

Py-ART CHEAT SHEET

LEARN MORE ABOUT PY-ART[1] AT [HTTP://ARM-DOE.GITHUB.IO/PYART/](http://arm-doe.github.io/pyart/)

Py-ART INTRODUCTION

The Python ARM Radar Toolkit, Py-ART, is a Python module containing a collection of weather radar algorithms and utilities. Py-ART is used by the Atmospheric Radiation Measurement (ARM) Climate Research Facility for working with data from a number of its precipitation and cloud radars, but has been designed so that it can be used by others in the radar and atmospheric communities. Py-ART has the ability to ingest (read) from a number of common weather radar formats. Radar data can be written to NetCDF files which conform to the CF/Radial convention. Py-ART also contains routines which can produce common radar plots including PPIs and RHIs. Algorithms in the module are able to performs a number of corrections on the radar moment data in antenna coordinate. A sophisticated mapping routines is able to efficiently create uniform Cartesian grids of radar fields from one or more radars.

INSTALLATION

The recommended way to install Py-ART is by using Anaconda or Miniconda, then create an environment and activate it:

- First install Anaconda or Miniconda

- Then create a conda environment:
\$ conda create -n pyart python=3.6

- Activate the Py-ART environment:
\$ source activate pyart

- Then install Py-ART:
\$ conda install -c conda-forge arm_pyart

For the most recent Py-ART developments, you need to get the latest master branch from github.com

- Clone the Py-ART repository:
\$ git clone <https://github.com/ARM-DOE/pyart.git>

- Go into the Py-ART directory and then:
\$ python setup.py install

CONTACT INFORMATION

Google Groups:
<https://groups.google.com/d/forum/pyart-users>

Email:
scollis@anl.gov
zsherman@anl.gov

Issues:
<https://github.com/ARM-DOE/pyart/issues>

GETTING STARTED

```
>>> import pyart           To import Py-ART.
>>> print(pyart.__version__) Check version.
```

READING AND WRITING DATA

Reading Data

```
>>> radar = pyart.io.read(filename)
    • Read a file supported by RSL.
    • Read a MDV file.
    • Read a Sigmet (IRIS) product file.
    • Read a Cfradial netCDF file.
    • Read a CSU-CHILL CHL file.
    • Read a NEXRAD Level 2 Archive file.
    • Read a Common Data Model NEXRAD 2 file.
    • Read a NEXRAD Level 3 product.
    • Read a UF File.
>>> radar = pyart.aux_io.read_d3r_gcpx_nc(file)
    • Read a D3R GCPEX netCDF file.
>>> radar = pyart.aux_io.read_gamic(filename)
    • Read a GAMIC hdf5 file.
```

READING AND WRITING DATA

```
>>> radar = pyart.aux_io.read_kazr(filename)
    • Read K-band ARM Zenith Radar (KAZR) data.
>>> radar = pyart.aux_io.read_noxp_iphex_nc(file)
    • Read a NOXP IPHEX netCDF file.
>>> radar = pyart.aux_io.read_odim_h5(filename)
    • Read a ODIM_H5 file.
>>> radar = pyart.aux_io.read_radx(filename)
    • Read a radx file by using RadxConvert.
>>> radar = pyart.aux_io.read_pattern(filename)
    • Read PATTERN project X-band radar file.
>>> radar = pyart.aux_io.read_rainbow_wrl(file)
    • Read a RAINBOW file.
```

Writing Radar Data

```
>>> pyart.io.write_cfradial(filename, radar)
>>> pyart.io.write_uf(filename, radar)
```

Reading Grid Data

```
>>> grid = pyart.io.read_grid(filename)
>>> grid = pyart.io.read_grid_mdv(filename)
```

Writing Grid Data

```
>>> pyart.io.write_grid(filename, grid)
>>> pyart.io.write_grid_mdv(filename, grid)
>>> pyart.io.write_grid_geotiff(grid, filename)
```

Reading Sonde Data

```
>>> sonde = pyart.io.read_sonde(filename)
>>> sonde = pyart.io.read_sonde_vap(
    filename[, radar, ...])
```

GRAPHING DATA

Radar Data

```
>>> pyart.graph.RadarDisplay(radar)
    • Display object to plot data from a radar object.
>>> pyart.graph.RadarMapDisplay(radar)
    • Plots data to a geographic map.
>>> pyart.graph.AirborneRadarDisplay(radar)
    • Plots data from a airborne radar object.
>>> pyart.graph.RadarMapDisplayCartopy(radar)
    • Plots data to a geographic map using Cartopy
```

After defining the display objects above, each have specific functions for plots. For Example:

```
>>> display.plot(field[, sweep])
>>> display.plot_ppi(field[, sweep, vmin, ...])
>>> display.plot_ppi_map(field[, sweep, ...])
>>> display.plot_rhi(field[, sweep, vmin, ...])
>>> display.plot_vpt(field[, vmin, vmax, ...])
```

Grid Data

```
>>> pyart.graph.GridMapDisplay(grid)
>>> display.plot_grid(field[, level, vmin, vmax, ...])
>>> display.plot_latitudinal_level(
    field, y_index[, ...])
>>> display.plot_longitudinal_level(
    field, x_index[, ...])
```

MAPPING DATA

```
>>> grid = pyart.map.grid_from_radars(
    radars, grid_shape, grid_limits)
    • Map one or more radars to a Cartesian grid
    returning a grid object.
>>> grids = pyart.map.map_to_grid(
    radars, grid_shape, grid_limits)
    • Map one or more radars to a Cartesian grid.
>>> grids = pyart.map.map_gates_to_grid(
    radars, grid_shape, grid_limits)
    • Map gates from radar(s) to a Cartesian grid.
```

CORRECTIONS

GateFilters

A class for building a boolean arrays for filtering gates based on a set of condition typically based on the values in the radar fields.

```
>>> gatefilter = pyart.correct.GateFilter(radar)
>>> gatefilter.exclude_all()
>>> gatefilter.exclude_below(field, 10)
>>> gatefilter.exclude_masked(field)
```

Velocity Unfolding

```
>>> pyart.correct.dealias_fourdd(radar[, ...])
    • Dealias Doppler velocities using 4DD
    algorithm [2].
>>> pyart.correct.dealias_region_based(radar[, ...])
    • Dealias velocities using a region based algorithm.
>>> pyart.correct.dealias_unwrap_phase(radar[, ...])
    • Dealias Doppler velocities using
    multi-dimensional phase unwrapping.
```

Other Corrections

```
>>> pyart.correct.calculate_attenuation(
    radar, z_offset[, ...])
    • Calculate the attenuation from a polarimetric
    radar using Z-PHI method.
>>> pyart.correct.phase_proc_lp(radar, offset[, ...])
    • Phase process using a LP method [3].
>>> pyart.correct.despeckle_field(radar, field[, ...])
    • Despeckle a radar volume by identifying small
    objects in each scan and masking them out.
>>> pyart.correct.correct_noise_rhohv(radar[, ...])
    • Corrects RhoHV for noise [4].
>>> pyart.correct.correct_bias(radar[, bias, ...])
    • Corrects a radar data bias.
```

RETRIEVALS

```
>>> pyart.retrieve.kdp_maesaka(radar[, ...])
    • Computes the specific differential phase (KDP)
    based on Maesaka et al [5].
>>> pyart.retrieve.compute_snr(radar[, ...])
    • Computes SNR from a reflectivity field and
    the noise in dBZ.
>>> pyart.retrieve.compute_l(radar[, ...])
    • Computes RhoHV in logarithmic scale according
    to L=-log10(1-RhoHV)
>>> pyart.retrieve.compute_cdr(radar[, ...])
    • Computes the Circular Depolarization Ratio.
>>> pyart.retrieve.steiner_conv_strat(grid[, ...])
    • Partition reflectivity into convective-stratiform
    using the Steiner et al. [6].
>>> pyart.retrieve.hydroclass_semisupervised(
    radar[, ...])
    • Classifies precipitation echoes following the
    approach by Besic et al. [7].
>>> pyart.retrieve.texture_of_complex_phase(
    radar[, ...])
    • Calculate the texture of the differential phase.
    field [8].
>>> pyart.retrieve.est_rain_rate_z(
    radar[, alpha, beta, ...])
    • Estimates rainfall rate from reflectivity using a
    power law.
>>> pyart.retrieve.est_rain_rate_kdp(
    radar[, alpha, beta, ...])
    • Estimates rainfall rate from kdp using alpha
    power law.
>>> pyart.retrieve.est_rain_rate_a(
    radar[, alpha, beta, ...])
    • Estimates rainfall rate from specific attenuation
    using alpha power law [9], [10].
>>> pyart.retrieve.est_rain_rate_hydro(
    radar[, alphazr, ...])
    • Estimates rainfall rate using different relations
    between R and the polarimetric variables
    depending on the hydrometeor type.
```


Py-ART CHEAT SHEET

LEARN MORE ABOUT PY-ART[1] AT [HTTP://ARM-DOE.GITHUB.IO/PYART/](http://arm-doe.github.io/pyart/)

UTILITIES

- Direction Statistics
- >>> pyart.utils.angular_mean(angles)
 - Compute the mean of a distribution of angles in radians.
- >>> pyart.utils.angular_std(angles)
 - Compute the standard deviation of a distribution of angles in radians.
- >>> pyart.utils.angular_mean_deg(angles)
 - Compute the mean of a distribution of angles in degrees.
- >>> pyart.utils.angular_std_deg(angles)
 - Compute the standard deviation of a distribution of angles in degrees.
- >>> pyart.utils.interval_mean(dist, interval_min, interval_max)
 - Compute the mean of a distribution within an interval.
- >>> pyart.utils.interval_std(dist, interval_min, interval_max)
 - Compute the standard deviation of a distribution within an interval.
- >>> pyart.utils.mean_of_two_angles(angles1, angles2)
 - Compute the element by element mean of two sets of angles.
- >>> pyart.utils.mean_of_two_angles_deg(angle1, angle2)
 - Compute the element by element mean of two sets of angles in degrees.
- Miscellaneous Utilities
- >>> pyart.utils.cross_section_ppi(radar, target_azimuths[, ...])
 - Extract cross sections from a PPI volume along one or more azimuth angles.
- >>> pyart.utils.cross_section_rhi(radar, target_elevations)
 - Extract cross sections from a RHI volume along one or more elevation angles.
- >>> pyart.utils.estimate_noise_hs74(spectrum[, navg])
 - Estimate noise parameters of a Doppler spectrum [11].
- >>> pyart.utils.to_vpt(radar[, single_scan])
 - Convert an existing Radar object to represent a vertical pointing scan.
- >>> pyart.utils.join_radar(radar1, radar2)
 - Combine two radar instances into one.
- >>> pyart.utils.simulated_vel_from_profile(radar, profile[, ...])
 - Create simulated radial velocities from a profile of horizontal winds.
- >>> pyart.utils.texture_along_ray(radar, var[, wind_size])
 - Compute field texture along ray using a user specified window size.
- >>> pyart.utils.rolling_window(a, window)
 - Create a rolling window object for application of functions.
- >>> pyart.utils.angular_texture_2d(image, N, interval)
 - Compute the angular texture of an image.

REFERENCES

[1] Helmus, J.J. Collis, S.M., 2016: The Python ARM Radar Toolkit (Py-ART), a Library for Working with Weather Radar Data in the Python Programming Language. *Journal of Open Research Software*. **4(1)**, p.e25.

[2] C. N. James and R. A Houze Jr, 2001: A Real-Time Four-Dimensional Doppler Dealising Scheme. *Journal of Atmospheric and Oceanic Technology*, **18**, 1674.

[3] Giangrande, S.E., R. McGraw, and L. Lei., 2013: An Application of Linear Programming to Polarimetric Radar Differential Phase Processing. *J. Atmos. and Oceanic Tech*, **30**, 1716.

[4] Gourley et al., 2006: Data Quality of the Meteo-France C-Band Polarimetric Radar. *JAOT*, **23**, 1340-1356

[5] Maesaka, T., Iwanami, K. and Maki, M., 2012: Non-negative KDP Estimation by Monotone Increasing PHIDP Assumption below Melting Layer. *The Seventh European Conference on Radar in Meteorology and Hydrology*.

[6] Steiner, M. R., R. A. Houze Jr., and S. E. Yuter, 1995: Climatological Characterization of Three-Dimensional Storm Structure from Operational Radar and Rain Gauge Data. *J. Appl. Meteor.*, **34**, 1978-2007.

[7] Besic, N., Figueras i Ventura, J., Grazioli, J., Gabella, M., Germann, U., and Berne, A., 2016: Hydrometeor classification through statistical clustering of polarimetric radar measurements: a semi-supervised approach. *Atmos. Meas. Tech.*, **9**, 4425-4445

[8] Gourley, J. J., P. Tabary, and J. Parent du Chatelet, 2007: A fuzzy logic algorithm for the separation of precipitating from nonprecipitating echoes using polarimetric radar observations. *Journal of Atmospheric and Oceanic Technology* **24(8)**, 1439-1451

[9] Diederich M., Ryzhkov A., Simmer C., Zhang P. and Tromel S., 2015: Use of Specific Attenuation for Rainfall Measurement at X-Band Radar Wavelengths. Part I: Radar Calibration and Partial Beam Blockage Estimation. *Journal of Hydrometeorology*, **16**, 487-502.

[10] Ryzhkov A., Diederich M., Zhang P. and Simmer C., 2014: Potential Utilization of Specific Attenuation for Rainfall Estimation, Mitigation of Partial Beam Blockage, and Radar Networking. *Journal of Atmospheric and Oceanic Technology*, **31**, 599-619.

[11] Hildebrand P.H., and R. S. Sekhon, 1974: Objective Determination of the Noise Level in Doppler Spectra. *Journal of Applied Meteorology*, **13**, 808-811.