Advanced Usage of the WRF Preprocessing System

Michael Duda

*NCAR is sponsored by the
National Science Foundation

Outline

- The GEOGRID.TBL file
  - What is the GEOGRID.TBL file?
  - Ingesting new static fields
  - Examples: Using high-resolution land use and topography data
- The METGRID.TBL file
  - What is the METGRID.TBL file?
  - Example: Defining interpolation options for a new field
  - Example: Using the METGRID.TBL file for a real-time system
- Utility programs example: fixing "hot lakes"

The GEOGRID.TBL File

- GEOGRID.TBL is the file that determines which fields are interpolated by geogrid *at runtime*
  - Each entry in GEOGRID.TBL corresponds to one field to be produced by geogrid
  - When new data sources are involved, or when the default treatment of fields is inadequate, user may want/need to edit GEOGRID.TBL
  - However, default GEOGRID.TBL is sufficient to initialize a WRF simulation

The GEOGRID.TBL File

- Format of GEOGRID.TBL file is simple text, with specifications of the form *keyword*=*value*
- Example entry for a 30" landuse data set:

```
===============================
name=LANDUSEF   # Houston, TX urban data
       priority   = 2
       dest_type  = categorical
       z_dim_name = land_cat
       interp_option = 30s:nearest_neighbor
       abs_path = 30s:/users/duda/Houston/
===============================
```

For a complete list of possible keywords *See p. 3-46*

The GEOGRID.TBL File

- Using the GEOGRID.TBL, we can
  - Change the method(s) used to interpolate a field
  - Apply smoothing filters to continuous fields
  - Derive fields from others
    - E.g., dominant category or slope fields
  - *Add new data for geogrid to interpolate*

New Fields in GEOGRID.TBL

There are three basic types of new data to be added through the GEOGRID.TBL file:

1) Completely new fields
   - fields that were previously not processed by geogrid

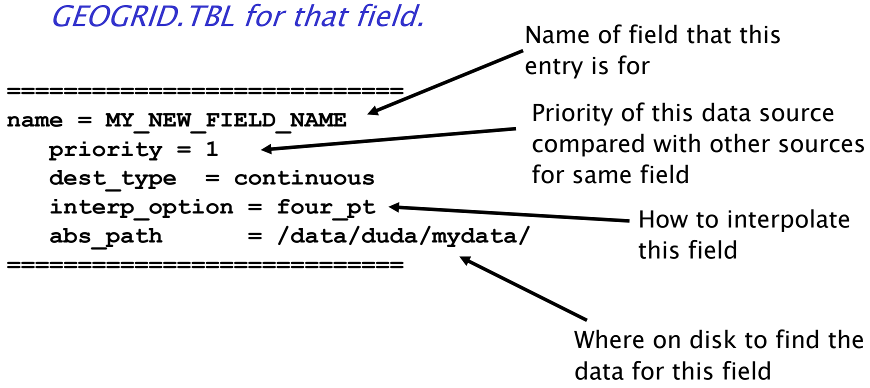
2) Different resolution data sets for an existing field
   - Such sources *do not need to be supplemented* by existing data
   - E.g., Adding a 90-meter resolution topography data set

3) Alternative sources for a field that *must be used in addition to an existing source*
   - E.g., A new soil category data set exists, but covers only South Korea

1) Completely new fields

Completely new fields:

For a new field, simply add an entry in GEOGRID.TBL for that field.



```
============================
name = MY_NEW_FIELD_NAME
   priority = 1
   dest_type  = continuous
   interp_option = four_pt
   abs_path      = /data/duda/mydata/
============================
```

Name of field that this entry is for

Priority of this data source compared with other sources for same field

How to interpolate this field

Where on disk to find the data for this field

See p. 3-46

2) Different resolution data set

Different resolution data sets for an existing field:

Specify the path to the new data set and which interpolation methods should be used for the new resolution in the existing entry for that field.

```
===============================
name = HGT_M
     priority = 1
     dest_type = continuous
     smooth_option = smth-desmth
     interp_option =         30s:special(4.0)+four_pt
     interp_option =      my_res:four_pt
     interp_option =     default:four_pt
     rel_path=      30s:topo_30s/
     rel_path=   my_res:new_topo_directory/
     rel_path=  default:topo_2m/
===============================
```

3) Alternative data sources

Alternative sources for a field that must be used in addition to an existing source :

Add a new entry for the field that has the same name as the field's existing entry, but make priority of new entry higher.

```
===============================
name = HGT_M
      priority = 2
      dest_type = continuous
      interp_option = default:four_pt
      rel_path      = default:some_path/
===============================
name = HGT_M
     priority = 1
     dest_type = continuous
     interp_option = default:four_pt
     rel_path      = default:topo_2m/
===============================
```

Preparing new geogrid data sets

To add a new data source, we need to

1) Write the data in the proper binary format

– See Chapter 3: "Writing Static Data to the Geogrid Binary Format"

– Can make use of read_geogrid.c and write_geogrid.c

2) Create an "index" metadata file for the data set

– This tells geogrid about the projection, coverage, resolution, type, and storage representation of the data set

3) Add/edit entry for the data in the GEOGRID.TBL file

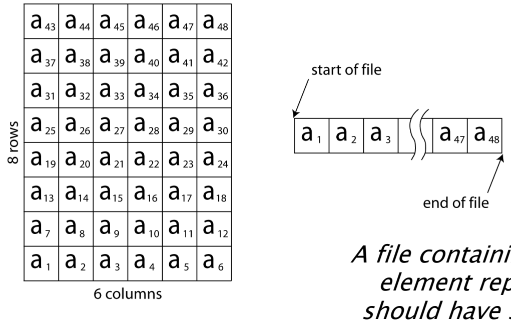
– The change to GEOGRID.TBL will follow one of the three cases mentioned before

The Geogrid Data Format

The geogrid format is a simple binary raster

- Elements of a rectangular array of data are written, row by row, to a file
- No record markers or any type of metadata are written to this file

See p. 3-37



a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}
a_{37}	a_{38}	a_{39}	a_{40}	a_{41}	a_{42}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}
a_{25}	a_{26}	a_{27}	a_{28}	a_{29}	a_{30}
a_{19}	a_{20}	a_{21}	a_{22}	a_{23}	a_{24}
a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}
a_{7}	a_{8}	a_{9}	a_{10}	a_{11}	a_{12}
a_{1}	a_{2}	a_{3}	a_{4}	a_{5}	a_{6}

*A file containing a N×M array, with each element represented using K bytes, should have size exactly N*M*K bytes!*

The Geogrid Data Format

Since the contents of the file contain only the values from the array, *care must be taken if using Fortran to write the array*

– Fortran unformatted writes add *record markers* to the beginning and end of each record

– So, rather than $X_1X_2X_3\ldots X_{n-1}X_n$ we get $RX_1X_2X_3\ldots X_{n-1}X_nR$, where R is a record marker

Instead of Fortran, the C routines `read_geogrid.c` and `write_geogrid.c` may be used to read and write binary files

– these may be called from either Fortran or C

The Geogrid Data Format

From python, one can use

```
numpy.fromfile(file, dtype=dt)
```

to read the geogrid binary files, and

```
numpy.ndarray.tofile(file)
```

to write the geogrid binary files.

The `dtype` argument and `numpy.ndarray.astype` may be used to match the *wordsize* and *endianness* used in the binary file!

- Values are always represented as integers

The Geogrid Data Format

The filenames of geogrid binary files should have the form:

xxxxx-XXXXX.yyyyy-YYYYY

where

xxxxx	is the starting x-index
XXXXX	is the ending x-index
yyyyy	is the starting y-index
YYYYY	is the ending y-index

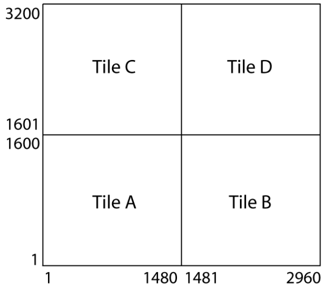
E.g., For a binary file containing an array with 500 columns and 750 rows, the file name would be

00001-00500.00001-00750

The Geogrid Data Format

If the data are not available in a single tile (array), multiple files may be used to store the data

- All tiles must have the same x-dimension
- All tiles must have the same y-dimension
- If necessary, a tile can be “padded” with missing values to expand it to the same size as other tiles in the data set

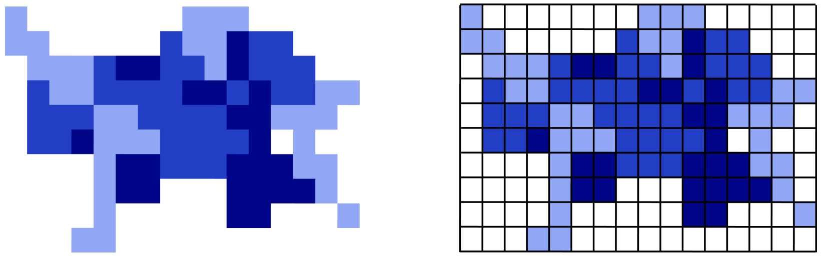


Tile A named	00001-01480.00001-01600
Tile B named	01481-02960.00001-01600
Tile C named	00001-01480.01601-03200
Tile D named	01481-02960.01601-03200

The Geogrid Data Format

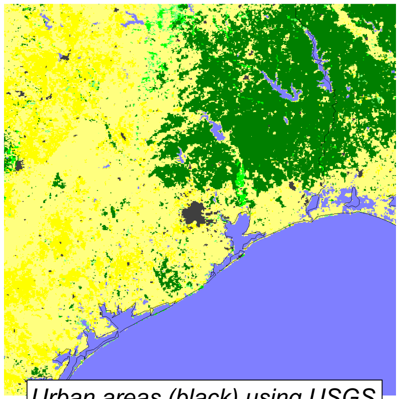
If the data do not cover a rectangular region, areas with no data are simply filled with a missing value so that the overall data set is rectangular

- The particular missing value used in the data set is specified in the `index` metadata file for the data set

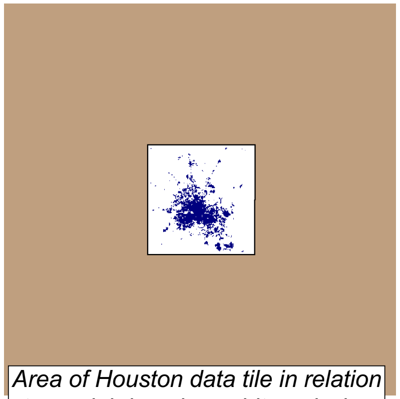


Example: Houston LU Data Set

- Given dataset for new Houston urban land use categories
  - Regular lat/lon projection, 30” resolution; categories 31, 32 & 33



Urban areas (black) using USGS 24-category data set



Area of Houston data tile in relation to model domain – white=missing data and blue=valid data

Example: Houston LU Data Set

To make use of the new data, we do the following:

1) Write the data to the binary format used by geogrid

2) Create an index file for the data

```
type=categorical
category_min=31; category_max=33
projection=regular_ll
dx=0.00833333; dy=0.00833333
known_x=1.0;   known_y=1.0
known_lat=29.3375
known_lon=-95.9958333
wordsize=1
tile_x=157; tile_y=143; tile_z=1
missing_value = 0.
units="category"
description="3-category urban LU"
```

Data set has categories 31 through 33

30 arc second resolution

Geographic location of data set

Treat 0 as “no data”

See p. 3-49

Example: Houston LU Data Set

3) Define an entry for the data in GEOGRID.TBL

```
===============================
name=LANDUSEF
        priority   = 2
        dest_type  = categorical
        z_dim_name = land_cat
        interp_option = default:nearest_neighbor
        abs_path  = default:/users/duda/Houston/
===============================
```

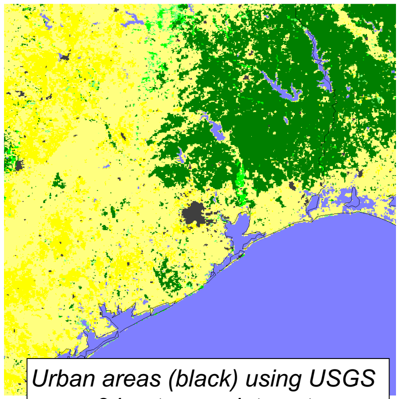
Give this data source priority over default data sources

How to interpolate this data source, and where to find it on disk

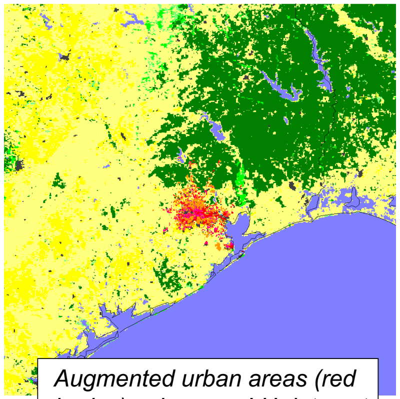
Example: Houston LU Data Set

4) Run geogrid.exe

Any gridpoints covered by Houston data will use it; otherwise default USGS data will be used



Urban areas (black) using USGS 24-category data set



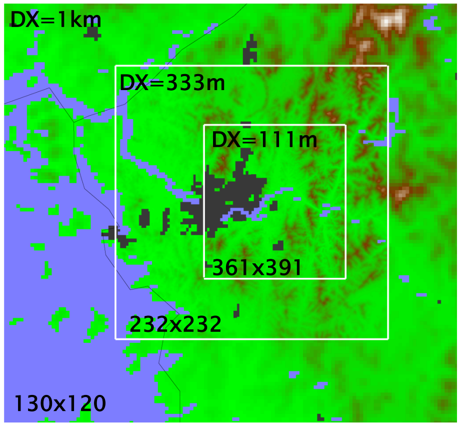
Augmented urban areas (red shades) using new LU data set

Example: South Korea

Shuttle Radar Topography Mission (SRTM) 3 arc second topography data

We would like to use the SRTM data, especially for domains 2 and 3.

Follow steps for adding a new resolution for an existing data set (case 2)



Example: Seoul

To use the SRTM topography data, we

1) Write data to geogrid binary format

2) Create an index file for the data set

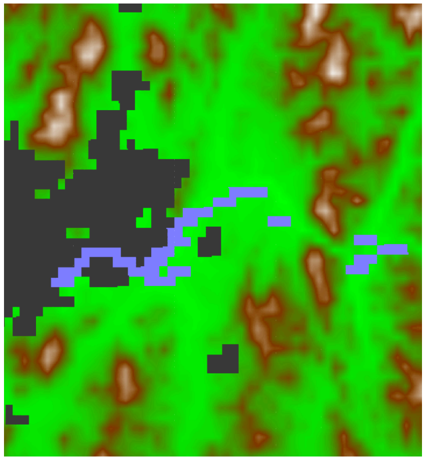
3) Modify the GEOGRID.TBL entries for HGT_M, HGT_U, and HGT_V

```
===============================
name = HGT_M
     priority = 1
     dest_type = continuous
     interp_option = default:average_gcell(4.0)+four_pt
     interp_option = SRTM:four_pt
     rel_path = default:gmted2010_30s/
     rel_path = SRTM:SRTM/
===============================
```

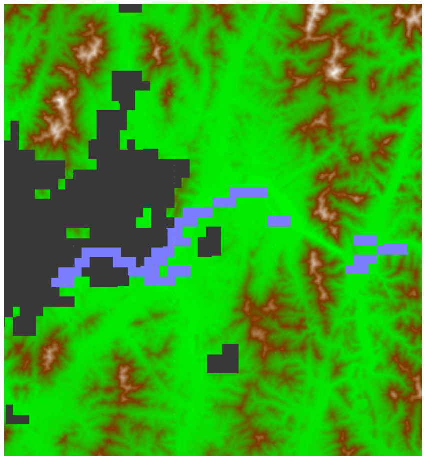
4) Specify that we should interpolate from SRTM in namelist by setting

```
geog_data_res = 'default','SRTM+default','SRTM+default'
```

Example: Seoul



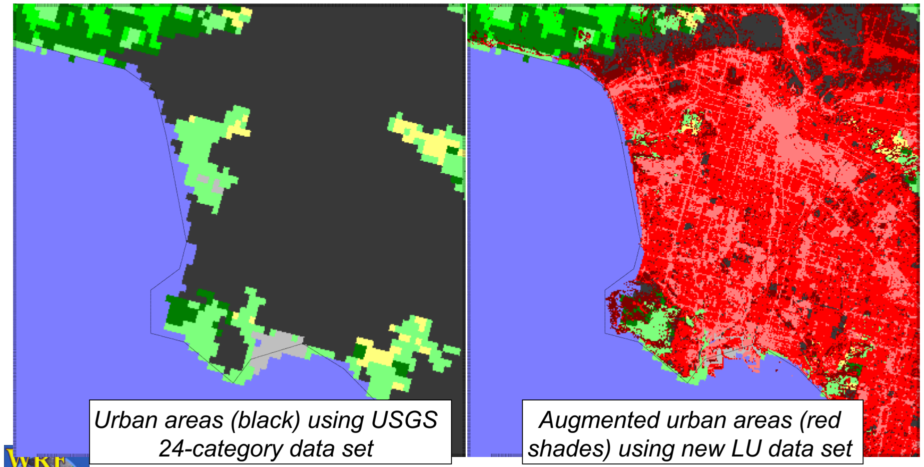
Domain 3 (DX=111m) using default 30" GMTED2010 topography



Domain 3 (DX=111m) using 3" SRTM topography

Another Example: Los Angeles

For Los Angeles, we have a 30-meter resolution, 3 urban land use category data set



Urban areas (black) using USGS 24-category data set

Augmented urban areas (red shades) using new LU data set

Outline

- The GEOGRID.TBL file
  - What is the GEOGRID.TBL file?
  - Ingesting new static fields
  - Examples: Using high-resolution land use and topography data
- The METGRID.TBL file
  - What is the METGRID.TBL file?
  - Example: Defining interpolation options for a new field
  - Example: Using the METGRID.TBL file for a real-time system
- Utility programs example: fixing “hot lakes”

The METGRID.TBL File

The METGRID.TBL file controls how meteorological fields are interpolated

- Unlike GEOGRID.TBL, METGRID.TBL *does not determine which fields will be processed*, only *how to process them* if they are encountered
- Every field in intermediate files will be interpolated
  - If no entry in METGRID.TBL for a field, a default interpolation scheme (nearest neighbor) will be used
  - It is possible to specify in METGRID.TBL that a field should be discarded

The METGRID.TBL File

- Suitable entries in METGRID.TBL are provided for common fields
  - *Thus, many users will rarely need to edit METGRID.TBL*
- When necessary, different interpolation methods (and other options) can be set in METGRID.TBL
  - Interpolation options can depend on the source of a field

The METGRID.TBL File

- Example METGRID.TBL entry (for “soil moisture 0-10 cm”)

```
========================================
name=SM000010
interp_option=sixteen_pt+four_pt+average_4pt
masked=water
interp_mask=LANDSEA(0)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

Example: A new METGRID.TBL entry

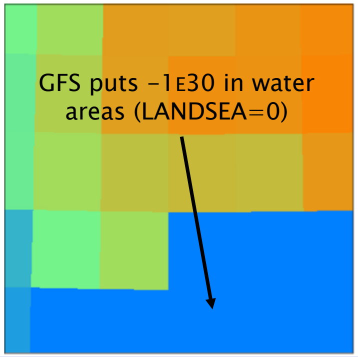
- Suppose we have a 1000x1000 domain over Houston (dx=500 m)
  - This is the same domain as in the urban land use example
- Meteorological data come from 1-degree GFS
  - *Note that we will be interpolating 1-degree data onto a 500-m grid!*
- We want to create an entry for a new soil moisture field, SM000010

Example: A new METGRID.TBL entry

- Initially, we run metgrid.exe and get the message:

```
INFORM: Entry in METGRID.TBL not found for field SM000010.
  Default options will be used for this field!
```

- The resulting SM000010 field looks very coarse
- We need to create a METGRID.TBL entry so metgrid will know how to interpolate this field!



GFS puts -1E30 in water areas (LANDSEA=0)

Example: A new METGRID.TBL entry

- We add an initial entry in METGRID.TBL for SM000010:

```
===========================
name = SM000010
masked = water
interp_mask = LANDSEA(0)
interp_option = sixteen_pt + nearest_neighbor
fill_missing = 0.
===========================
```

Specify that the field should *not* be interpolated to model water points

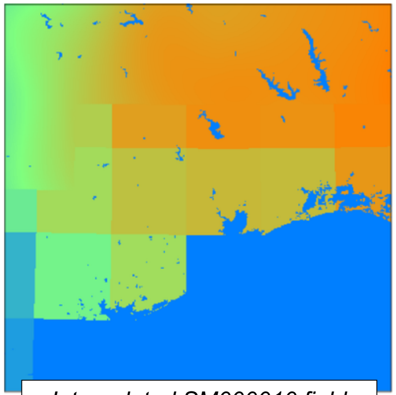
Specify that metgrid should not use points in source where LANDSEA field equals 0

Fill model points that don't receive an interpolated value (like water) to 0

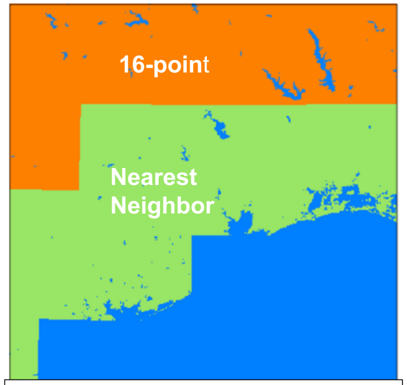
For a complete list of possible keywords *See p. 3-52*

Example: A new METGRID.TBL entry

- Now, after running metgrid.exe again, the SM000010 field looks like



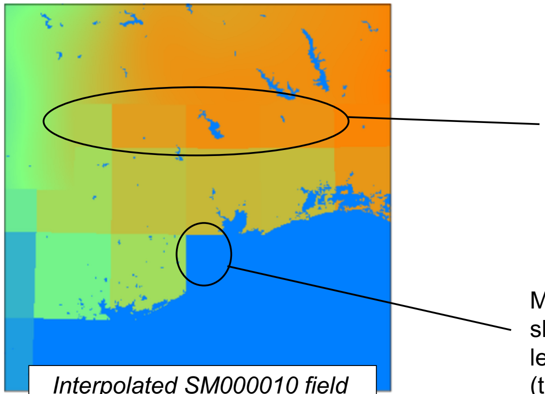
Interpolated SM000010 field (sixteen_pt + nearest_neighbor)



Which interpolator was used at each model grid point

Example: A new METGRID.TBL entry

- But, the interpolated field still looks bad near the coastline



Should be sufficient data to use 4-point interpolation in these areas

Model grid points here should be adjacent to at least one valid GFS point (though not nearest)

Interpolated SM000010 field (sixteen_pt + nearest_neighbor)

Example: A new METGRID.TBL entry

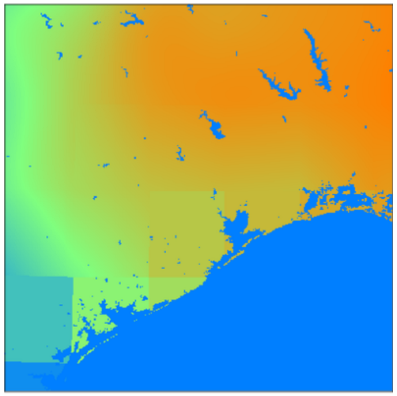
- Update the METGRID.TBL entry for SM000010

```
===========================
name = SM000010
masked = water
interp_mask = LANDSEA(0)
interp_option = sixteen_pt + four_pt + average_4pt
fill_missing = 0.
===========================
```

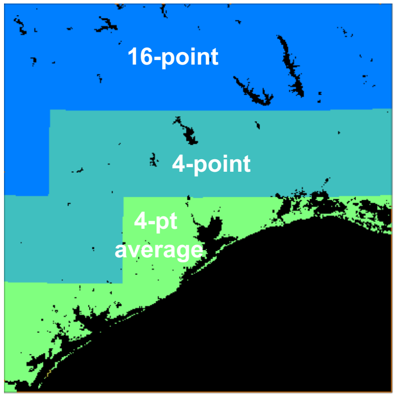
- If 16-pt doesn't work, then try 4-pt before reverting to a 4-point average
  - Note that 4-point average will work anywhere nearest_neighbor would (missing/masked values not counted in the average)

Example: A new METGRID.TBL entry

- The resulting field, below-left:



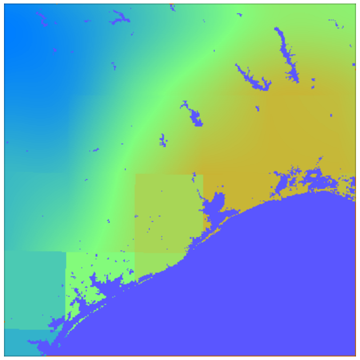
Interpolated SM000010 field (sixteen_pt + four_pt + average_4pt)



Which interpolator was used at each model grid point

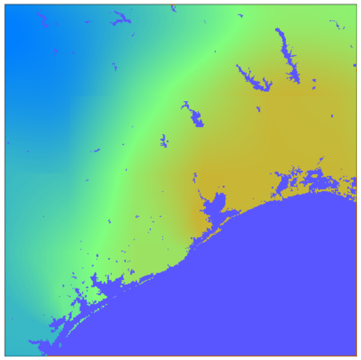
Example: A new METGRID.TBL entry

- By using **`wt_average_4pt`** instead of **`average_4pt`**:



*sixteen_pt + four_pt + **average_4pt***

→



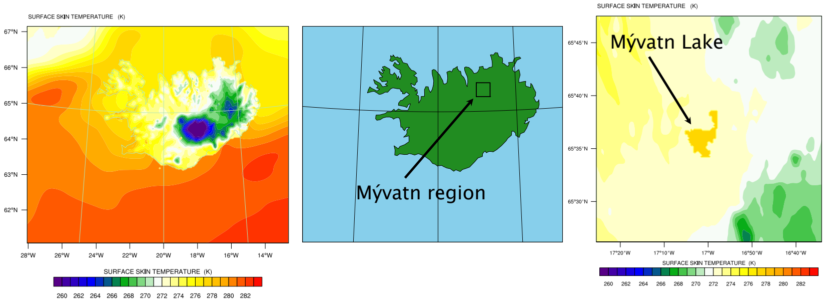
*sixteen_pt + four_pt + **wt_average_4pt***

Outline

- The GEOGRID.TBL file
  - What is the GEOGRID.TBL file?
  - Ingesting new static fields
  - Examples: Using high-resolution land use and topography data
- The METGRID.TBL file
  - What is the METGRID.TBL file?
  - Example: Defining interpolation options for a new field
  - Example: Using the METGRID.TBL file for a real-time system
- Utility programs example: fixing “hot lakes”

Motivating Problem

The “Hot Lake” problem: Inland water bodies that are not resolved by SST data sets often receive extrapolated values from nearby oceans or other resolved water bodies.



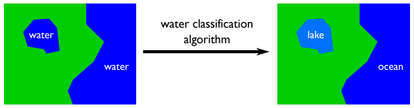
Above left: Skin temperature field (TSK) for Iceland and surrounding ocean on 26 January 2011 1200 UTC from NCEP GFS and RTG SST data.

Above right: TSK in the Mývatn region. SST for Mývatn Lake is ~277 K!

Approach

In WRF v3.3 and later, let the *real* preprocessor know which water points are inland water bodies, and provide it a more accurate estimate of SST to be used only over these water bodies.

1) Identify inland water bodies in the land cover data set



1) Provide a suitable proxy for SST field over inland water bodies
   - E.g., Average surface air temperature for X days prior, 273 K for frozen lakes, etc.
2) Modify the SST field in the WRF input file
   - Use new capability in v3.3 real.exe program

Identifying Lakes

Some data sets already identify lakes with separate categories

- MODIS, CORINE

For others, we need a way to do this

- Should be automated
  - don’t want to spend long hours clicking on pixels for each data set
- Should be tunable
  - what constitutes a lake will naturally depend on what our SST data set is able to resolve
- Ideally, would not require auxiliary data

This is the default as of WPS v3.9

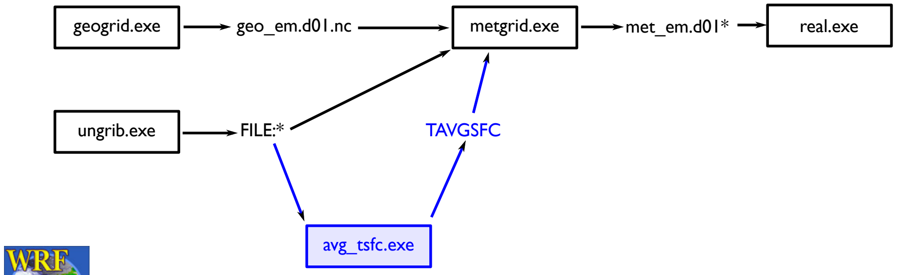
In *namelist.wps*, set:

- `geog_data_res = “usgs_lakes+default”` for USGS land use (16=ocean, 28=lake)
- `geog_data_res = “modis_30s_lake+default”` for MODIS land use (17=ocean, 21=lake)

Creating a Proxy SST Field

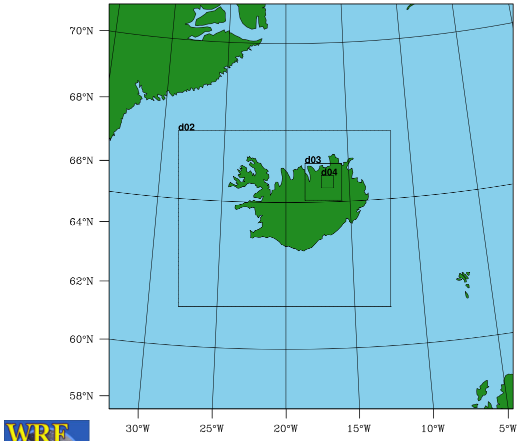
The *avg_tsfc.exe* utility program may be used to compute the average 2-m air temperature field for any number of full diurnal cycles

- Number of cycles determined by available intermediate files and date range in namelist
- The resulting TAVGSFC intermediate file may be provided to the metgrid program



Test case: Lake Mývatn

To confirm that everything is working as expected, try correcting the temperature for Lake Mývatn in the winter



Grid ID	Resolution	Size
1	16 km	99x99
2	4 km	208x172
3	1 km	136x128
4	250 m	160x160

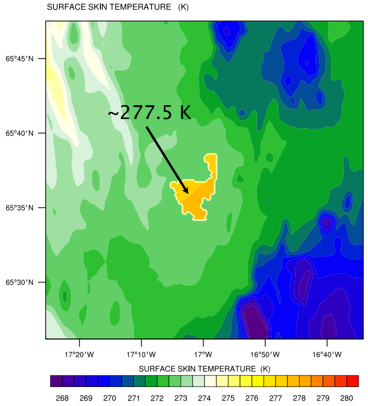
ICs + BCs from NCEP GFS

Sea surface temperatures from RTG SST

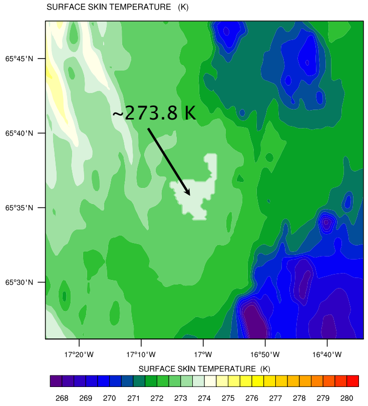
Initial time: 26 January 2011, 1200 UTC

Test case: Lake Mývatn

26 January 2011, 12 UTC



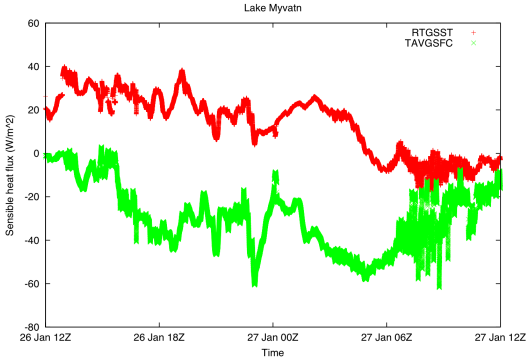
Initial skin temperature field using extrapolated SST values



Initial skin temperature field using previous 5-day average of 2-m air temperature for lake SST

Test case: Lake Mývatn

Time series of sensible heat flux in the center of the lake show a significant decrease when using a more realistic SST (TAVGSFC)



Latent heat flux time series from simulation using TAVGSFC for SST also shows a decrease from RTG SST time series as well

Summary

- In this lecture, we've seen
  - What the GEOGRID.TBL and METGRID.TBL files do
  - How to use new geographical data sources in the WPS
    - High-resolution land use and topography data
  - How to use the METGRID.TBL file to correct interpolation-related problems
  - How utility programs can be used to improve simulations
- For other features of the WPS, see Chapter 3 of the User's Guide
- For more information about using high-resolution topography data or urban land use data (over the U.S.), see http://www2.mmm.ucar.edu/people/duda/files/how_to_hires.html

Questions?

Bonus slides: A second METGRID.TBL example

METGRID.TBL: Real-time System Example

- Suppose we have a real-time system that:
  - Uses GFS for initial and boundary conditions
  - When possible (i.e., if the files are available soon enough) uses *soil moisture* and *soil temperature* fields from AGRMET
- In our system, it may occasionally happen that the AGRMET files are not ready when we want to start our WRF run
  - Because system is real-time, we want to proceed using just the GFS land surface fields!

METGRID.TBL: Real-time System Example

- We already know how to run ungrib on multiple sources of data to get

  GFS:YYYY-MM-DD_HH

  and

  AGRMET:YYYY-MM-DD_HH

  intermediate files, and specify

  fg_name = 'GFS', 'AGRMET',

  in the `&metgrid` namelist record to use both sources

See p. 3-24

METGRID.TBL: Real-time System Example

Without further changes, what happens if:

Only GFS data are available when we run metgrid

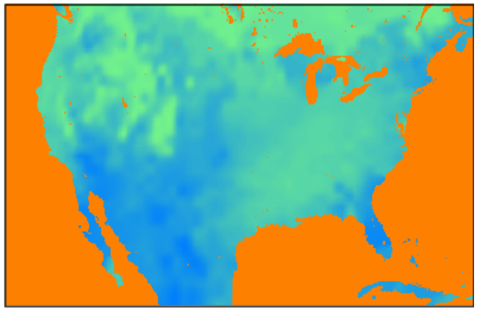
Metgrid runs and warns that no AGRMET data files were found:

```
Processing 2012-04-01_00
    GFS
    AGRMET
 WARNING: Couldn't open file AGRMET:2012-04-01_00 for
 input.
```

Metgrid will finish, but will only use GFS data!

METGRID.TBL: Real-time System Example

And the 0-10 cm soil moisture field (SM000010) looks like:

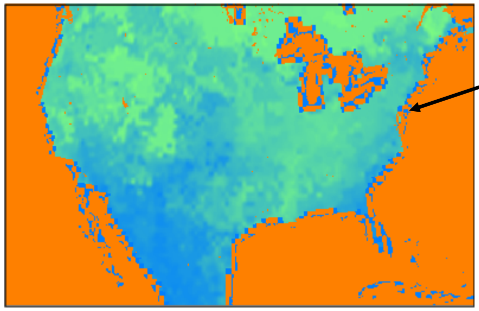


METGRID.TBL: Real-time System Example

However, what happens if:

Both GFS and AGRMET files are available when we run metgrid?

Our SM000010 field looks like:



We get unreasonable values with magnitude ~1E30 near land-water boundaries!

METGRID.TBL: Real-time System Example

Why are there bad values near coastlines? What went wrong?

In both Vtable.GFS and Vtable.AGRMET, the land-sea mask field is named LANDSEA

- In METGRID.TBL, our entry for SM000010 says:

```
========================================
name=SM000010
interp_option=sixteen_pt+four_pt+wt_average_4pt+search
masked=water
interp_mask=LANDSEA(0)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

METGRID.TBL: Real-time System Example

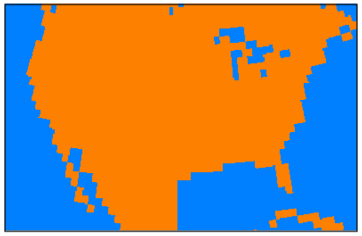
```
========================================
name=SM000010
interp_option=sixteen_pt+four_pt+wt_average_4pt+search
masked=water
interp_mask=LANDSEA(0)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

After metgrid reads in LANDSEA from GFS file *to use as an interpolation mask*, it ignored the LANDSEA field from AGRMET *for use as a mask*.

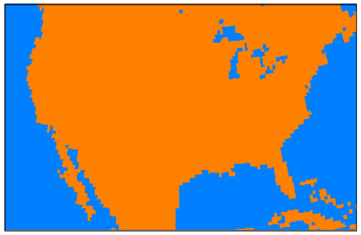
- So, metgrid used the GFS LANDSEA mask even when interpolating AGRMET data!

METGRID.TBL: Real-time System Example

When metgrid interpolated SM000010, it used the GFS landmask for a field masked by the AGRMET landmask!



GFS LANDSEA field



AGRMET LANDSEA field

Note the disagreement between the two data sources near coastlines.

METGRID.TBL: Real-time System Example

Solution:

- Rename LANDSEA to *AGR_LAND* in Vtable.AGRMET
- Rename LANDSEA to *GFS_LAND* in Vtable.GFS
- Create separate entries in METGRID.TBL

  one for GFS SM000010 field

  another for AGRMET SM000010 field

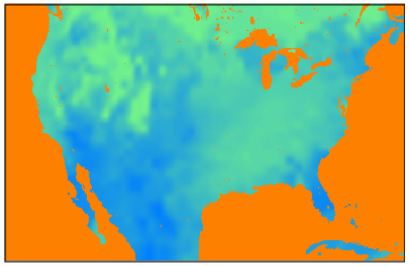
METGRID.TBL: Real-time System Example

```
========================================
name=SM000010; from_input=GFS
interp_option=sixteen_pt+four_pt+wt_average_4pt+search
masked=water
interp_mask=GFS_LAND(0)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

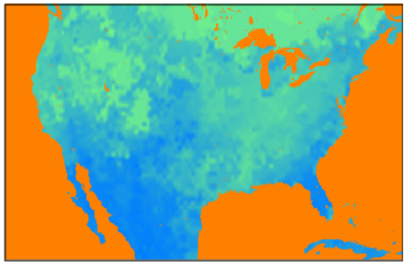
```
========================================
name=SM000010; from_input=AGRMET
interp_option=sixteen_pt+four_pt+wt_average_4pt+search
masked=water
interp_mask=AGR_LAND(-1.E30)
fill_missing=1.
flag_in_output=FLAG_SM000010
========================================
```

METGRID.TBL: Real-time System Example

With modified Vtables and METGRID.TBL:



The SM000010 field when only GFS files are available



The SM000010 field when both GFS and AGRMET files are available