# The Python-ARM Radar Toolkit (Py-ART), an open source package for weather radar.

Jonathan Helmus[1], Scott Collis[1], Karen Johnson[2], Kirk North[3], Scott Giangrande[2], and Michael Jensen[2].

[1]Environmental Science Division, Argonne National Laboratory
[2]Environmental Sciences Department, Brookhaven National Laboratory
[3]Atmospheric and Oceanic Sciences, McGill University

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

## Abstract

With the acquisition of a number of X- and C-band scanning precipitation radars, the Atmospheric Radiation Measurement (ARM) climate facility now routinely deals with large amounts of complex data from these instruments. The measured parameters from these radars as well as value-added products (VAPs) derived from these data are made available to the radar, cloud and climate modeling communities. Preparing these data for dissemination requires extensive use of computational resources and algorithms, which are not well addressed by current software packages. To address these needs we have developed the Python-ARM Radar Toolkit (Py-ART), an open source package for working with weather radar data.

Py-ART offers a powerful interpreted environment for ingesting radar data from a number of formats, correcting for aliasing and attenuation, mapping radar coordinate data from a single or multiple radars to a Cartesian grid, and performing a number of geophysical retrievals on the data. The package is capable of creating Climate and Forecast (CF) standard NetCDF files as well as files in the emerging CF-Radial format for antenna coordinate data. Py-ART is written in the Python programming language, taking advantage of the powerful scientific libraries NumPy, SciPy, and matplotlib, available to the language, as well as interfacing with legacy C and FORTRAN radar code. The package is available freely under a BSD license and can be downloaded from https://github.com/ARM-DOE/pyart/.

## Ingest and Writing

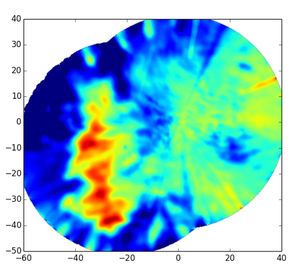
Py-ART has the ability to natively ingest (read) radar data from MDV, Sigmet, NEXRAD Level 2, and CF/Radial, as well as other NetCDF based formats. Using a Cython interface, file formats supported by the NASA TRMM Radar Software Library (UF, Lassen) can also be ingested and used by Py-ART. Field data and instrument metadata are stored in memory as a **Radar** object with which the routines in Py-ART can interact. Data can be written out to Climate and Forecast (CF) standard NetCDF files which conform to the the CF/Radial standard.

| Format | Native Py-ART Ingest | RSL Ingest |
|---|---|---|
| CF/Radial | X (and writing) | |
| MDV | X | |
| Sigmet | X | X |
| UF | | X |
| NEXRAD (WSR-88D) | Level II | Level II |
| Lassen | | X |

## 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.

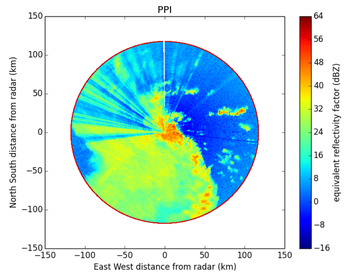
```python
import matplotlib.pyplot as plt
import pyart

# read in the data from the SW and SE ARM-XSAPR radars
radar_sw = pyart.io.read_netcdf('swx_20120520_0641.nc')
radar_se = pyart.io.read_netcdf('sex_20120520_0641.nc')

# perform Cartesian mapping, limit to the reflectivity field.
grid = pyart.map.grid_from_radars(
    (radar_se, radar_sw),
    grid_shape=(401, 401, 2),
    grid_limits=((-60000, 40000), (-50000, 40000), (0, 1000)),
    grid_origin = (36.57861, -97.363611),
    refl_field='corrected_reflectivity_horizontal',
    max_refl=100.)

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



## 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. More elaborate plots can be created using the functionality provided by Py-ART and the matplotlib plotting package.

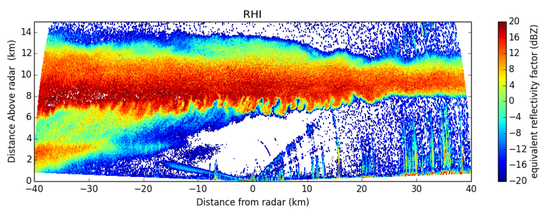
### PPI




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

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

### RHI

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

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

fig = plt.figure(figsize=(11, 4))
ax = fig.add_subplot(111)
display.plot_rhi('reflectivity_horizontal', 0,
                 vmin=-20, vmax=20, title='RHI')
display.set_limits(ylim=[0, 15])
plt.tight_layout()
plt.savefig('figure.png')
```




## 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 (jhelmus@anl.gov).

Fork me on GitHub

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

Documentation and examples are available online at: **http://arm-doe.github.io/pyart**

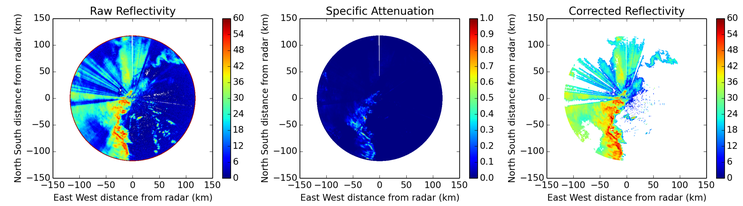
Thanks go to the following for providing algorithms, code and support:
Alexander Ryzhkov, Matthias Steiner, Bart Kelley, Eric Bruning… *and many others!*

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.

## Antenna coordinate moment corrections

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

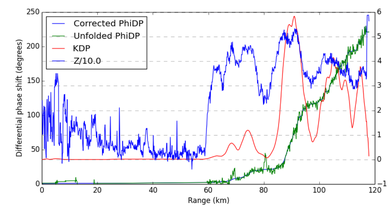
### Z-PHI based attenuation correction




```python
import pyart

# read in the data
radar = pyart.io.read_netcdf('sgpcsaprsurcmacI7.c0.20110520.095101.nc')

# perform attenuation correction
spec_at, cor_z = pyart.correct.calculate_attenuation(radar, 0)
radar.fields['specific_attenuation'] = spec_at
radar.fields['corrected_reflectivity_horizontal'] = cor_z
```

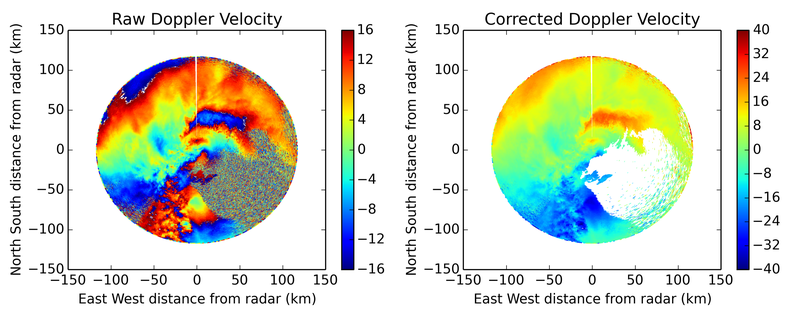
### Phase correction using an linear programming (LP) algorithm

```python
import pyart

# perform LP phase processing
radar = pyart.io.read('095636.mdv')
phidp, kdp = pyart.correct.phase_proc_lp(radar, 0.0, debug=True)
radar.fields['proc_dp_phase_shift'] = phidp
radar.fields['recalculated_diff_phase'] = kdp
```




### Univ. Washington 4D Velocity Dealiasing

```python
import netCDF4
import pyart

# read in the data
radar = pyart.io.read('095636.mdv')

# find and extract sonde data
target = netCDF4.num2date(radar.time['data'][0], radar.time['units'])
interp_sounde = netCDF4.Dataset('sgpinterpolatedsondeC1.c1.20110510.000000.cdf')
t = pyart.correct.find_time_in_interp_sonde(interp_sounde, target)
height, speed, direction = t

# perform dealiasing
dealias_data = pyart.correct.dealias_fourdd(radar, height * 1000.0, speed,
                                            direction, target)
radar.fields['corrected_mean_doppler_velocity'] = dealias_data
```




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.