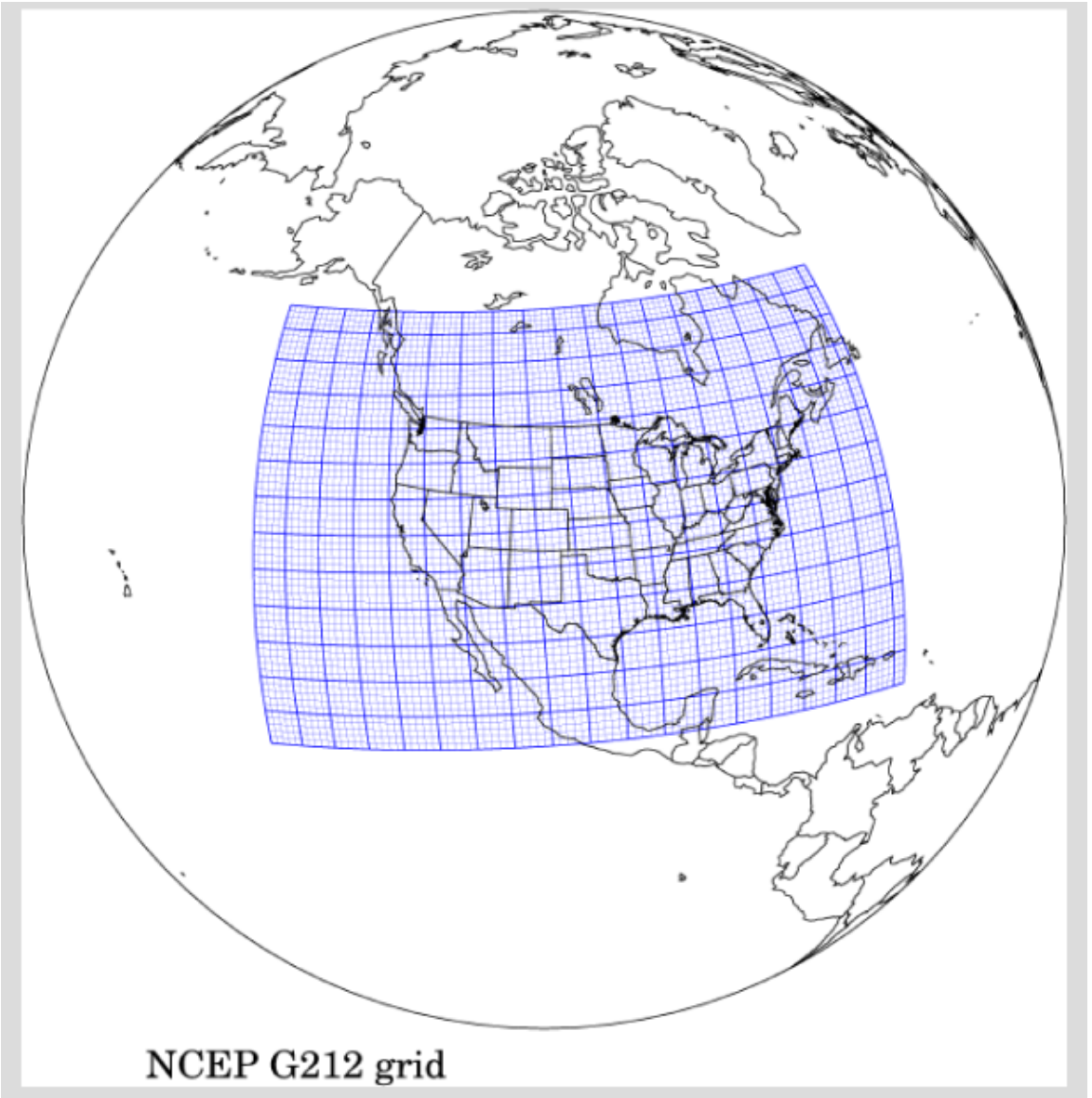
# Gridded File Formats & Python Embedding

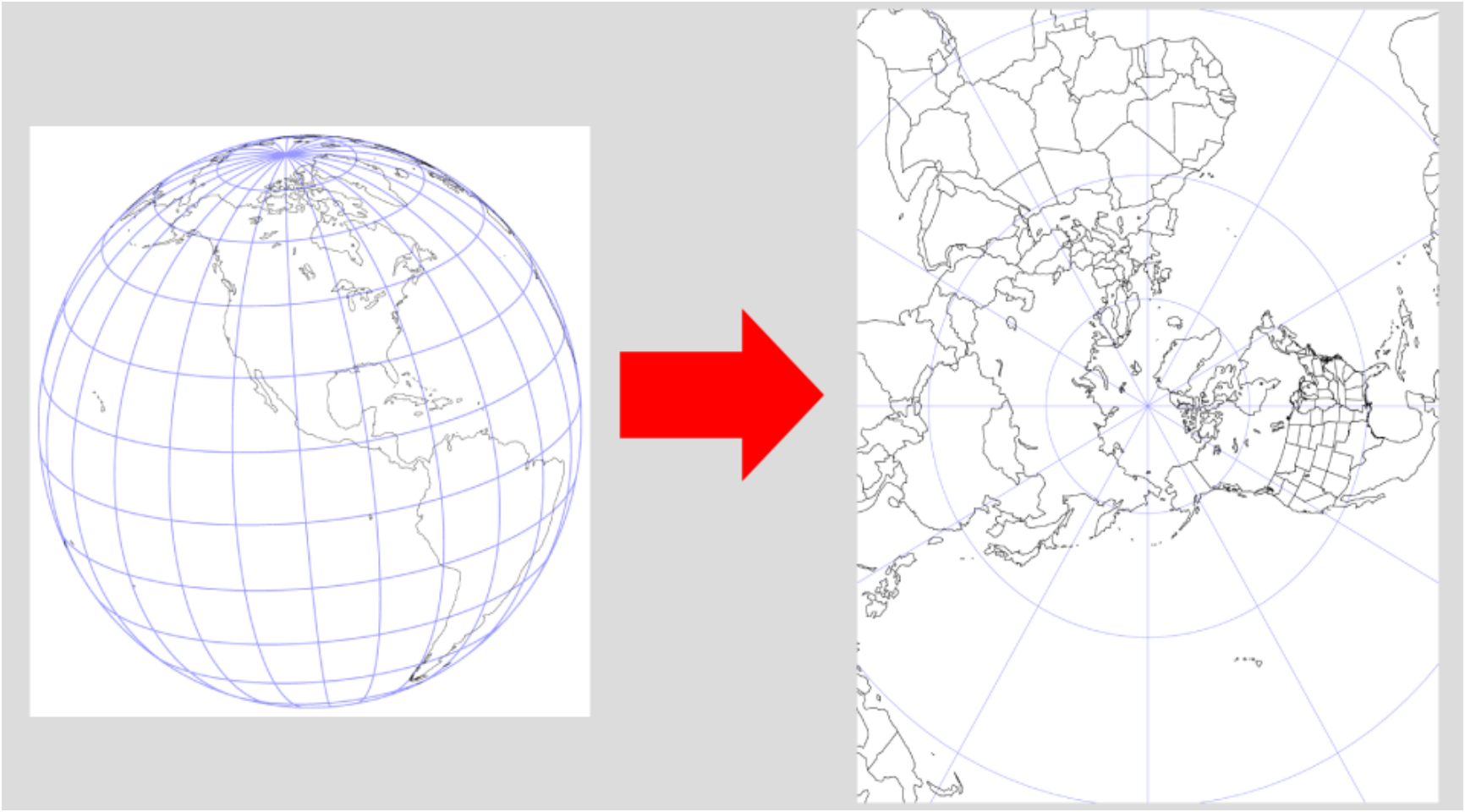
John Halley Gotway

METplus Tutorial
July 31 – August 2, 2019
NRL – Monterey, CA

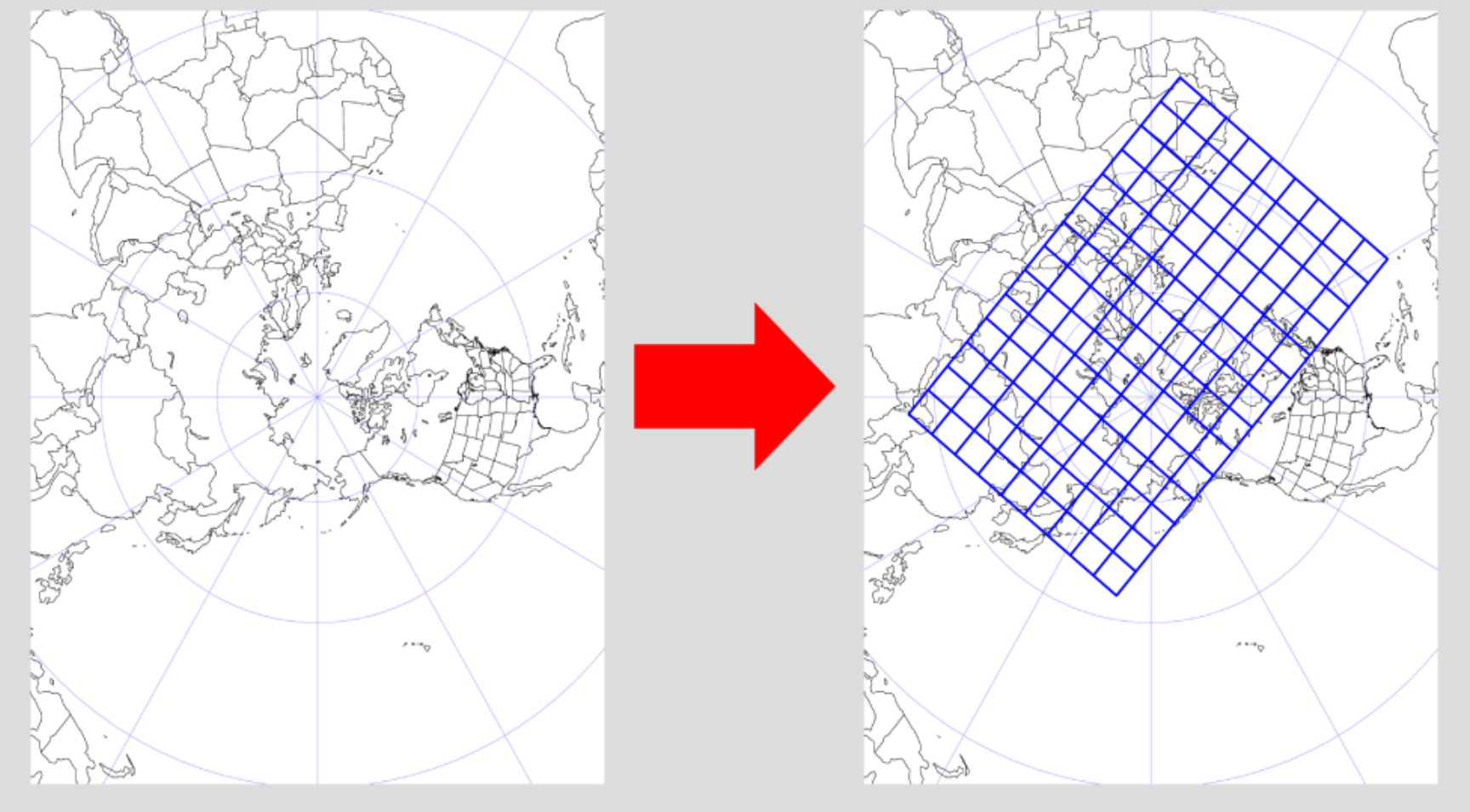
# What are Grids?



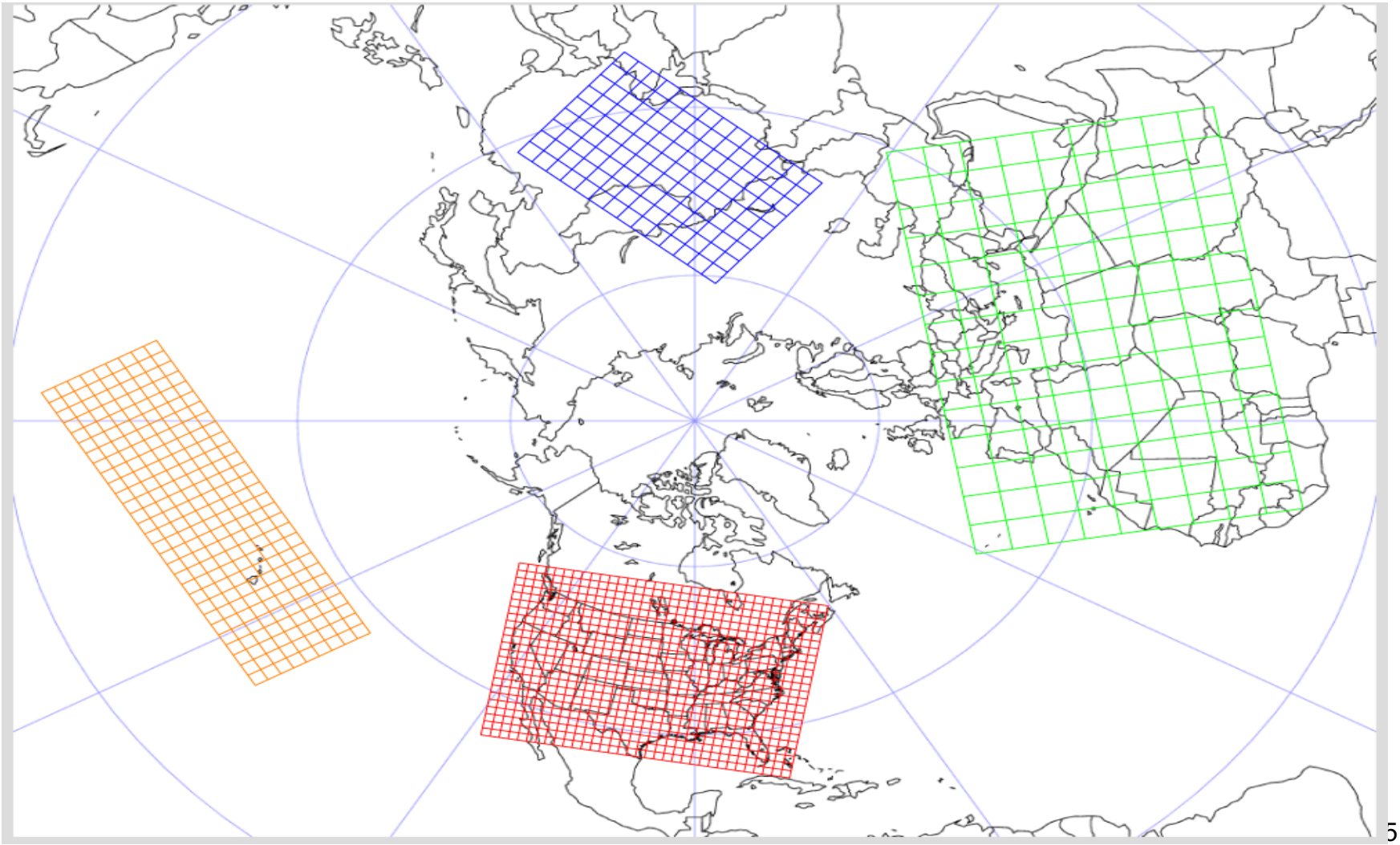
# Step #1:
# Apply Standard Map Projection



# Step #2:
# Graph Paper



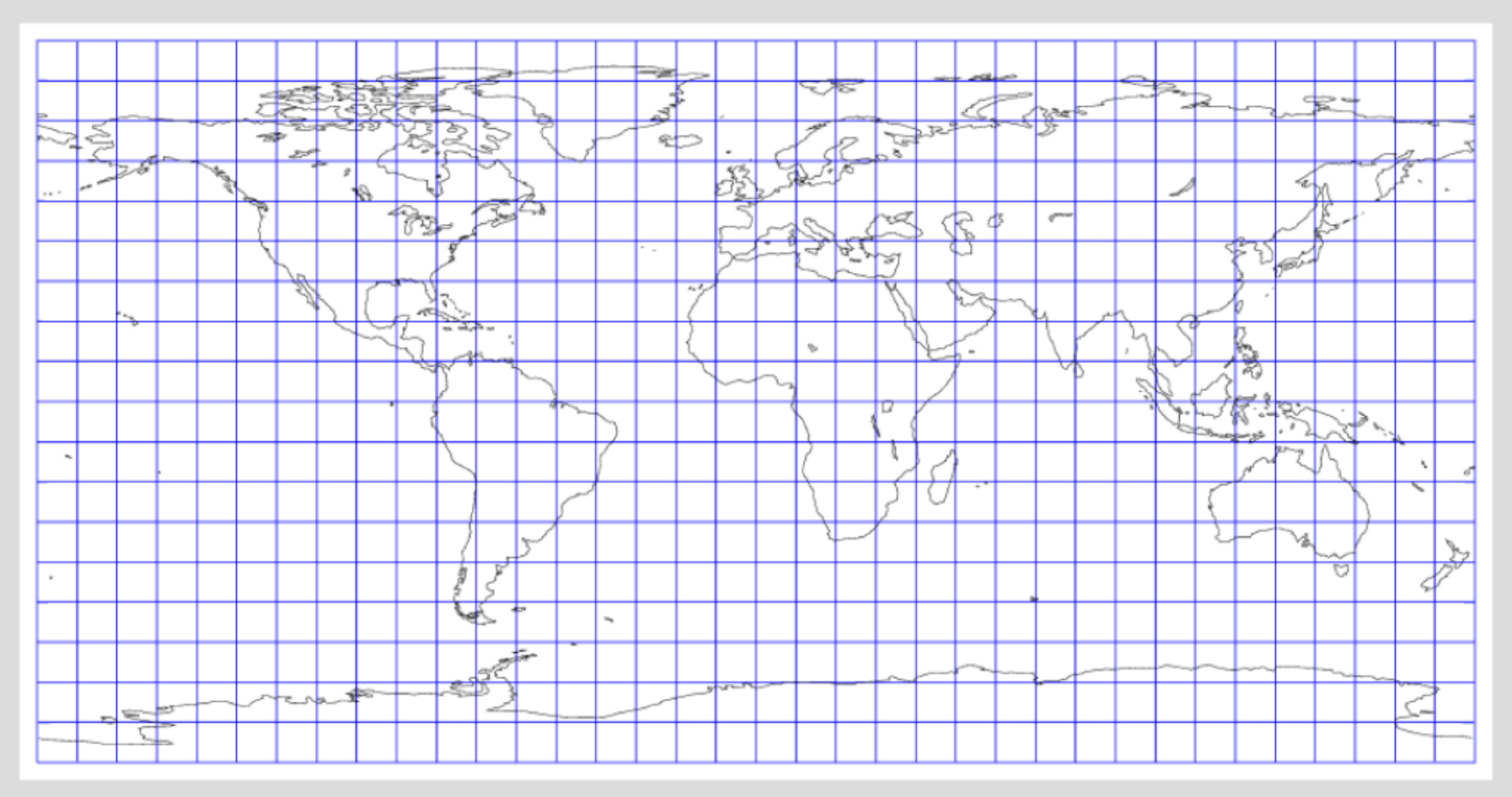
# Step #3:
# Many Options



# Supported Projections

- Mercator
  - mercator Nx Ny lat_ll lon_ll lat_ur lon_ur
- Polar Stereographic
  - stereo Nx Ny lat_ll lon_ll lon_orient D_km R_km lat_scale N|S
- Lambert Conformal
  - lambert Nx Ny lat_ll lon_ll lon_orient D_km R_km standard_parallel_1 [standard_parallel_2] N|S
- Lat/Lon
  - a.k.a: Plate Caree or Cyclindrical Equidistant
  - latlon Nx Ny lat_ll lon_ll delta_lat delta_lon
- Rotated Lat/Lon
  - rotlatlon Nx Ny lat_ll lon_ll delta_lat delta_lon true_lat true_lon aux_rot

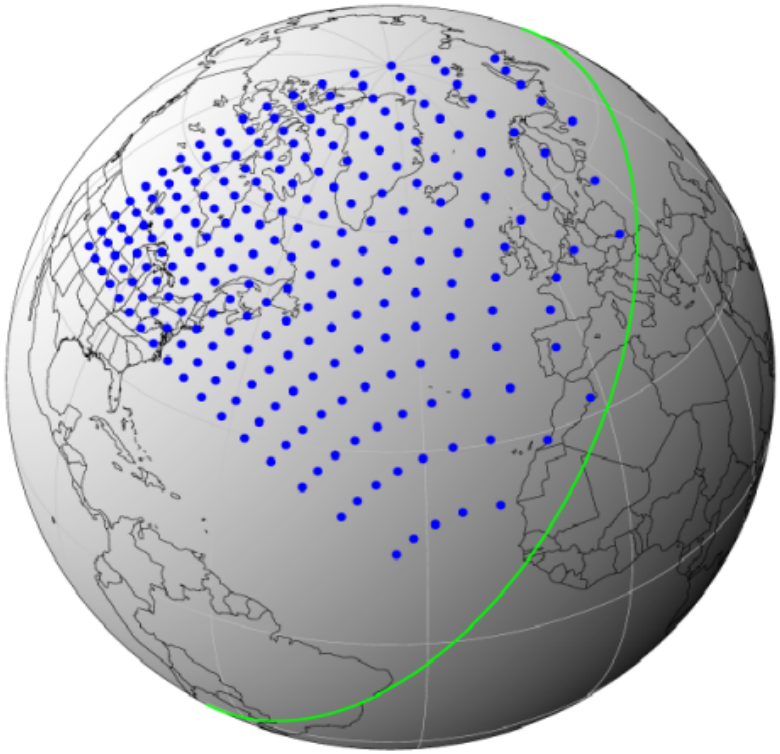
# Distortion



# Satellite Raster Geometry

$\phi = 40°$

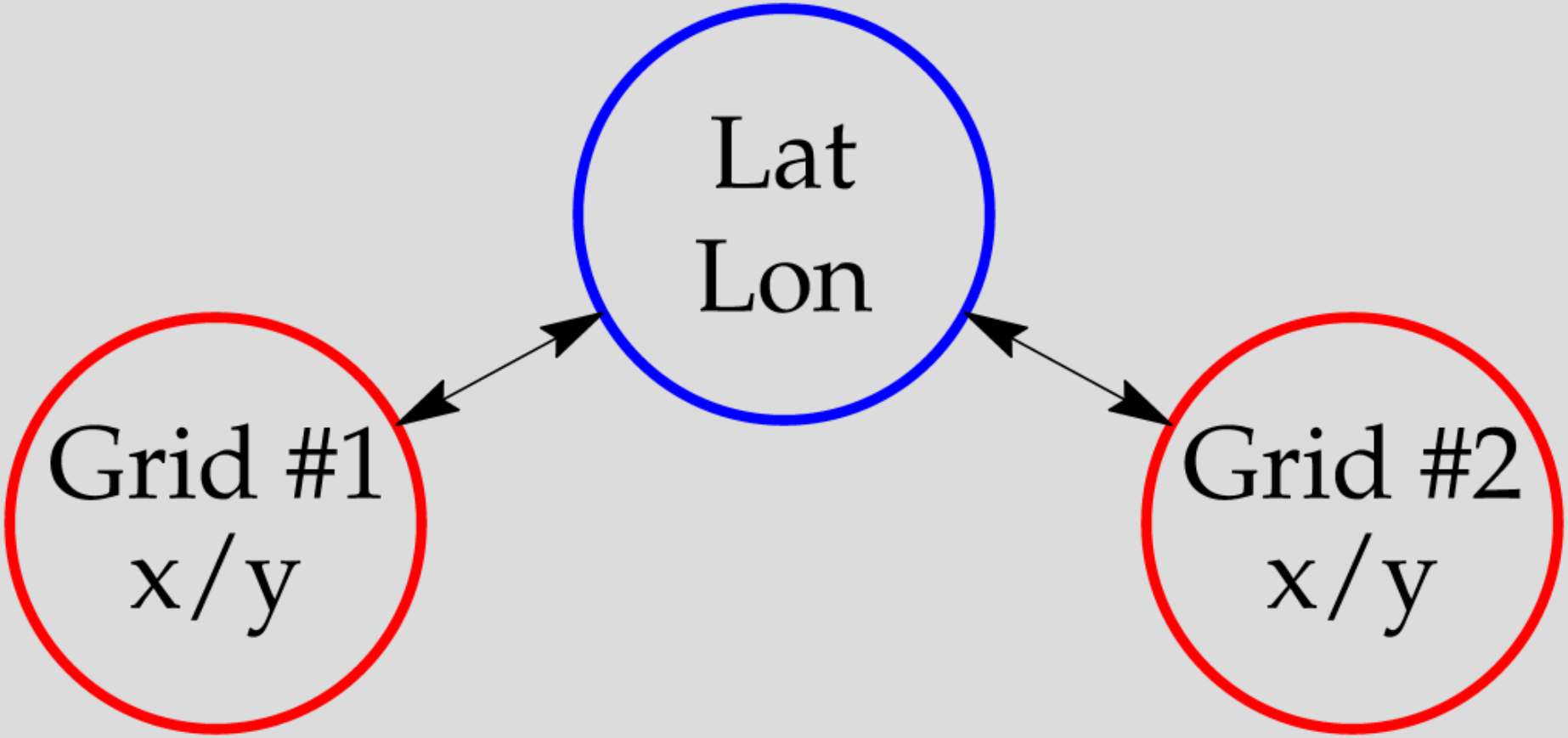
$L = 105°$



$\phi = 40°$

$L = 35°$

# Working with Multiple Grids



# Gridded File Formats

- GRIB1 and GRIB2
  - Uses embedded GRIB1 and GRIB2 tables.
  - Use $MET_GRIB_TABLES to customize.
  - Configure with “--enable-grib2”
- NetCDF
  - Output created by other MET tool.
  - Output of the wrf-interp utility.
  - Climate Forecast (CF) convention (details to follow).
- Use “file_type” config option to explicitly specify.
  - e.g. **file_type = NETCDF_NCCF;**
  - GRIB1, GRIB2, NETCDF_MET, NETCDF_PINT, NETCDF_NCCF

# NetCDF Input to MET

- Recommend CF-Compliance
  - http://cfconventions.org/
  - Time variable and dimension.
  - Forecast reference time.
  - Data for each variable.
  - Grid definition.
  - ”Projection” attribute.

```
netcdf gtg_obs_forecast.20130404.i12.f06 {
dimensions:
        y = 337 ;
        x = 451 ;
        z = 51 ;
        name_len = 32 ;
        time = 1 ;
variables:
        int grid_mapping ;
                grid_mapping:grid_mapping_name = "lambert_conformal_conic" ;
                grid_mapping:standard_parallel = 25. ;
                grid_mapping:longitude_of_central_meridian = -95. ;
                grid_mapping:latitude_of_projection_origin = 25. ;
                grid_mapping:false_easting = 0 ;
                grid_mapping:false_northing = 0 ;
                grid_mapping:GRIB_earth_shape = "spherical" ;
                grid_mapping:GRIB_earth_shape_code = 0 ;
```

# NetCDF Input to MET

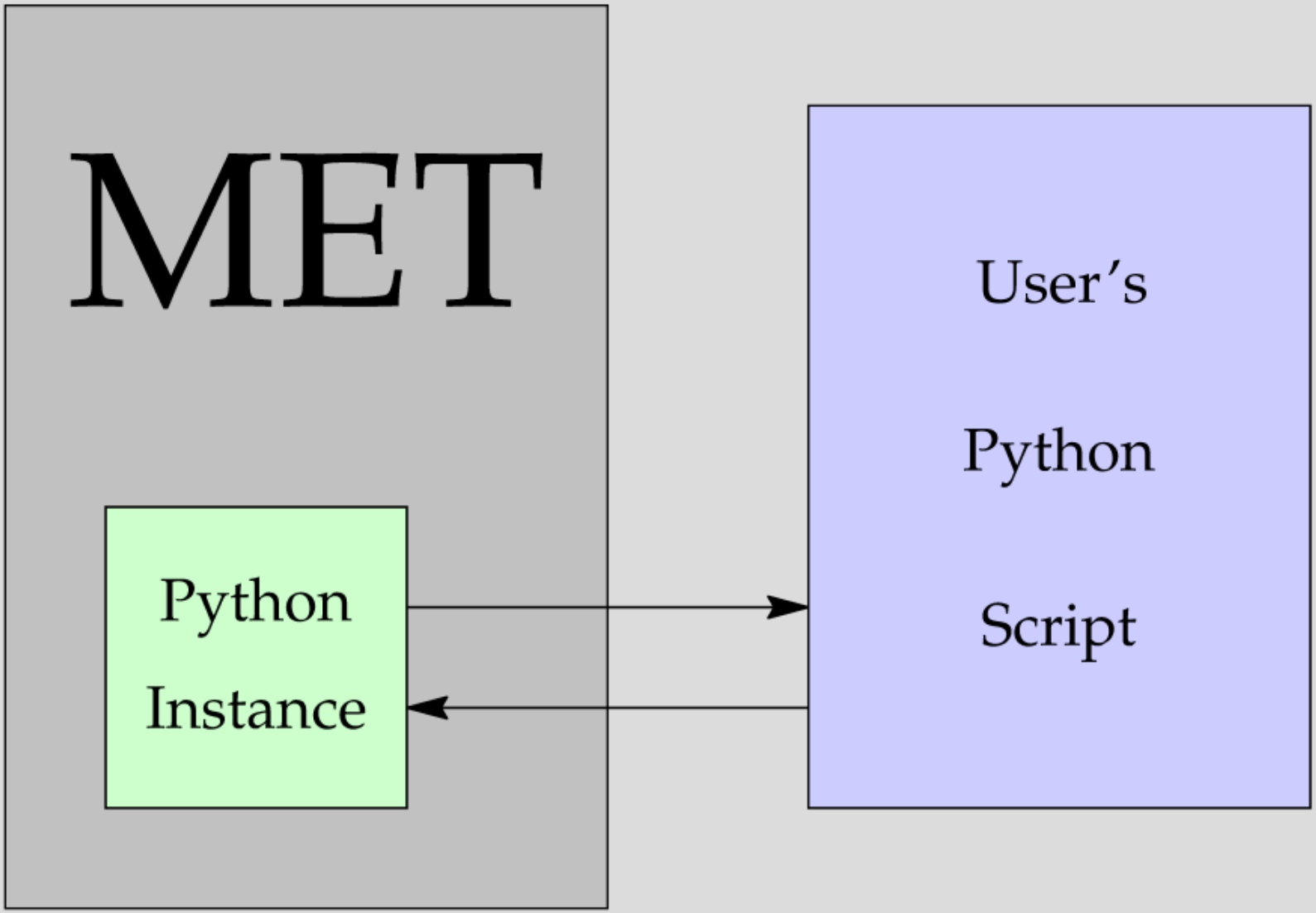
```
double time(time) ;
        time:long_name = "forecast time" ;
        time:units = "seconds since 1970-1-1" ;
        time:axis = "T" ;
double forecast_reference_time ;
        forecast_reference_time:long_name = "reference time" ;
        forecast_reference_time:units = "seconds since 1970-1-1" ;
int forecast_period ;
        forecast_period:units = "seconds" ;
double x(x) ;
        x:standard_name = "projection_x_coordinate" ;
        x:units = "m" ;
        x:grid_spacing = "13545.087 m" ;
        x:axis = "X" ;
double y(y) ;
        y:standard_name = "projection_y_coordinate" ;
        y:units = "m" ;
        y:grid_spacing = "13545.087 m" ;
        y:axis = "Y" ;
double lat(y, x) ;
        lat:units = "degrees_north" ;
        lat:long_name = "latitude coordinate" ;
        lat:standard_name = "latitude" ;
        lat:_CoordinateAxisType = "Lat" ;
double lon(y, x) ;
        lon:units = "degrees_east" ;
        lon:long_name = "longitude coordinate" ;
        lon:standard_name = "longitude" ;
        lon:_CoordinateAxisType = "Lon" ;
        double z(z) ;
        z:standard_name = "altitude" ;
        z:long_name = "flight_level" ;
        z:units = "feet" ;
        z:positive = "up" ;
        z:axis = "Z" ;
int field_id ;
        field_id:long_name = "field indentification number" ;
char field_name(name_len) ;
        field_name:long_name = "field name" ;
float edr(time, z, y, x) ;
        edr:units = "m^(2/3) s^-1" ;
        edr:long_name = "Eddy dissipation rate" ;
        edr:coordinates = "lat lon" ;
        edr:_FillValue = -9999.f ;
        edr:grid_mapping = "grid_mapping" ;

// global attributes:
        :Conventions = "CF-1.4" ;
        :Originating_center = "NCAR" ;
        :Product_Status = "Experimental Products" ;
        :Product_Type = "Forecast/Uninitialized Analysis" ;
        :DataType = "GRID" ;
        :DatasetLocation = "" ;
}
```

# MET Python Embedding

## Embedding Python

## vs.

## Extending Python



# Python Embedding - Motivation

- Support for gridded GRIB1, GRIB2, and NetCDF files is a good start but is not sufficient.
- Many met-help questions about reading gridded data.
  - NetCDF files which do not follow the CF-convention.
  - Binary data files used in operations.
  - Satellite datasets, perhaps in HDF.
  - Fields derived by post-processing one or more input fields.
- Leverage the flexibility of Python to read/derive unsupported data types and pass it in memory to the MET tools.
  - Rather than reformatting data files, use Python to read it.
- Configuring MET with Python support.
  - Disabled by default because of dependencies.
  - Set **MET_PYTHON_CC = `python-config --cflags`**
  - Set **MET_PYTHON_LD = `python-config --ldflags`**
  - Configure with the **--enable-python** option.
- Python 2.7 will move to Python 3 with METplus.

# Python Embedding - Script

1. **User writes a script to read gridded data into a 2-dimensional array.**
   - Use NumPy (tested and described here) or Xarray (untested).
   - Script may use a combination of environment variables and/or command line arguments.
2. **The 2D NumPy array must be named met_data.**
3. **The script defines a dictionary named attr which defines:**
   - **valid** and initialization (**init**) times as strings in YYYYMMDD[_HH[MMSS]] format.
   - **lead** and accumulation (**accum**) times as strings in HH[MMSS] format.
   - **name**, **long_name**, **level**, and **units** as strings.
   - **grid** dictionary defining the projection and grid information in the same way as the gridded NetCDF files produced by MET.
4. **On the command line for the MET tools, replace the path to a gridded data file with the string PYTHON_NUMPY or PYTHON_XARRAY.**
5. **In the field string, set name = Python command to be executed.**

# Python Embedding Example

https://dtcenter.org/community-code/model-evaluation-tools-met/sample-analysis-scripts


## MODEL EVALUATION TOOLS (MET) | SAMPLE ANALYSIS SCRIPTS

View | Edit | Outline | Revisions

This page provides sample scripts that may be run on MET output files to analyze, summarize, and/or plot the data. Feel free to modify these sample scripts to perform the type of analysis you need.

Users are encouraged to submit their own analysis scripts via *met_help@ucar.edu* to be posted on this page.

### Python Scripts

MET version 8.0 includes support for passing data to the MET tools in memory as described in Appendix F of the MET User's Guide. Listed below are examples of using Python to pass data (ASCII, NetCDF, and Binary) to MET's **plot_data_plane** utility:

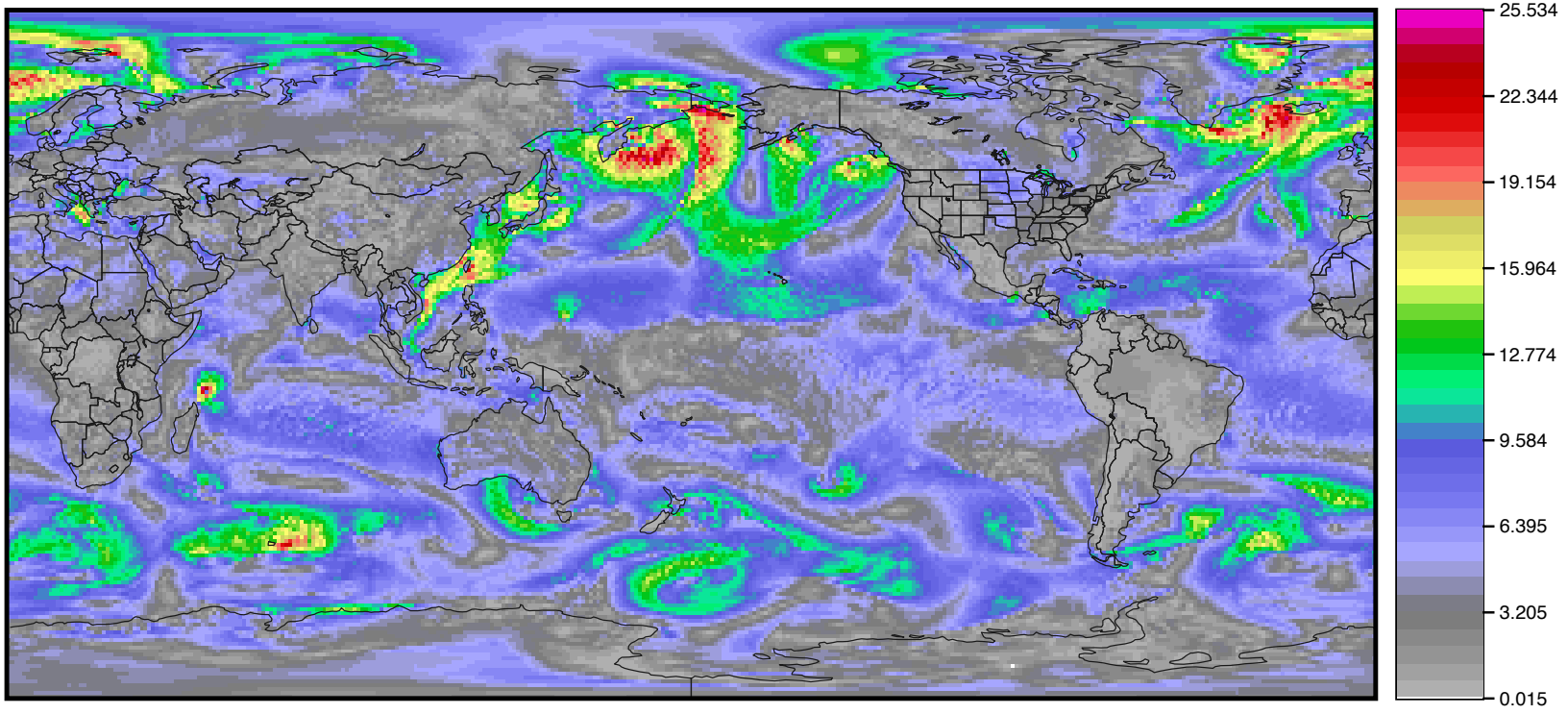
- Plot the 10-th record from NOAA/CPC Legacy GEFS binary dataset:
  **plot_data_plane PYTHON_NUMPY gefs_rec10.ps'name="read_CPC_binary.py gefs-legacy-00z_rfcst-cal_tmean_20170807_8-14day.bin 10";'**

- Plot binary NRL 10-meter wind speed:
  **plot_data_plane PYTHON_NUMPY wind_z10.ps'name="read_NRL_binary.py wndspd_zht_0010.0_0000.0_glob360x181_2016122412_01200000_fcstfld";'**



LATEST RELEASE
MET Version 8.1.1

# Python Embedding Example




PYTHON_NUMPY

**plot_data_plane PYTHON_NUMPY wind_z10.ps**

**'name="read_NRL_binary.py wndspd_zht_0010.0_0000.0_glob360x181_2016122412_01200000_fcstfld";'**

# Future Python Work

- Increased Python Embedding
  - Pass multiple fields of data in memory (e.g. Ensemble-Stat, Series-Analysis, MTD)
  - Pass point observations in memory using pandas data frame.
- Extending Python with MET to enable Python to call MET functions.
- Increased use of Python data libs.
- Increased use of Python data analysis.
- Increased use of Python graphics.