Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

The MeteoSwiss Py-ART

Daniel Wolfensberger, Jordi Figueras i Ventura

Contents

1. Introduction

What is Py-ART?

- The Python ARM Radar Toolkit (Py-ART) was initially created to work with the data produced by radars of the Atmospheric Radiation Measurement Climate Research facility (ARM) programme of the US Department of Energy
- It was first released in 2013 as an open-source software
- It relies heavily on the scientific python stack (numpy, scipy, matplotlib, pandas, cartopy, etc.)
- The ARM-DOE Py-ART can be used for:
  - Reading radar data in a variety of file formats
  - Creating plots and visualization of radar data
  - Some corrections of radar moments (Doppler de-aliasing, attenuation correction, etc.)
  - Mapping data from one or more radars onto a Cartesian grid
  - Performing some retrievals
  - Writing radar and Cartesian data to NetCDF files

Why using the MeteoSwiss Py-ART ?

- The ARM-DOE Py-ART is a library of basic building blocks for data reading and visualization. The software is high quality and well maintained but has a limited scope
- The MeteoSwiss Py-ART adds many additional corrections and retrievals that were developed to serve semi-operational data processing chains
- Some functionalities available on the MeteoSwiss Py-ART are transferred to the ARM-DOE Py-ART

2. Py-ART architecture

Py-ART modules

core	Data objects Coordinate transforms
io	Reading and writing
aux_io	Non-standard Reading and writing
filters	Filtering (removing of undesired gates)
correct	Correction of radar fields e.g. attenuation, dealiasing
retrive	Radar retrieval e.g. rainfall rate, melting layer, etc.
map	Mapping radar data from radar to Cartesian coordinates
graph	Plots of radar and grid fields
util	Auxiliary functions
bridge	Bridge to other software packages, e.g. wradlib
testing	Utilities to facilitate the generation of unit tests
tests	Unit tests

Py-ART data objects

Object	Purpose	Comment
Radar	Store radar data in antenna coordinates. The data structure is based on C/F Radial V1	Data fields are stored in a 2-D matrix (ray, range) Some Pyrad applications use the same structure to store (time, range). **Assumes uniform range resolution !**
RadarSpectra	Store IQ spectral and spectral data in antenna coordinates. Inherits from Radar object	Data fields are stored in a 3-D complex matrix (ray, range, Doppler bin/slow time)
Grid	Store rectilinear gridded data in Cartesian coordinates	Data fields are stored in a 3-D matrix (z, y, x) The grid is referenced as distance from the grid origin. **Assumes uniform spacing of the data !**
HorizontalWindProfile	Store horizontal wind profile data	Not used by Pyrad

Py-ART plotting objects

Object	Purpose	Functions
radardisplay	Display object to create plots from data in a radar object	plot_ray, plot_ppi, plot_rhi, plot_azimuth_to_rhi (pseudo-RHI), plot_vpt (time-height for vertically pointing radar), plot_xsection
radardisplay_airborne	Same as above but for airborne radar data. Inherits from radardisplay	Not used by Pyrad
radarmapdisplay	Display object to create plots on a geographic map from data in a radar object. Inherits from radardisplay	plot_ppi_map
gridmapdisplay	Display object to create plots from data in a grid object	plot_grid (plot grid data projected into a map), plot_grid_raw (plot grid in native Cartesian coordinates) plot_grid_contour (plot contours of grid data projected into a map) plot_latitude_slice (plot slice along a given latitude) plot_longitude_slice (plot slice along a given longitude) plot_latlon_slice (plot slice crossing 2 arbitrary coordinate points)

3. Data processing with Py-ART

Basic data processing

Level	Location	Purpose
IQ and spectral data processing	retrieve/iq.py retrieve/spectra.py	Computation of raw moments from IQ and/or spectral data
Noise and bias corrections	retrieve/simple_moment_calculations.py correct/bias_and_noise.py correct/despeckle.py	Compute noise level and correct raw moments for noise and biases
Clutter suppression	filters/gatefilter.py	Mask undesired data
PhiDP/KDP retrieval	correct/phase_proc.py retrieve/kdp_proc.py	Implementation of various PhiDP/KDP retrieval algorithms available in literature
Doppler velocity unfolding	correct/dealias.py correct/region_dealias.py correct/unwrap.py	Implementation of various unfolding techniques
Melting layer detection	retrieve/ml.py	Implementation of various melting layer detection algorithms available in literature
Attenuation correction	correct/attenuation.py	Implementation of various attenuation correction algorithms available in literature
VPR correction	correct/vpr.py	Implementation of the VPR correction algorithm operational at Météo-France

Retrievals

Level	Location	Purpose
Hydrometeor classification	retrieve/echo_class.py	Convective/stratiform classification and MeteoSwiss hydrometeor classification
Rainfall Rate retrieval	retrieve/qpe.py	Implementation of various rainfall rate retrieval algorithms
Wind retrievals	retrieve/vad.py retrieve/wind.py	Wind retrievals using VAD techniques and other


Thank you!
Grazie mille!
Moltes Gràcies!
Merci!

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

MeteoSchweiz
Operation Center 1
CH-8058 Zürich-Flughafen
T +41 58 460 91 11
www.meteoschweiz.ch

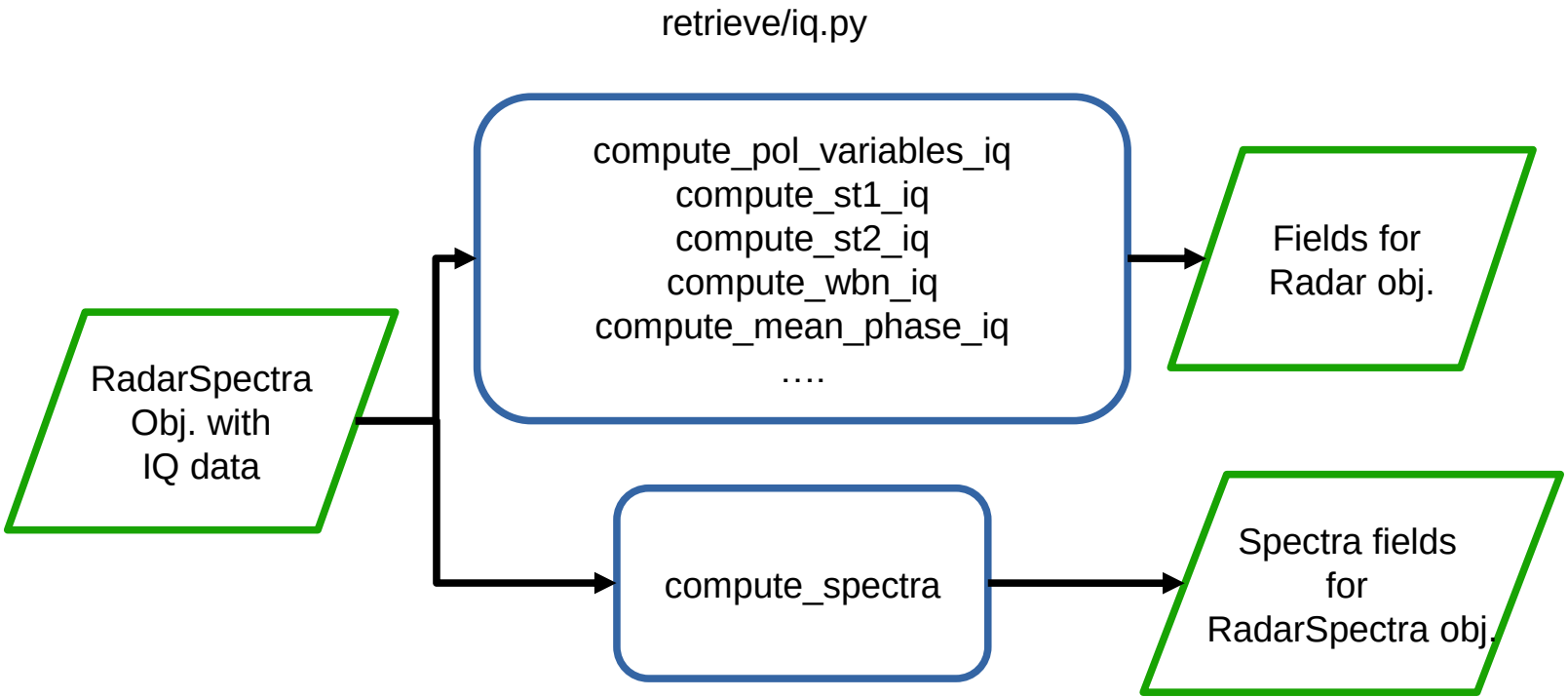
MeteoSvizzera
Via ai Monti 146
CH-6605 Locarno-Monti
T +41 58 460 92 22
www.meteosvizzera.ch

MétéoSuisse
7bis, av. de la Paix
CH-1211 Genève 2
T +41 58 460 98 88
www.meteosuisse.ch

MétéoSuisse
Chemin de l'Aérologie
CH-1530 Payerne
T +41 58 460 94 44
www.meteosuisse.ch

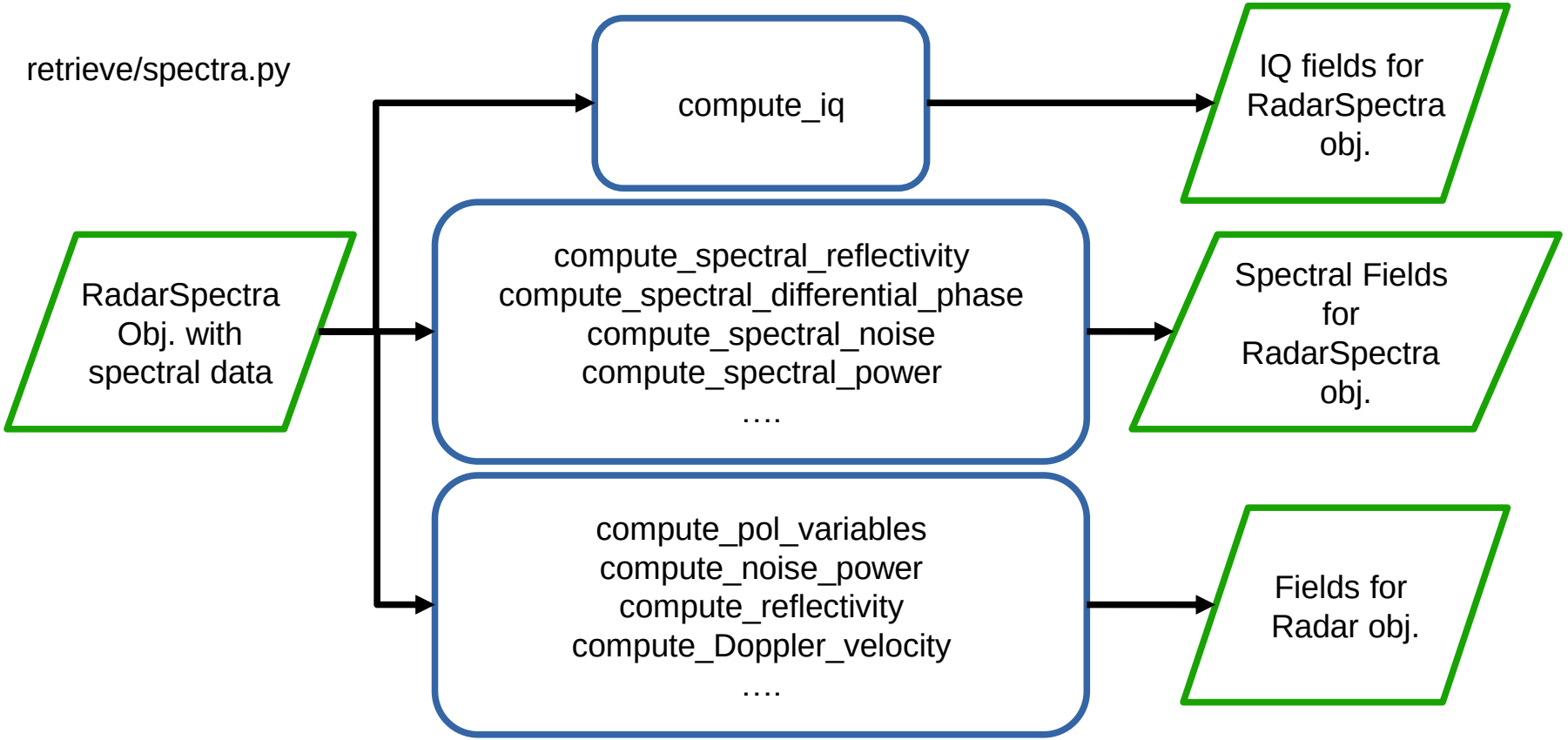
Appendix. Data processing details

IQ data

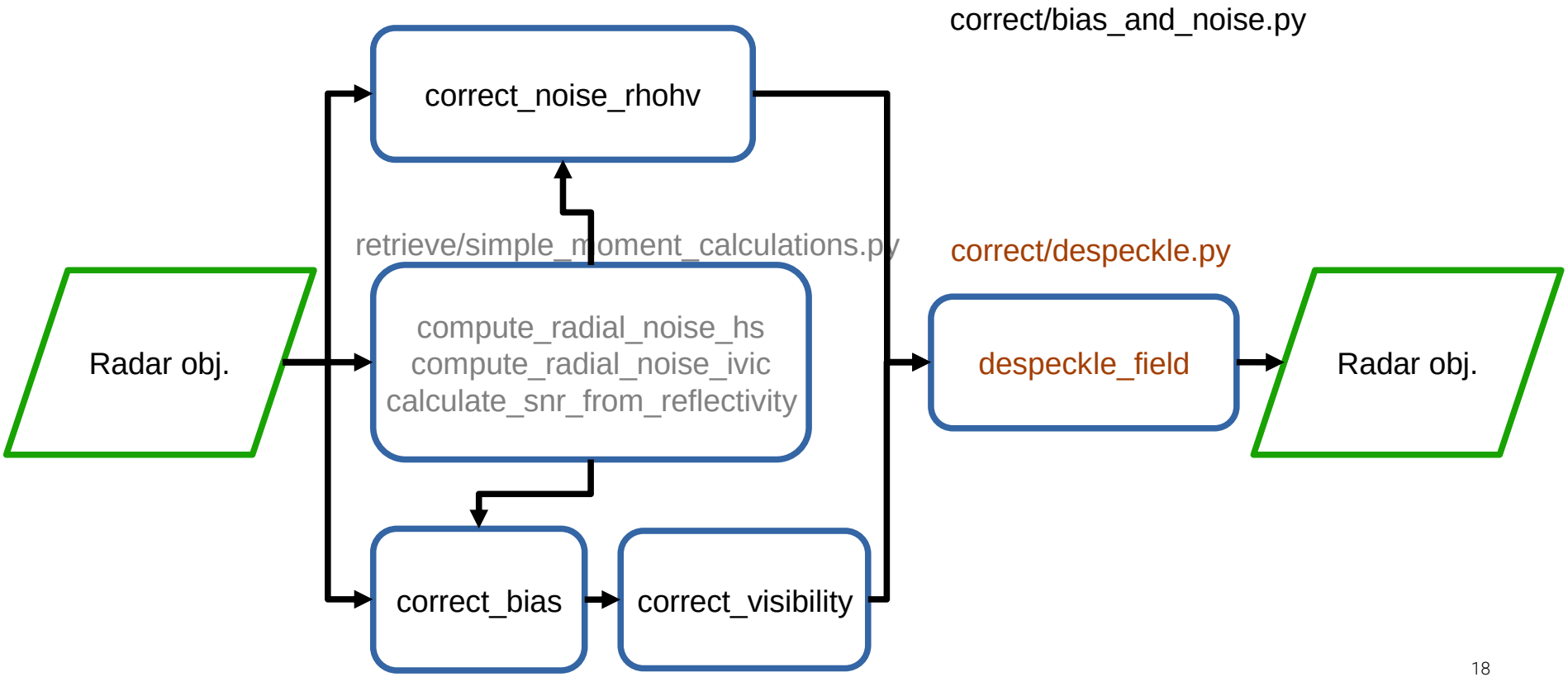

retrieve/iq.py
compute_pol_variables_iq
compute_st1_iq
compute_st2_iq
compute_wbn_iq
compute_mean_phase_iq
….
Fields for
Radar obj.
RadarSpectra
Obj. with
IQ data
compute_spectra
Spectra fields
for
RadarSpectra obj.

Spectral data

retrieve/spectra.py



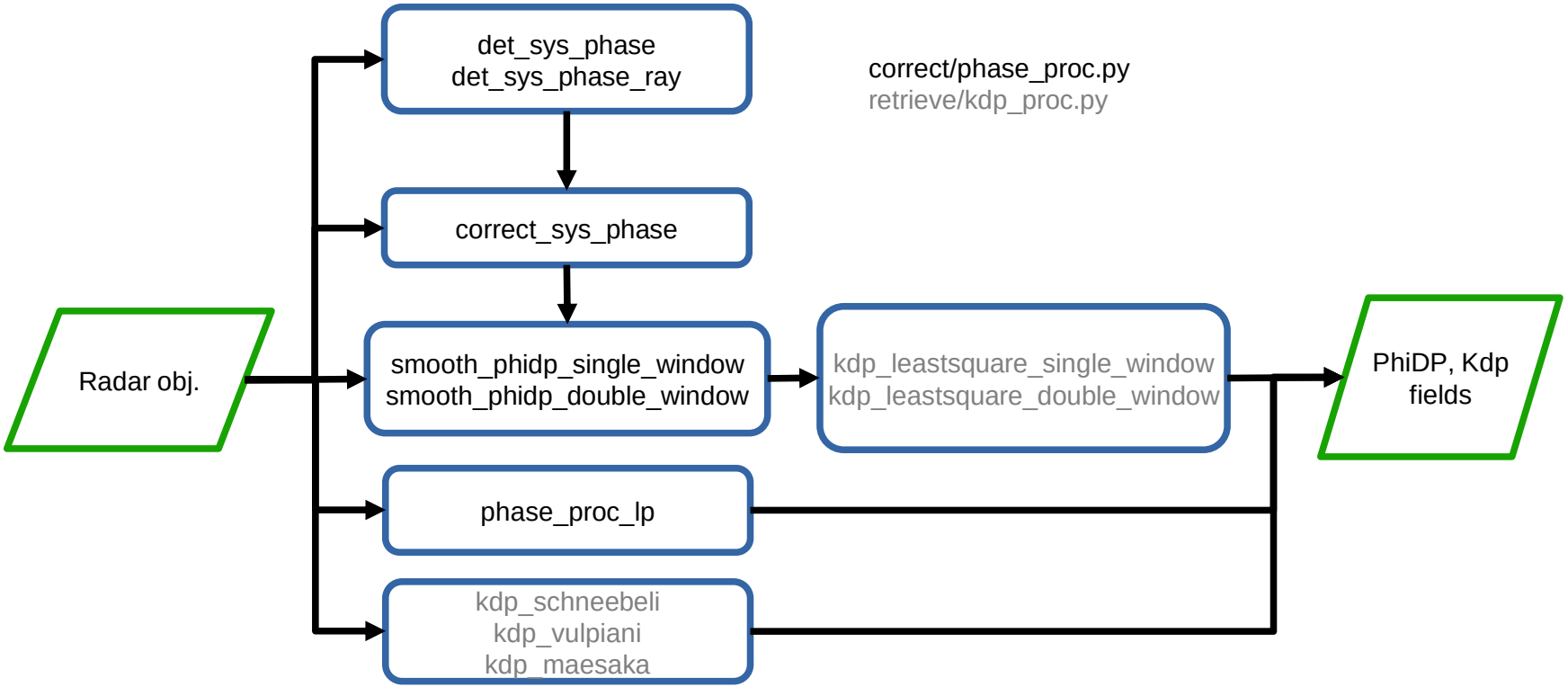
Noise and bias correction



Filtering undesired echoes

GateFilter	Purpose
moment_based_gate_filter	Masked based on thresholds on reflectivity and RhoHV
moment_and_texture_based_gate_filter	Masked based on thresholds on raw moments (dBZ, RhoHV, ZDR, PhiDP) and their textures
snr_based_gate_filter	Masked based on SNR threshold
class_based_gate_filter	Masked based on desired hydrometeors
visibility_based_gate_filter	Masked based on visibility threshold
temp_based_gate_filter	Masked based on temperature from and NWP model (removes non-liquid precipitation)
iso0_based_gate_filter	As above but using the height of the gate with respect to the iso-0° altitude
birds_gate_filter	Mask suspected bird echoes. Based on thresholds on moments and velocity

Raw PhiDP processing



Detect the melting layer

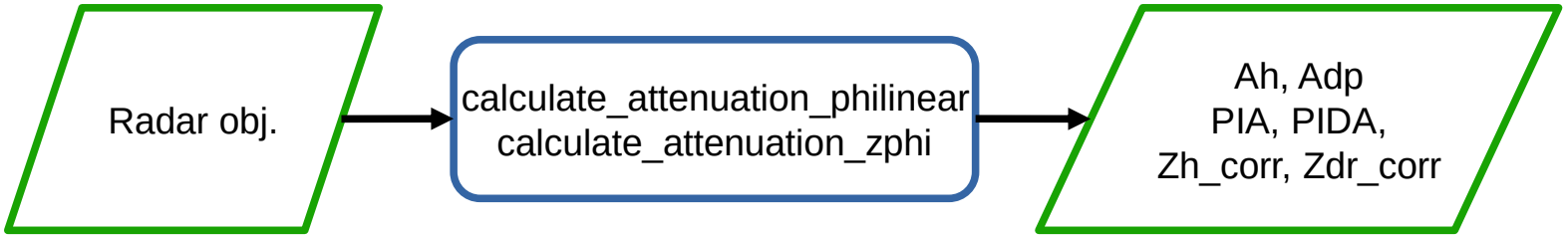
Function	Type
melting_layer_mf	Operational MF algorithm. Based on finding the theoretical RhoHV profile that best compares with the observed one. Only one profile per radar volume is found.
melting_layer_giangrande	Algorithm described in Giangrande et al. (2008). Captures the azimuthal variation of the melting layer
melting_layer_hydroclass	Melting layer is determined from the results of an hydrometeor classification given as input
detect_ml	Algorithm described in Wolfensberger et al. (2016). Uses RHIs or pseudo-RHIs. Needs good volumetric coverage

All provide:

- ml_dict: a field of flags indicating the position of the gate with respect to the melting layer
- ml_obj: a Radar-like object containing the top (range pos. 1) and bottom (range pos. 0) of the melting layer for each azimuth
- iso0_dict: a field with the altitude of the gate with respect to the iso-0° altitude (assuming iso-0° altitude=ml top)

Compute attenuation

correct/attenuation.py



Hydrometeor classification

Function	Purpose
steiner_conv_strat	Convective/stratiform determination following Steiner et al. (1995) algorithm
hydroclass_semisupervised	Semi-supervised hydrometeor classification described in Besic et al. (2016). Provides the dominant hydrometeor, the entropy and the proportion of each hydrometeor at each range gate

retrieve/echo_class.py

VPR correction

Function	Purpose
correct_vpr	Operational MF VPR algorithm described in Tabary 2007. Based on finding the theoretical VPR profile that best fits observations of ratios of reflectivity at different elevation angles. Only one profile per radar volume is obtained. Provides the corrected reflectivity, the correction applied and the theoretical VPR profile used in the correction.
correct_vpr_spatialised	As above but once the profile is obtained the correction applied to each range gate is adapted to the altitude of the gate with respect to the iso-0° altitude

correct/vpr.py

RR retrieval

Function	Retrieval
est_rain_rate_zpoly	Retrieve rainfall rate from reflectivity by applying a polynomial Z-R relation
est_rain_rate_z	Retrieval using a power law on Z
est_rain_rate_kdp	Retrieval using a power law on KDP
est_rain_rate_a	Retrieval using a power law on Ah
est_rain_rate_zkdp	Retrieval using Z or KDP depending on the rainfall intensity
est_rain_rate_za	Retrieval using Z or Ah depending on the rainfall intensity
est_rain_rate_hydro	Retrieval using estimates adapted to the dominant hydrometeor type at each range gate

retrieve/qpe.py

Velocity unfolding

Function	Retrieval
dealias_fourdd (correct/dealias.py)	De-aliasing using the 4DD algorithm described in James and Houze (2001)
dealias_region_based (correct/region_dealias.py)	De-aliasing using a region-based approach. Unfolding is performed by grouping regions with similar velocities and trying to determine which regions have to be unfolded by looking at neighbouring regions
dealias_unwrap_phase (correct/unwrap.py)	De-aliasing by using multi-dimensional phase unwrapping

Velocity retrievals

Function	Retrieval
vad_michelson (retrieve/vad.py)	VAD retrieval following Michelson et al. (2000) algorithm
vad_browning (retrieve/vad.py)	VAD retrieval following Browning and Wexler (1968) algorithm
est_wind_profile (retrieve/wind.py)	Another VAD retrieval
est_wind_vel (retrieve/wind.py)	Estimates wind velocity from Vr. Projects Vr into a horizontal plane (azimuthal horizontal wind) or a vertical plane (vertical wind component). Assumes the velocity in the orthogonal axis is negligible.
est_vertical_windshear (retrieve/wind.py)	Estimates wind shear from azimuthal horizontal wind

Auxiliary processing

Py-ART monitoring functions

correct/bias_and_noise.py

Function	Purpose
sun_retrieval	Estimate sun parameters from sun hits
get_sun_hits	Detect sun hits. Uses Hildebrand and Sekhon (1974) noise estimate
get_sun_hits_ivic	Detect sun hits. Uses Ivic (2013) noise estimate
get_sun_hits_psr	Detect sun hits. Uses the noise estimated from the Doppler spectra
est_rhohv_rain	Keeps data that can be used to determine the RhoHV in rain
est_zdr_precip	Keeps data that can be used to estimate the ZDR bias using either moderate rain or from a vertically pointing scan
est_zdr_snow	Keeps data that can be used to estimate the ZDR bias using measurements in snow
selfconsistency_bias	Estimates reflectivity bias at each ray using the self-consistency algorithm by Gourley

Py-ART DEM processing

Py-ART can provide parameters useful in radar data processing from a DEM.

e.g. Expected RCS from ground clutter, Expected dBm from ground clutter, Expected dBZ from froung clutter, visibility

retrieve/gecsx.py

Py-ART QVP family

retrieve/qvp.py

Function	Purpose
compute_qvp	Quasi Vertical Profile
compute_rqvp	Range-defined Quasi Vertical Profile
compute_evp	Enhanced Vertical Profile (non radar centric)
compute_svp	Slanted Vertical Profile (non radar centric)
compute_vp	Compute Vertical Profile at a given location
compute_ts_along_coord	Computes Time Series along one of the radar coordinates (rng, azi or ele.)

- Output is stored in a radar-like object where each ray represents a time step

Mapping into a grid

- Function grid_from_radars in map/grid_mapper.py