=water" ;
        LANDMASK:stagger = "M" ;
        LANDMASK:sr_x = 1 ;
        LANDMASK:sr_y = 1 ;
    float LANDUSEF(Time, land_cat, south_north, west_east) ;
        LANDUSEF:FieldType = 104 ;
        LANDUSEF:MemoryOrder = "XYZ" ;
        LANDUSEF:units = "category" ;
        LANDUSEF:description = "24-category USGS landuse" ;
        LANDUSEF:stagger = "M" ;
        LANDUSEF:sr_x = 1 ;
        LANDUSEF:sr_y = 1 ;
    float LU_INDEX(Time, south_north, west_east) ;
```

```
       LU_INDEX:FieldType = 104 ;
       LU_INDEX:MemoryOrder = "XY " ;
       LU_INDEX:units = "category" ;
       LU_INDEX:description = "Dominant category" ;
       LU_INDEX:stagger = "M" ;
       LU_INDEX:sr_x = 1 ;
       LU_INDEX:sr_y = 1 ;
float HCNVX(Time, south_north, west_east) ;
       HCNVX:FieldType = 104 ;
       HCNVX:MemoryOrder = "XY " ;
       HCNVX:units = "whoknows" ;
       HCNVX:description = "something" ;
       HCNVX:stagger = "M" ;
       HCNVX:sr_x = 1 ;
       HCNVX:sr_y = 1 ;
float HSTDV(Time, south_north, west_east) ;
       HSTDV:FieldType = 104 ;
       HSTDV:MemoryOrder = "XY " ;
       HSTDV:units = "whoknows" ;
       HSTDV:description = "something" ;
       HSTDV:stagger = "M" ;
       HSTDV:sr_x = 1 ;
       HSTDV:sr_y = 1 ;
float HASYW(Time, south_north, west_east) ;
       HASYW:FieldType = 104 ;
       HASYW:MemoryOrder = "XY " ;
       HASYW:units = "whoknows" ;
       HASYW:description = "something" ;
       HASYW:stagger = "M" ;
       HASYW:sr_x = 1 ;
       HASYW:sr_y = 1 ;
float HASYS(Time, south_north, west_east) ;
       HASYS:FieldType = 104 ;
       HASYS:MemoryOrder = "XY " ;
       HASYS:units = "whoknows" ;
       HASYS:description = "something" ;
       HASYS:stagger = "M" ;
       HASYS:sr_x = 1 ;
       HASYS:sr_y = 1 ;
float HASYSW(Time, south_north, west_east) ;
       HASYSW:FieldType = 104 ;
       HASYSW:MemoryOrder = "XY " ;
       HASYSW:units = "whoknows" ;
       HASYSW:description = "something" ;
       HASYSW:stagger = "M" ;
       HASYSW:sr_x = 1 ;
       HASYSW:sr_y = 1 ;
float HASYNW(Time, south_north, west_east) ;
       HASYNW:FieldType = 104 ;
       HASYNW:MemoryOrder = "XY " ;
       HASYNW:units = "whoknows" ;
       HASYNW:description = "something" ;
       HASYNW:stagger = "M" ;
       HASYNW:sr_x = 1 ;
       HASYNW:sr_y = 1 ;
float HLENW(Time, south_north, west_east) ;
       HLENW:FieldType = 104 ;
       HLENW:MemoryOrder = "XY " ;
       HLENW:units = "whoknows" ;
       HLENW:description = "something" ;
       HLENW:stagger = "M" ;
       HLENW:sr_x = 1 ;
       HLENW:sr_y = 1 ;
```

```
float HLENS(Time, south_north, west_east) ;
       HLENS:FieldType = 104 ;
       HLENS:MemoryOrder = "XY " ;
       HLENS:units = "whoknows" ;
       HLENS:description = "something" ;
       HLENS:stagger = "M" ;
       HLENS:sr_x = 1 ;
       HLENS:sr_y = 1 ;
float HLENSW(Time, south_north, west_east) ;
       HLENSW:FieldType = 104 ;
       HLENSW:MemoryOrder = "XY " ;
       HLENSW:units = "whoknows" ;
       HLENSW:description = "something" ;
       HLENSW:stagger = "M" ;
       HLENSW:sr_x = 1 ;
       HLENSW:sr_y = 1 ;
float HLENNW(Time, south_north, west_east) ;
       HLENNW:FieldType = 104 ;
       HLENNW:MemoryOrder = "XY " ;
       HLENNW:units = "whoknows" ;
       HLENNW:description = "something" ;
       HLENNW:stagger = "M" ;
       HLENNW:sr_x = 1 ;
       HLENNW:sr_y = 1 ;
float HANIS(Time, south_north, west_east) ;
       HANIS:FieldType = 104 ;
       HANIS:MemoryOrder = "XY " ;
       HANIS:units = "whoknows" ;
       HANIS:description = "something" ;
       HANIS:stagger = "M" ;
       HANIS:sr_x = 1 ;
       HANIS:sr_y = 1 ;
float HSLOP(Time, south_north, west_east) ;
       HSLOP:FieldType = 104 ;
       HSLOP:MemoryOrder = "XY " ;
       HSLOP:units = "whoknows" ;
       HSLOP:description = "something" ;
       HSLOP:stagger = "M" ;
       HSLOP:sr_x = 1 ;
       HSLOP:sr_y = 1 ;
float HANGL(Time, south_north, west_east) ;
       HANGL:FieldType = 104 ;
       HANGL:MemoryOrder = "XY " ;
       HANGL:units = "whoknows" ;
       HANGL:description = "something" ;
       HANGL:stagger = "M" ;
       HANGL:sr_x = 1 ;
       HANGL:sr_y = 1 ;
float HZMAX(Time, south_north, west_east) ;
       HZMAX:FieldType = 104 ;
       HZMAX:MemoryOrder = "XY " ;
       HZMAX:units = "whoknows" ;
       HZMAX:description = "something" ;
       HZMAX:stagger = "M" ;
       HZMAX:sr_x = 1 ;
       HZMAX:sr_y = 1 ;
float HGT_M(Time, south_north, west_east) ;
       HGT_M:FieldType = 104 ;
       HGT_M:MemoryOrder = "XY " ;
       HGT_M:units = "meters MSL" ;
       HGT_M:description = "Topography height" ;
       HGT_M:stagger = "M" ;
       HGT_M:sr_x = 1 ;
```

```
       HGT_M:sr_y = 1 ;
float HGT_V(Time, south_north, west_east) ;
       HGT_V:FieldType = 104 ;
       HGT_V:MemoryOrder = "XY " ;
       HGT_V:units = "meters MSL" ;
       HGT_V:description = "Topography height" ;
       HGT_V:stagger = "V" ;
       HGT_V:sr_x = 1 ;
       HGT_V:sr_y = 1 ;
float SOILTEMP(Time, south_north, west_east) ;
       SOILTEMP:FieldType = 104 ;
       SOILTEMP:MemoryOrder = "XY " ;
       SOILTEMP:units = "Kelvin" ;
       SOILTEMP:description = "Annual mean deep soil temperature" ;
       SOILTEMP:stagger = "M" ;
       SOILTEMP:sr_x = 1 ;
       SOILTEMP:sr_y = 1 ;
float SOILCTOP(Time, soil_cat, south_north, west_east) ;
       SOILCTOP:FieldType = 104 ;
       SOILCTOP:MemoryOrder = "XYZ" ;
       SOILCTOP:units = "category" ;
       SOILCTOP:description = "16-category top-layer soil type" ;
       SOILCTOP:stagger = "M" ;
       SOILCTOP:sr_x = 1 ;
       SOILCTOP:sr_y = 1 ;
float SOILCBOT(Time, soil_cat, south_north, west_east) ;
       SOILCBOT:FieldType = 104 ;
       SOILCBOT:MemoryOrder = "XYZ" ;
       SOILCBOT:units = "category" ;
       SOILCBOT:description = "16-category top-layer soil type" ;
       SOILCBOT:stagger = "M" ;
       SOILCBOT:sr_x = 1 ;
       SOILCBOT:sr_y = 1 ;
float ALBEDO12M(Time, month, south_north, west_east) ;
       ALBEDO12M:FieldType = 104 ;
       ALBEDO12M:MemoryOrder = "XYZ" ;
       ALBEDO12M:units = "percent" ;
       ALBEDO12M:description = "Monthly surface albedo" ;
       ALBEDO12M:stagger = "M" ;
       ALBEDO12M:sr_x = 1 ;
       ALBEDO12M:sr_y = 1 ;
float GREENFRAC(Time, month, south_north, west_east) ;
       GREENFRAC:FieldType = 104 ;
       GREENFRAC:MemoryOrder = "XYZ" ;
       GREENFRAC:units = "fraction" ;
       GREENFRAC:description = "Monthly green fraction" ;
       GREENFRAC:stagger = "M" ;
       GREENFRAC:sr_x = 1 ;
       GREENFRAC:sr_y = 1 ;
float SNOALB(Time, south_north, west_east) ;
       SNOALB:FieldType = 104 ;
       SNOALB:MemoryOrder = "XY " ;
       SNOALB:units = "percent" ;
       SNOALB:description = "Maximum snow albedo" ;
       SNOALB:stagger = "M" ;
       SNOALB:sr_x = 1 ;
       SNOALB:sr_y = 1 ;
float SLOPECAT(Time, south_north, west_east) ;
       SLOPECAT:FieldType = 104 ;
       SLOPECAT:MemoryOrder = "XY " ;
       SLOPECAT:units = "category" ;
       SLOPECAT:description = "Dominant category" ;
       SLOPECAT:stagger = "M" ;
```

```
              SLOPECAT:sr_x = 1 ;
              SLOPECAT:sr_y = 1 ;

// global attributes:
              :TITLE = "OUTPUT FROM GEOGRID V3.4" ;
              :SIMULATION_START_DATE = "0000-00-00_00:00:00" ;
              :WEST-EAST_GRID_DIMENSION = 19 ;
              :SOUTH-NORTH_GRID_DIMENSION = 39 ;
              :BOTTOM-TOP_GRID_DIMENSION = 0 ;
              :WEST-EAST_PATCH_START_UNSTAG = 1 ;
              :WEST-EAST_PATCH_END_UNSTAG = 19 ;
              :WEST-EAST_PATCH_START_STAG = 1 ;
              :WEST-EAST_PATCH_END_STAG = 19 ;
              :SOUTH-NORTH_PATCH_START_UNSTAG = 1 ;
              :SOUTH-NORTH_PATCH_END_UNSTAG = 39 ;
              :SOUTH-NORTH_PATCH_START_STAG = 1 ;
              :SOUTH-NORTH_PATCH_END_STAG = 39 ;
              :GRIDTYPE = "E" ;
              :DX = 0.289143f ;
              :DY = 0.287764f ;
              :DYN_OPT = 4 ;
              :CEN_LAT = 32.f ;
              :CEN_LON = -83.f ;
              :TRUELAT1 = 1.e+20f ;
              :TRUELAT2 = 1.e+20f ;
              :MOAD_CEN_LAT = 0.f ;
              :STAND_LON = 1.e+20f ;
              :POLE_LAT = 90.f ;
              :POLE_LON = 0.f ;
              :corner_lats = 26.39329f, 37.31068f, 37.31068f, 26.39329f,
26.40831f, 37.3276f, 37.29281f, 26.37742f, 0.f, 0.f, 0.f, 0.f, 0.f, 0.f, 0.f,
0.f ;
              :corner_lons = -88.78565f, -89.519f, -76.481f, -77.21435f, -
88.46474f, -89.1577f, -76.11986f, -76.89354f, 0.f, 0.f, 0.f, 0.f, 0.f, 0.f,
0.f, 0.f ;
              :MAP_PROJ = 203 ;
              :MMINLU = "USGS" ;
              :NUM_LAND_CAT = 24 ;
              :ISWATER = 16 ;
              :ISLAKE = -1 ;
              :ISICE = 24 ;
              :ISURBAN = 1 ;
              :ISOILWATER = 14 ;
              :grid_id = 1 ;
              :parent_id = 1 ;
              :i_parent_start = 0 ;
              :j_parent_start = 0 ;
              :i_parent_end = 19 ;
              :j_parent_end = 39 ;
              :parent_grid_ratio = 1 ;
              :sr_x = 1 ;
              :sr_y = 1 ;
}
```

In addition to the fields in a geogrid output file (e.g., geo_nmm.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_nmm.d01.2008-01-11_00\:00\:00 {
dimensions:
```

```
      Time = UNLIMITED ; // (1 currently)
      DateStrLen = 19 ;
      west_east = 19 ;
      south_north = 39 ;
      num_metgrid_levels = 27 ;
      num_sm_levels = 4 ;
      num_st_levels = 4 ;
      z-dimension0012 = 12 ;
      z-dimension0016 = 16 ;
      z-dimension0024 = 24 ;
variables:
      char Times(Time, DateStrLen) ;
      float PRES(Time, num_metgrid_levels, south_north, west_east) ;
            PRES:FieldType = 104 ;
            PRES:MemoryOrder = "XYZ" ;
            PRES:units = "" ;
            PRES:description = "" ;
            PRES:stagger = "M" ;
            PRES:sr_x = 1 ;
            PRES:sr_y = 1 ;
      float SMC_WPS(Time, num_sm_levels, south_north, west_east) ;
            SMC_WPS:FieldType = 104 ;
            SMC_WPS:MemoryOrder = "XYZ" ;
            SMC_WPS:units = "" ;
            SMC_WPS:description = "" ;
            SMC_WPS:stagger = "M" ;
            SMC_WPS:sr_x = 1 ;
            SMC_WPS:sr_y = 1 ;
      float STC_WPS(Time, num_st_levels, south_north, west_east) ;
            STC_WPS:FieldType = 104 ;
            STC_WPS:MemoryOrder = "XYZ" ;
            STC_WPS:units = "" ;
            STC_WPS:description = "" ;
            STC_WPS:stagger = "M" ;
            STC_WPS:sr_x = 1 ;
            STC_WPS:sr_y = 1 ;
      float GHT(Time, num_metgrid_levels, south_north, west_east) ;
            GHT:FieldType = 104 ;
            GHT:MemoryOrder = "XYZ" ;
            GHT:units = "m" ;
            GHT:description = "Height" ;
            GHT:stagger = "M" ;
            GHT:sr_x = 1 ;
            GHT:sr_y = 1 ;
      float SNOW(Time, south_north, west_east) ;
            SNOW:FieldType = 104 ;
            SNOW:MemoryOrder = "XY " ;
            SNOW:units = "kg m-2" ;
            SNOW:description = "Water equivalent snow depth" ;
            SNOW:stagger = "M" ;
            SNOW:sr_x = 1 ;
            SNOW:sr_y = 1 ;
      float SKINTEMP(Time, south_north, west_east) ;
            SKINTEMP:FieldType = 104 ;
```

```
            SKINTEMP:MemoryOrder = "XY " ;
            SKINTEMP:units = "K" ;
            SKINTEMP:description = "Skin temperature (can use for SST
also)" ;
            SKINTEMP:stagger = "M" ;
            SKINTEMP:sr_x = 1 ;
            SKINTEMP:sr_y = 1 ;
      float SOILHGT(Time, south_north, west_east) ;
            SOILHGT:FieldType = 104 ;
            SOILHGT:MemoryOrder = "XY " ;
            SOILHGT:units = "m" ;
            SOILHGT:description = "Terrain field of source analysis"
;
            SOILHGT:stagger = "M" ;
            SOILHGT:sr_x = 1 ;
            SOILHGT:sr_y = 1 ;
      float LANDSEA(Time, south_north, west_east) ;
            LANDSEA:FieldType = 104 ;
            LANDSEA:MemoryOrder = "XY " ;
            LANDSEA:units = "proprtn" ;
            LANDSEA:description = "Land/Sea flag (1=land, 0 or
2=sea)" ;
            LANDSEA:stagger = "M" ;
            LANDSEA:sr_x = 1 ;
            LANDSEA:sr_y = 1 ;
      float SEAICE(Time, south_north, west_east) ;
            SEAICE:FieldType = 104 ;
            SEAICE:MemoryOrder = "XY " ;
            SEAICE:units = "proprtn" ;
            SEAICE:description = "Ice flag" ;
            SEAICE:stagger = "M" ;
            SEAICE:sr_x = 1 ;
            SEAICE:sr_y = 1 ;
      float ST100200(Time, south_north, west_east) ;
            ST100200:FieldType = 104 ;
            ST100200:MemoryOrder = "XY " ;
            ST100200:units = "K" ;
            ST100200:description = "T 100-200 cm below ground layer
(Bottom)" ;
            ST100200:stagger = "M" ;
            ST100200:sr_x = 1 ;
            ST100200:sr_y = 1 ;
      float ST040100(Time, south_north, west_east) ;
            ST040100:FieldType = 104 ;
            ST040100:MemoryOrder = "XY " ;
            ST040100:units = "K" ;
            ST040100:description = "T 40-100 cm below ground layer
(Upper)" ;
            ST040100:stagger = "M" ;
            ST040100:sr_x = 1 ;
            ST040100:sr_y = 1 ;
      float ST010040(Time, south_north, west_east) ;
            ST010040:FieldType = 104 ;
            ST010040:MemoryOrder = "XY " ;
```

```
            ST010040:units = "K" ;
            ST010040:description = "T 10-40 cm below ground layer
(Upper)" ;
            ST010040:stagger = "M" ;
            ST010040:sr_x = 1 ;
            ST010040:sr_y = 1 ;
      float ST000010(Time, south_north, west_east) ;
            ST000010:FieldType = 104 ;
            ST000010:MemoryOrder = "XY " ;
            ST000010:units = "K" ;
            ST000010:description = "T 0-10 cm below ground layer
(Upper)" ;
            ST000010:stagger = "M" ;
            ST000010:sr_x = 1 ;
            ST000010:sr_y = 1 ;
      float SM100200(Time, south_north, west_east) ;
            SM100200:FieldType = 104 ;
            SM100200:MemoryOrder = "XY " ;
            SM100200:units = "kg m-3" ;
            SM100200:description = "Soil Moist 100-200 cm below gr
layer" ;
            SM100200:stagger = "M" ;
            SM100200:sr_x = 1 ;
            SM100200:sr_y = 1 ;
      float SM040100(Time, south_north, west_east) ;
            SM040100:FieldType = 104 ;
            SM040100:MemoryOrder = "XY " ;
            SM040100:units = "kg m-3" ;
            SM040100:description = "Soil Moist 40-100 cm below grn
layer" ;
            SM040100:stagger = "M" ;
            SM040100:sr_x = 1 ;
            SM040100:sr_y = 1 ;
      float SM010040(Time, south_north, west_east) ;
            SM010040:FieldType = 104 ;
            SM010040:MemoryOrder = "XY " ;
            SM010040:units = "kg m-3" ;
            SM010040:description = "Soil Moist 10-40 cm below grn
layer" ;
            SM010040:stagger = "M" ;
            SM010040:sr_x = 1 ;
            SM010040:sr_y = 1 ;
      float SM000010(Time, south_north, west_east) ;
            SM000010:FieldType = 104 ;
            SM000010:MemoryOrder = "XY " ;
            SM000010:units = "kg m-3" ;
            SM000010:description = "Soil Moist 0-10 cm below grn
layer (Up)" ;
            SM000010:stagger = "M" ;
            SM000010:sr_x = 1 ;
            SM000010:sr_y = 1 ;
      float PSFC(Time, south_north, west_east) ;
            PSFC:FieldType = 104 ;
            PSFC:MemoryOrder = "XY " ;
```

```
      PSFC:units = "Pa" ;
      PSFC:description = "Surface Pressure" ;
      PSFC:stagger = "M" ;
      PSFC:sr_x = 1 ;
      PSFC:sr_y = 1 ;
float RH(Time, num_metgrid_levels, south_north, west_east) ;
      RH:FieldType = 104 ;
      RH:MemoryOrder = "XYZ" ;
      RH:units = "%" ;
      RH:description = "Relative Humidity" ;
      RH:stagger = "M" ;
      RH:sr_x = 1 ;
      RH:sr_y = 1 ;
float VV(Time, num_metgrid_levels, south_north, west_east) ;
      VV:FieldType = 104 ;
      VV:MemoryOrder = "XYZ" ;
      VV:units = "m s-1" ;
      VV:description = "V" ;
      VV:stagger = "V" ;
      VV:sr_x = 1 ;
      VV:sr_y = 1 ;
float UU(Time, num_metgrid_levels, south_north, west_east) ;
      UU:FieldType = 104 ;
      UU:MemoryOrder = "XYZ" ;
      UU:units = "m s-1" ;
      UU:description = "U" ;
      UU:stagger = "V" ;
      UU:sr_x = 1 ;
      UU:sr_y = 1 ;
float TT(Time, num_metgrid_levels, south_north, west_east) ;
      TT:FieldType = 104 ;
      TT:MemoryOrder = "XYZ" ;
      TT:units = "K" ;
      TT:description = "Temperature" ;
      TT:stagger = "M" ;
      TT:sr_x = 1 ;
      TT:sr_y = 1 ;
float PMSL(Time, south_north, west_east) ;
      PMSL:FieldType = 104 ;
      PMSL:MemoryOrder = "XY " ;
      PMSL:units = "Pa" ;
      PMSL:description = "Sea-level Pressure" ;
      PMSL:stagger = "M" ;
      PMSL:sr_x = 1 ;
      PMSL:sr_y = 1 ;
float SLOPECAT(Time, south_north, west_east) ;
      SLOPECAT:FieldType = 104 ;
      SLOPECAT:MemoryOrder = "XY " ;
      SLOPECAT:units = "category" ;
      SLOPECAT:description = "Dominant category" ;
      SLOPECAT:stagger = "M" ;
      SLOPECAT:sr_x = 1 ;
      SLOPECAT:sr_y = 1 ;
float SNOALB(Time, south_north, west_east) ;
```

```
            SNOALB:FieldType = 104 ;
            SNOALB:MemoryOrder = "XY " ;
            SNOALB:units = "percent" ;
            SNOALB:description = "Maximum snow albedo" ;
            SNOALB:stagger = "M" ;
            SNOALB:sr_x = 1 ;
            SNOALB:sr_y = 1 ;
      float GREENFRAC(Time, z-dimension0012, south_north, west_east)
;
            GREENFRAC:FieldType = 104 ;
            GREENFRAC:MemoryOrder = "XYZ" ;
            GREENFRAC:units = "fraction" ;
            GREENFRAC:description = "Monthly green fraction" ;
            GREENFRAC:stagger = "M" ;
            GREENFRAC:sr_x = 1 ;
            GREENFRAC:sr_y = 1 ;
      float ALBEDO12M(Time, z-dimension0012, south_north, west_east)
;
            ALBEDO12M:FieldType = 104 ;
            ALBEDO12M:MemoryOrder = "XYZ" ;
            ALBEDO12M:units = "percent" ;
            ALBEDO12M:description = "Monthly surface albedo" ;
            ALBEDO12M:stagger = "M" ;
            ALBEDO12M:sr_x = 1 ;
            ALBEDO12M:sr_y = 1 ;
      float SOILCBOT(Time, z-dimension0016, south_north, west_east)
;
            SOILCBOT:FieldType = 104 ;
            SOILCBOT:MemoryOrder = "XYZ" ;
            SOILCBOT:units = "category" ;
            SOILCBOT:description = "16-category top-layer soil type"
;
            SOILCBOT:stagger = "M" ;
            SOILCBOT:sr_x = 1 ;
            SOILCBOT:sr_y = 1 ;
      float SOILCTOP(Time, z-dimension0016, south_north, west_east)
;
            SOILCTOP:FieldType = 104 ;
            SOILCTOP:MemoryOrder = "XYZ" ;
            SOILCTOP:units = "category" ;
            SOILCTOP:description = "16-category top-layer soil type"
;
            SOILCTOP:stagger = "M" ;
            SOILCTOP:sr_x = 1 ;
            SOILCTOP:sr_y = 1 ;
      float SOILTEMP(Time, south_north, west_east) ;
            SOILTEMP:FieldType = 104 ;
            SOILTEMP:MemoryOrder = "XY " ;
            SOILTEMP:units = "Kelvin" ;
            SOILTEMP:description = "Annual mean deep soil
temperature" ;
            SOILTEMP:stagger = "M" ;
            SOILTEMP:sr_x = 1 ;
            SOILTEMP:sr_y = 1 ;
```

```
float HGT_V(Time, south_north, west_east) ;
      HGT_V:FieldType = 104 ;
      HGT_V:MemoryOrder = "XY " ;
      HGT_V:units = "meters MSL" ;
      HGT_V:description = "Topography height" ;
      HGT_V:stagger = "V" ;
      HGT_V:sr_x = 1 ;
      HGT_V:sr_y = 1 ;
float HGT_M(Time, south_north, west_east) ;
      HGT_M:FieldType = 104 ;
      HGT_M:MemoryOrder = "XY " ;
      HGT_M:units = "meters MSL" ;
      HGT_M:description = "Topography height" ;
      HGT_M:stagger = "M" ;
      HGT_M:sr_x = 1 ;
      HGT_M:sr_y = 1 ;
float HZMAX(Time, south_north, west_east) ;
      HZMAX:FieldType = 104 ;
      HZMAX:MemoryOrder = "XY " ;
      HZMAX:units = "whoknows" ;
      HZMAX:description = "something" ;
      HZMAX:stagger = "M" ;
      HZMAX:sr_x = 1 ;
      HZMAX:sr_y = 1 ;
float HANGL(Time, south_north, west_east) ;
      HANGL:FieldType = 104 ;
      HANGL:MemoryOrder = "XY " ;
      HANGL:units = "whoknows" ;
      HANGL:description = "something" ;
      HANGL:stagger = "M" ;
      HANGL:sr_x = 1 ;
      HANGL:sr_y = 1 ;
float HSLOP(Time, south_north, west_east) ;
      HSLOP:FieldType = 104 ;
      HSLOP:MemoryOrder = "XY " ;
      HSLOP:units = "whoknows" ;
      HSLOP:description = "something" ;
      HSLOP:stagger = "M" ;
      HSLOP:sr_x = 1 ;
      HSLOP:sr_y = 1 ;
float HANIS(Time, south_north, west_east) ;
      HANIS:FieldType = 104 ;
      HANIS:MemoryOrder = "XY " ;
      HANIS:units = "whoknows" ;
      HANIS:description = "something" ;
      HANIS:stagger = "M" ;
      HANIS:sr_x = 1 ;
      HANIS:sr_y = 1 ;
float HLENNW(Time, south_north, west_east) ;
      HLENNW:FieldType = 104 ;
      HLENNW:MemoryOrder = "XY " ;
      HLENNW:units = "whoknows" ;
      HLENNW:description = "something" ;
      HLENNW:stagger = "M" ;
```

```
      HLENNW:sr_x = 1 ;
      HLENNW:sr_y = 1 ;
float HLENSW(Time, south_north, west_east) ;
      HLENSW:FieldType = 104 ;
      HLENSW:MemoryOrder = "XY " ;
      HLENSW:units = "whoknows" ;
      HLENSW:description = "something" ;
      HLENSW:stagger = "M" ;
      HLENSW:sr_x = 1 ;
      HLENSW:sr_y = 1 ;
float HLENS(Time, south_north, west_east) ;
      HLENS:FieldType = 104 ;
      HLENS:MemoryOrder = "XY " ;
      HLENS:units = "whoknows" ;
      HLENS:description = "something" ;
      HLENS:stagger = "M" ;
      HLENS:sr_x = 1 ;
      HLENS:sr_y = 1 ;
float HLENW(Time, south_north, west_east) ;
      HLENW:FieldType = 104 ;
      HLENW:MemoryOrder = "XY " ;
      HLENW:units = "whoknows" ;
      HLENW:description = "something" ;
      HLENW:stagger = "M" ;
      HLENW:sr_x = 1 ;
      HLENW:sr_y = 1 ;
float HASYNW(Time, south_north, west_east) ;
      HASYNW:FieldType = 104 ;
      HASYNW:MemoryOrder = "XY " ;
      HASYNW:units = "whoknows" ;
      HASYNW:description = "something" ;
      HASYNW:stagger = "M" ;
      HASYNW:sr_x = 1 ;
      HASYNW:sr_y = 1 ;
float HASYSW(Time, south_north, west_east) ;
      HASYSW:FieldType = 104 ;
      HASYSW:MemoryOrder = "XY " ;
      HASYSW:units = "whoknows" ;
      HASYSW:description = "something" ;
      HASYSW:stagger = "M" ;
      HASYSW:sr_x = 1 ;
      HASYSW:sr_y = 1 ;
float HASYS(Time, south_north, west_east) ;
      HASYS:FieldType = 104 ;
      HASYS:MemoryOrder = "XY " ;
      HASYS:units = "whoknows" ;
      HASYS:description = "something" ;
      HASYS:stagger = "M" ;
      HASYS:sr_x = 1 ;
      HASYS:sr_y = 1 ;
float HASYW(Time, south_north, west_east) ;
      HASYW:FieldType = 104 ;
      HASYW:MemoryOrder = "XY " ;
      HASYW:units = "whoknows" ;
```

```
            HASYW:description = "something" ;
            HASYW:stagger = "M" ;
            HASYW:sr_x = 1 ;
            HASYW:sr_y = 1 ;
      float HSTDV(Time, south_north, west_east) ;
            HSTDV:FieldType = 104 ;
            HSTDV:MemoryOrder = "XY " ;
            HSTDV:units = "whoknows" ;
            HSTDV:description = "something" ;
            HSTDV:stagger = "M" ;
            HSTDV:sr_x = 1 ;
            HSTDV:sr_y = 1 ;
      float HCNVX(Time, south_north, west_east) ;
            HCNVX:FieldType = 104 ;
            HCNVX:MemoryOrder = "XY " ;
            HCNVX:units = "whoknows" ;
            HCNVX:description = "something" ;
            HCNVX:stagger = "M" ;
            HCNVX:sr_x = 1 ;
            HCNVX:sr_y = 1 ;
      float LU_INDEX(Time, south_north, west_east) ;
            LU_INDEX:FieldType = 104 ;
            LU_INDEX:MemoryOrder = "XY " ;
            LU_INDEX:units = "category" ;
            LU_INDEX:description = "Dominant category" ;
            LU_INDEX:stagger = "M" ;
            LU_INDEX:sr_x = 1 ;
            LU_INDEX:sr_y = 1 ;
      float LANDUSEF(Time, z-dimension0024, south_north, west_east)
;
            LANDUSEF:FieldType = 104 ;
            LANDUSEF:MemoryOrder = "XYZ" ;
            LANDUSEF:units = "category" ;
            LANDUSEF:description = "24-category USGS landuse" ;
            LANDUSEF:stagger = "M" ;
            LANDUSEF:sr_x = 1 ;
            LANDUSEF:sr_y = 1 ;
      float LANDMASK(Time, south_north, west_east) ;
            LANDMASK:FieldType = 104 ;
            LANDMASK:MemoryOrder = "XY " ;
            LANDMASK:units = "none" ;
            LANDMASK:description = "Landmask : 1=land, 0=water" ;
            LANDMASK:stagger = "M" ;
            LANDMASK:sr_x = 1 ;
            LANDMASK:sr_y = 1 ;
      float F(Time, south_north, west_east) ;
            F:FieldType = 104 ;
            F:MemoryOrder = "XY " ;
            F:units = "-" ;
            F:description = "Coriolis F parameter" ;
            F:stagger = "M" ;
            F:sr_x = 1 ;
            F:sr_y = 1 ;
      float E(Time, south_north, west_east) ;
```

```
            E:FieldType = 104 ;
            E:MemoryOrder = "XY " ;
            E:units = "-" ;
            E:description = "Coriolis E parameter" ;
            E:stagger = "M" ;
            E:sr_x = 1 ;
            E:sr_y = 1 ;
      float XLONG_V(Time, south_north, west_east) ;
            XLONG_V:FieldType = 104 ;
            XLONG_V:MemoryOrder = "XY " ;
            XLONG_V:units = "degrees longitude" ;
            XLONG_V:description = "Longitude on velocity grid" ;
            XLONG_V:stagger = "V" ;
            XLONG_V:sr_x = 1 ;
            XLONG_V:sr_y = 1 ;
      float XLAT_V(Time, south_north, west_east) ;
            XLAT_V:FieldType = 104 ;
            XLAT_V:MemoryOrder = "XY " ;
            XLAT_V:units = "degrees latitude" ;
            XLAT_V:description = "Latitude on velocity grid" ;
            XLAT_V:stagger = "V" ;
            XLAT_V:sr_x = 1 ;
            XLAT_V:sr_y = 1 ;
      float XLONG_M(Time, south_north, west_east) ;
            XLONG_M:FieldType = 104 ;
            XLONG_M:MemoryOrder = "XY " ;
            XLONG_M:units = "degrees longitude" ;
            XLONG_M:description = "Longitude on mass grid" ;
            XLONG_M:stagger = "M" ;
            XLONG_M:sr_x = 1 ;
            XLONG_M:sr_y = 1 ;
      float XLAT_M(Time, south_north, west_east) ;
            XLAT_M:FieldType = 104 ;
            XLAT_M:MemoryOrder = "XY " ;
            XLAT_M:units = "degrees latitude" ;
            XLAT_M:description = "Latitude on mass grid" ;
            XLAT_M:stagger = "M" ;
            XLAT_M:sr_x = 1 ;
            XLAT_M:sr_y = 1 ;

// global attributes:
            :TITLE = "OUTPUT FROM METGRID V3.4" ;
            :SIMULATION_START_DATE = "2008-01-11_00:00:00" ;
            :WEST-EAST_GRID_DIMENSION = 19 ;
            :SOUTH-NORTH_GRID_DIMENSION = 39 ;
            :BOTTOM-TOP_GRID_DIMENSION = 27 ;
            :WEST-EAST_PATCH_START_UNSTAG = 1 ;
            :WEST-EAST_PATCH_END_UNSTAG = 19 ;
            :WEST-EAST_PATCH_START_STAG = 1 ;
            :WEST-EAST_PATCH_END_STAG = 19 ;
            :SOUTH-NORTH_PATCH_START_UNSTAG = 1 ;
            :SOUTH-NORTH_PATCH_END_UNSTAG = 39 ;
            :SOUTH-NORTH_PATCH_START_STAG = 1 ;
            :SOUTH-NORTH_PATCH_END_STAG = 39 ;
```

```
            :GRIDTYPE = "E" ;
            :DX = 0.289143f ;
            :DY = 0.287764f ;
            :DYN_OPT = 4 ;
            :CEN_LAT = 32.f ;
            :CEN_LON = -83.f ;
            :TRUELAT1 = 1.e+20f ;
            :TRUELAT2 = 1.e+20f ;
            :MOAD_CEN_LAT = 0.f ;
            :STAND_LON = 1.e+20f ;
            :POLE_LAT = 90.f ;
            :POLE_LON = 0.f ;
            :corner_lats = 26.39329f, 37.31068f, 37.31068f,
26.39329f, 26.40831f, 37.3276f, 37.29281f, 26.37742f, 0.f, 0.f, 0.f,
0.f, 0.f, 0.f, 0.f, 0.f ;
            :corner_lons = -88.78565f, -89.519f, -76.481f, -
77.21435f, -88.46474f, -89.1577f, -76.11986f, -76.89354f, 0.f, 0.f,
0.f, 0.f, 0.f, 0.f, 0.f, 0.f ;
            :MAP_PROJ = 203 ;
            :MMINLU = "USGS" ;
            :NUM_LAND_CAT = 24 ;
            :ISWATER = 16 ;
            :ISLAKE = -1 ;
            :ISICE = 24 ;
            :ISURBAN = 1 ;
            :ISOILWATER = 14 ;
            :grid_id = 1 ;
            :parent_id = 1 ;
            :i_parent_start = 0 ;
            :j_parent_start = 0 ;
            :i_parent_end = 19 ;
            :j_parent_end = 39 ;
            :parent_grid_ratio = 1 ;
            :sr_x = 1 ;
            :sr_y = 1 ;
            :NUM_METGRID_SOIL_LEVELS = 4 ;
            :FLAG_METGRID = 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_SOILHGT = 1 ;
}
```