Technical and social challenges in creating the Python ARM Radar Toolkit (Py-ART)

Jonathan Helmus, Scott Collis
Argonne National Laboratory, Argonne, IL

Argonne National Laboratory is a U.S. Department of Energy laboratory managed by U Chicago Argonne, LLC.

Abstract

The Atmospheric Radiation Measurement (ARM) Climate Research Facility is currently in the process of bringing online and running more than 30 cloud and precipitation radars with the goal of providing meaningful data from these devices to the meteorology and climate modeling communities. Preparing this data for dissemination requires extensive use of numerical algorithms specific to the radar science and remote sensing fields. Although these algorithms have been published and vetted by the community, many lack a robust implementation, either available under an open license or in proprietary software packages. Rather then keep our implementation of these algorithms and other supporting routines internal, we have created the open source Python-ARM Radar Toolkit (Py-ART) to disseminate not only data to end users but also the source code which was used to create this data.

Py-ART offers a powerful interpreted environment for ingesting radar data from a number of formats, correcting for aliasing and attenuation, mapping data to Cartesian grids, creating standard plots of radar fields, and performing a number of geophysical retrievals on the data. The package is also capable of writing data to Climate and Forecast (CF) standard NetCDF files as well as the emerging CF-Radial format for antenna coordinate data.

In addition to discussing the technical aspects of the project some of the social, legal and academic issues in developing a community-focused project will be discussed including: the challenge of creating software which appeals to the wide range of user abilities often found in the scientific community, licensing difficulties when adapting existing codes as well when working with academic institutions and government agencies, the obstacles of developing software in collaboration with scientist with a variety of research interests, funding levels, and software philosophies.

Ingest and writing

Py-ART has the ability to natively ingest (read) radar data from MDV, Sigmet, CF-Radial, as well as other NetCDF based formats. Using a Cython interface, file formats supported by the NASA TRMM Radar Software Library (UF, Lassen, etc) can also be ingested and used by Py-ART. Field data and instrument metadata are stored in memory as a **Radar** object which the routines in Py-ART can interact with. Data can be written out to Climate and Forecast (CF) standard NetCDF files which conform to the the CF/Radial standard.

At the conference, I'll be working to add native support for WSR-88D (NEXRAD) files, if you want to help let me know!

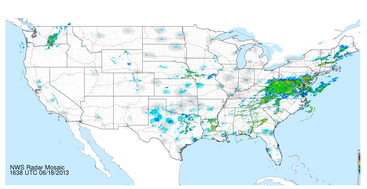
CF-Radial

MDV

Sigmet

UF

Lassen



WSR-88D

Plotting

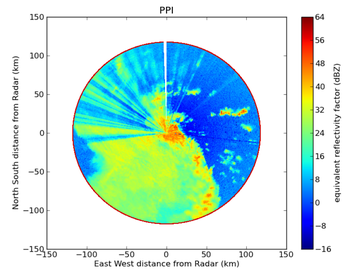
Py-ART can quickly create high quality plots of radar moments. Support for creating plan position indicator (PPI) and range-height indicator (RHI) plots as well as plotting individual rays is included.

PPI

```python
#! /usr/bin/env python

import matplotlib.pyplot as plt
import pyart

radar = pyart.io.read_mdv('110635.mdv')
display = pyart.graph.RadarDisplay(radar)
fig = plt.figure()
ax = fig.add_subplot(111)
display.plot_ppi('reflectivity_horizontal', 0, vmin=-16, vmax=64, title='PPI',
                 ax=ax)
fig.savefig('ppi_plot.png')
```



RHI

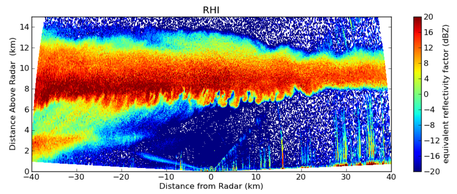
```python
#! /usr/bin/env python

import matplotlib.pyplot as plt
import pyart

radar = pyart.io.read_netcdf('sgpxsaprrhicmacI5.c0.20110524.015604_NC4.nc')
display = pyart.graph.RadarDisplay(radar)

fig = plt.figure(figsize=(12, 4))
fig.subplots_adjust(hspace=0.4)
ax = fig.add_subplot(111)

display.plot_rhi('reflectivity_horizontal', 0, vmin=-20, vmax=20, title='RHI',
                 ax=ax)
display.set_limits(ylim=[0, 15])
fig.savefig('rhi_plot.png')
```



Mapping and Gridding

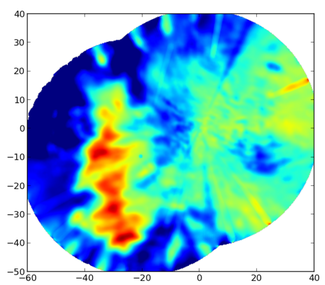
Radar data is collected in antenna coordinates (similar to radial coordinates). For many purposes, it is useful to map these data to a Cartesian grid. Py-ART includes routines to perform this gridding on moments from one or multiple radars using a nearest neighbor distance-weighted interpolation which utilizes a KD-Tree or Ball Tree for efficient nearest neighbor lookups. The resulting gridded data and metadata is stored in a **Grid** object.

```python
import matplotlib.pyplot as plt
import pyart

# read in the data
radar_e = pyart.io.read_netcdf('east_radar.nc')
radar_w = pyart.io.read_netcdf('west_radar.nc')

# perform gridding
grid = pyart.map.grid_from_radars(
    (radar_e, radar_w),
    (401, 401, 2),
    ((-60000, 40000), (-50000, 40000), (0, 1000)),
    grid_origin=(36.57861, -97.363611),
    max_refl=100.)

# create plot
fig = plt.figure()
ax = fig.add_subplot(111)
refl = grid.fields['reflectivity_horizontal']['data']
ax.imshow(refl[1], origin='lower', vmin=0, vmax=48,
          extent=(-60, 40, -50, 40))
fig.savefig('figure.png')
```



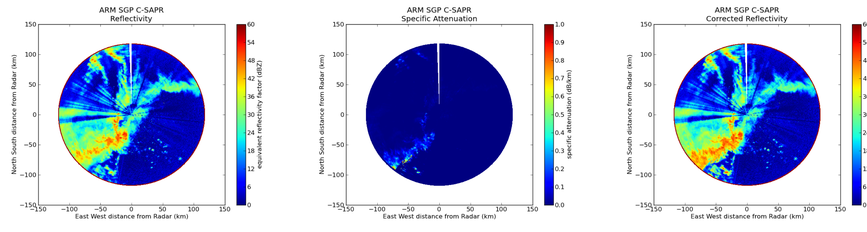
This poster has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. This research was supported by the Office of Biological and Environmental Research of the U.S. Department of Energy as part of the Atmospheric Radiation Measurement Climate Research Facility.

U.S. DEPARTMENT OF ENERGY

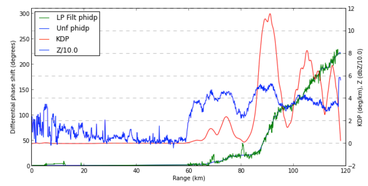
Antenna coordinate moment corrections

Py-ART includes implementations of many algorithms which can be used to correct and analyze radar moments in antenna coordinates.

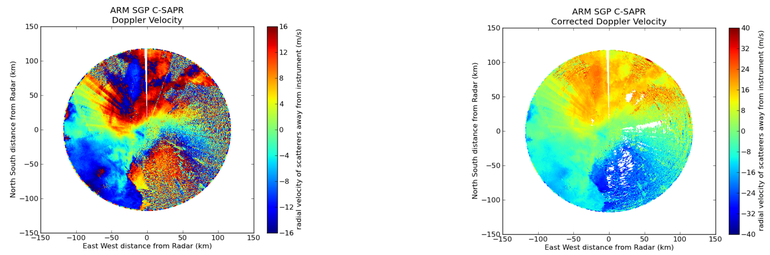
Z-PHI based attenuation correction



Phase correction using an LP algorithm



Univ. Washington 4D Velocity Dealiasing



Additional information and Acknowledgements

Py-ART is an open source project which is distributed under a **BSD license**. The source code is hosted at GitHub. We welcome forks, contributions, bug-reports and suggestions. For additional information or questions about the package please contact the lead developer, Jonathan Helmus (jjhelmus@anl.gov).

Fork me on GitHub

https://github.com/ARM-DOE/pyart

Documentation for Py-ART is available online at: **http://arm-doe.github.io/pyart/dev/index.html**

Thanks go to the following for providing algorithms, code and support:
Scott Collis, Scott Giangrande, Kirk North, Alexander Ryzhkov, Matthias Steiner, Bart Kelley, Eric Bruning,
… and many others!

Py-ART would not be possible without the hard work of many other open source Scientific Python packages such as NumPy, SciPy, matplotlib, Cython, and python-netcdf4.

NumPy Cython matplotlib