Computational & Information Systems Laboratory

CISL

An Introduction to WRF-Python

Bill Ladwig

NCAR/CISL

NCAR

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

NSF

Motivation

- Part of a larger effort to export NCL numerical routines to Python.
- Began exporting NCL-WRF routines to Python prior to joining NCAR.
- Python is being taught at many universities to science majors.
- WRF not well supported by existing Python tools.

WRF Challenges

- Not NetCDF-CF compliant.
- Curvilinear coordinates, with terrain following vertical (eta levels), on staggered grids.
- Commonly used variables not calculated directly by the model (CAPE, helicity, SLP, etc).
- Output can be one time per file, multiple times per file, multiple times in multiple files.
- Domain nests can move.

Overview of wrf-python

- Similar to the NCL-WRF package.
- Provides over 30 diagnostics calculations.
- Interpolation routines – horizontal level, cross section, vertical surface (e.g. theta-e).
- Works with sequences of output files (lists, dictionaries, generators, and iterables).
- Currently single threaded, but parallel support shouldn't be difficult.
- Works with Python 2 and 3.
- Currently only WRF-ARW supported.

Metadata

- Numpy is the package for array based operations in Python.
- Xarray wraps a numpy array and provides metadata, similar to NCL.
- If xarray is not installed, a numpy array will be returned.
- If xarray is installed, an xarray.DataArray will be returned (unless disabled).
- Xarray is planned for use by the larger AOS Python community for interoperability between packages.

Commonly Used Functions

- **getvar** – Computes a diagnostic variable (cloud top temperature, updraft helcity, etc).
- **interplevel** – Interpolate a 3D field to a horizontal level (pressure or height).
- **vertcross** – Interpolate a 3D field to a vertical cross section.
- **vinterp** – Interpolate a 3D field to a different surface at specified levels (e.g. theta-e at 300 K).

Simple Example

```
from Nio import open_file
from wrf import getvar

# Open a WRF output NetCDF file
ncfile = open_file("wrfout_d01_2016-10-07_00_00_00.nc")

# Get the cloud top temperature
ctt = getvar(ncfile, "ctt")

# Print the xarray object
print (ctt)
```

Simple Example Result

```
<xarray.DataArray u'ctt' (south_north: 1059, west_east: 1799)>
array([[ 22.97872353, 22.96666908, 22.93905067, ...,]], dtype=float32)
Coordinates:
        XLONG (south_north, west_east) float32 -122.72 -122.693 ...
        XLAT (south_north, west_east) float32 21.1381 21.1451 ...
        Time datetime64[ns] 2016-10-07
        * south_north (south_north) int64 0 1 2 3 4 5 6 7 8 9 10 11 ...
        * west_east (west_east) int64 0 1 2 3 4 5 6 7 8 9 10 11 12 ...
Attributes:
        FieldType: 104
        MemoryOrder: XY
        description: cloud top temperature
        units: degC
        stagger:
        coordinates: XLONG XLAT
        projection: LambertConformal(stand_lon=-97.5,
                                     moad_cen_lat=38.5000038147,
                                     truelat1=38.5, truelat2=38.5,
                                     pole_lat=90.0, pole_lon=0.0)
```

CISL Computational & Information Systems Laboratory

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

NSF

Moving Nest Example

```
from Nio import open_file
from wrf import getvar
From glob import glob

# Get the domain 2 output files
wrf_filenames = glob("wrf_files/wrf_vortex_multi/wrfout_d02_*")

# Create a list of wrf file objects from the list of wrf filenames.
wrf_files = [open_file(x + ".nc") for x in wrf_filenames]

# Calculate cloud top temperature for all files.  Concatenate on Time
# dimension.
ctt_all = getvar(wrf_files, "ctt", timeidx=ALL_TIMES, method="cat")

# Print the xarray output
print (ctt_all)
```

Moving Nest Result

<xarray.DataArray u'ctt' (Time: 9, south_north: 96, west_east: 96)>
array([[[30.28734779, 30.25170326, 30.2116642 , ...]]], dtype=float32)
Coordinates:
    XLONG (Time, south_north, west_east) float32 -90.5741 -90.4841 ...
    XLAT (Time, south_north, west_east) float32 21.302 21.302 21.302 ...
    XTIME (Time) float64 0.0 180.0 360.0 540.0 720.0 900.0 1.08e+03 ...
    * Time (Time) datetime64[ns] 2005-08-28 2005-08-28T03:00:00 ...
    * south_north (south_north) int64 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ...
    * west_east (west_east) int64 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ...
    datetime (Time) datetime64[ns] 2005-08-28 2005-08-28T03:00:00 ...
Attributes:
    FieldType: 104
    MemoryOrder: XY
    description: cloud top temperature
    units: degC
    stagger:
    coordinates: XLONG XLAT XTIME
    projection: Mercator(stand_lon=-89.0, moad_cen_lat=27.9999923706,
                         truelat1=0.0, truelat2=0.0, pole_lat=90.0, pole_lon=0.0)

Plot Example (Hurricane Matthew)

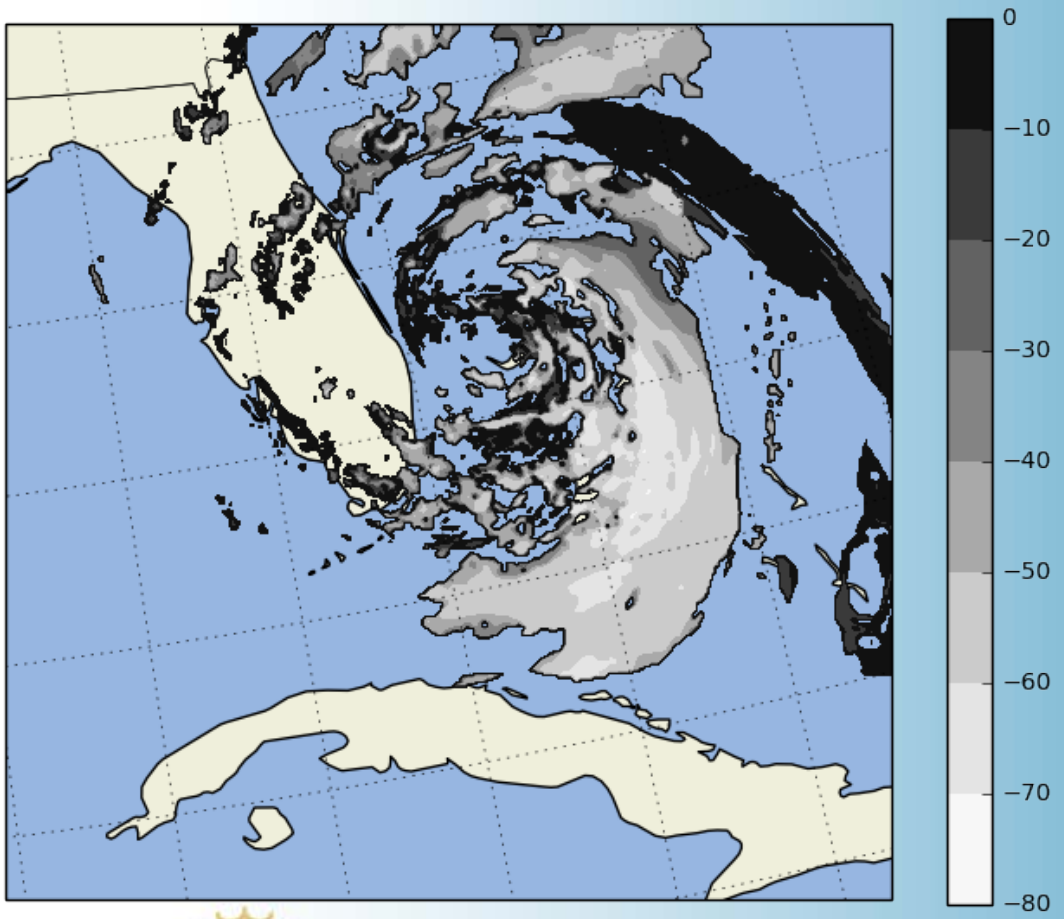
```
from wrf import to_np, getvar, get_cartopy, latlon_coords
from matplotlib import pyplot as plt
from cartopy import crs
.
. (Extract cloud top temperature, etc...)
.
# Get the latitude and longitude points
lats, lons = latlon_coords(ctt)

# Get the cartopy projection object
cart_proj = get_cartopy(ctt)

# Create a figure
fig = plt.figure(figsize=(8,8))
ax = plt.axes([0.1,0.1,0.8,0.8], projection=cart_proj)
.
. (Setting up map options)
.
# Plot the contours, using lat/lon coordinates for each data point.
contour_levels = np.arange(-80.0, 10.0, 10.0)
plt.contourf(to_np(lons), to_np(lats), to_np(ctt), contour_levels,
             cmap=get_cmap("Greys"), transform=crs.PlateCarree(), zorder=2)


# Crop the map to the hurricane region, using lat/lon coordinates
ax.set_extent([-85., -75.0, np.amin(lats), 30.0], crs=crs.PlateCarree())
```

Plot of Cloud Top Temperature

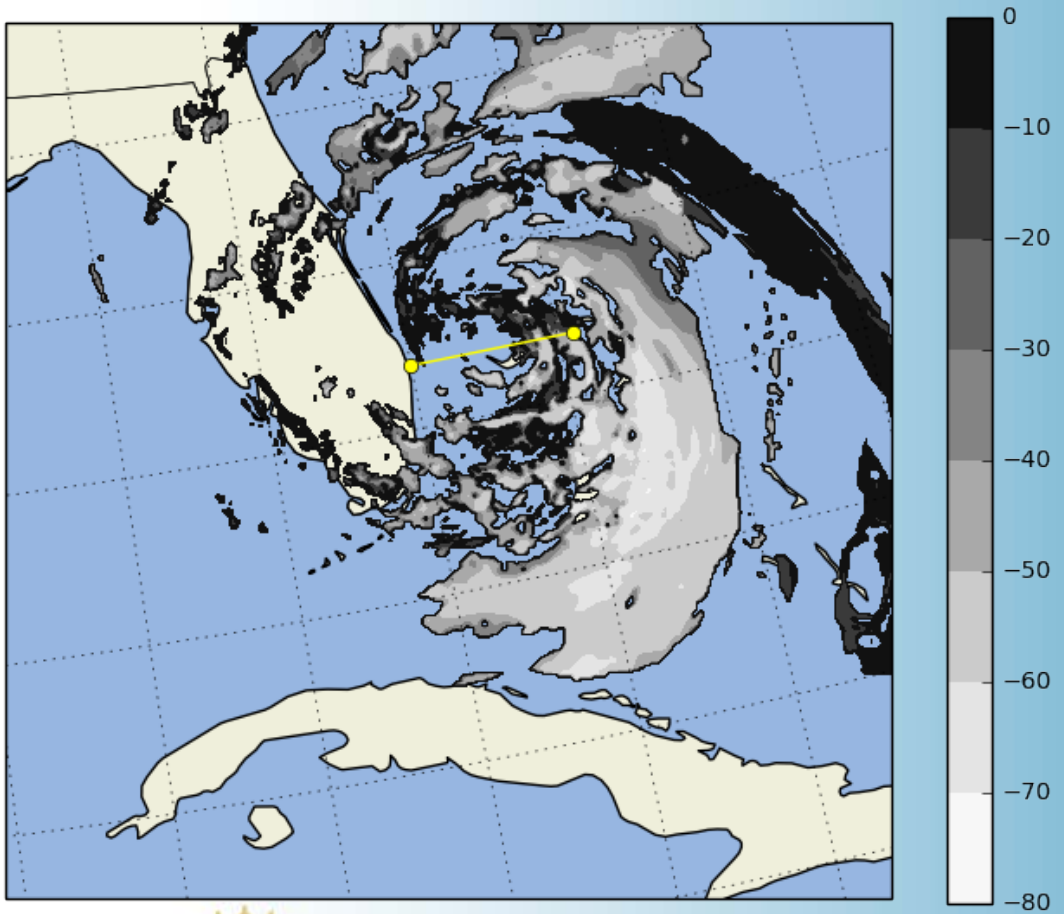


CISL Computational & Information Systems Laboratory

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

NSF

Vertical Cross Section



Vertical Cross Section

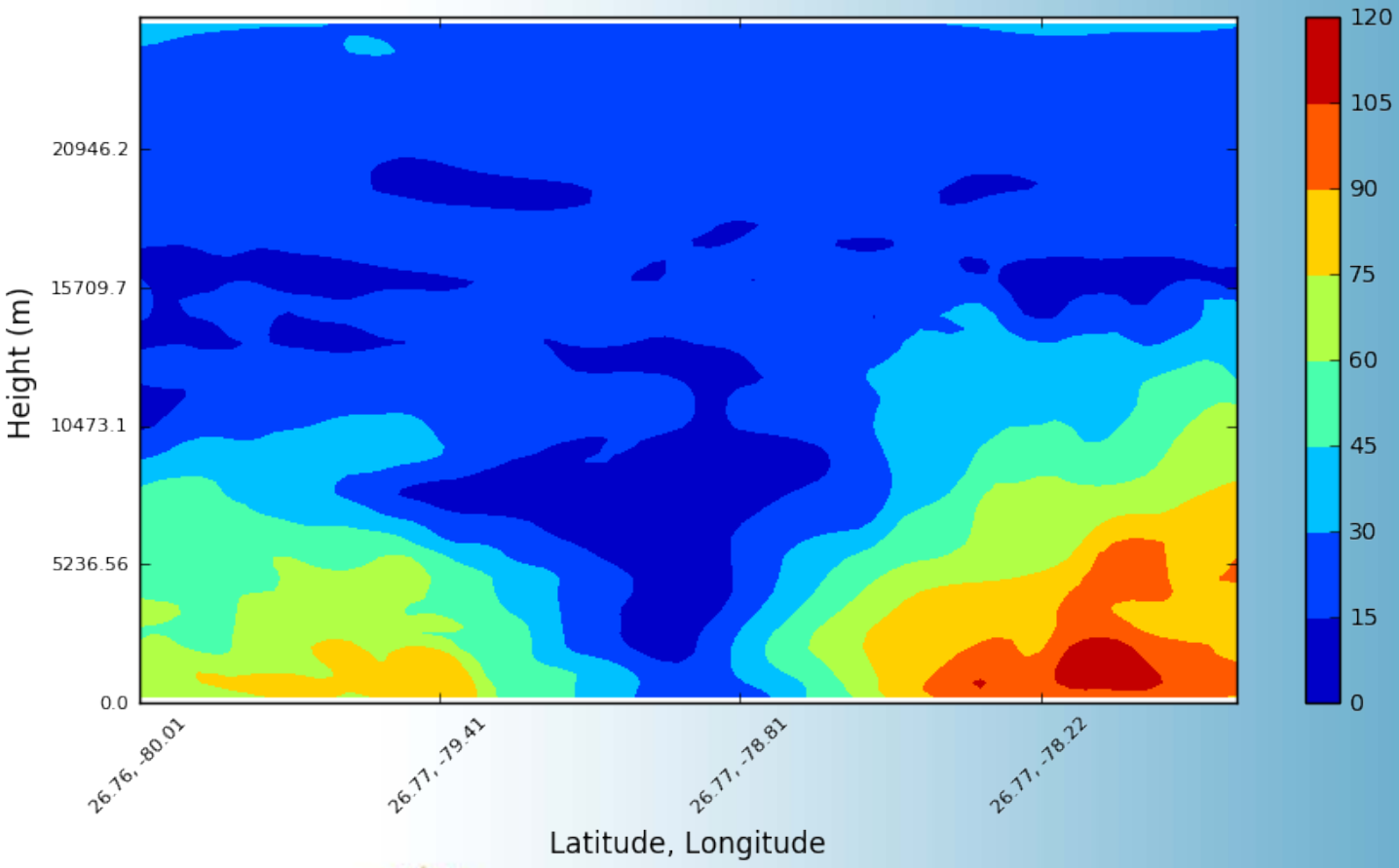
```
from wrf import to_np, getvar, CoordPair, vertcross
from cartopy import crs
From matplotlib import pyplot as plt
.
. (Read netcdf file, etc.)
.

# Get height and wind speed
z = getvar(ncfile, "z"")
wspd =  getvar(ncfile, "uvmet_wspd_wdir", units="kt")[0,:]

# Define the start point and end point, using lat/lon coordinates.
start_point = CoordPair(lat=26.76, lon=-80.0)
end_point = CoordPair(lat=26.76, lon=-77.8)


# Compute the cross section
wspd_cross = vertcross(wspd, z, wrfin=ncfile, start_point=start_point,
                       end_point=end_point)
.
.  (Create the figure, map boundaries, etc.)
.
# Draw contours
wspd_contours = ax.contourf(to_np(wspd_cross))
.
. (Make colorbar, make the axis labels, show figure, etc.)
.
```

Cross Section of Wind Speed (kt)



Computational & Information Systems Laboratory

CISL

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

NSF

Links

- Installing via conda
  - conda install –c conda-forge wrf-python
- Documentation
  - wrf-python.readthedocs.io
- Source
  - https://github.com/NCAR/wrf-python