pyart-mch library reference for users

Release 0.1.0

meteoswiss-mdr

Jan 27, 2020

CONTENTS

Contents:

CHAPTER

ONE

INPUT AND OUTPUT (PYART.IO)

Functions to read and write radar and grid data to and from a number of file formats.

1.1 Reading radar data

In most cases the *pyart.io.read()* function should be used to read in radar data from a file. In certain cases the function the read function for the format in question should be used.

read(filename[, use_rsl])	Read a radar file and return a radar object.
read_rsl(filename[, field_names, ...])	Read a file supported by RSL.
read_mdv(filename[, field_names, ...])	Read a MDV file.
read_sigmet(filename[, field_names, ...])	Read a Sigmet (IRIS) product file.
read_cfradial(filename[, field_names, ...])	Read a Cfradial netCDF file.
read_cfradial2(filename[, field_names, ...])	Read a Cfradial2 netCDF file.
read_chl(filename[, field_names, ...])	Read a CSU-CHILL CHL file.
read_nexrad_archive(filename[, field_names, ...])	Read a NEXRAD Level 2 Archive file.
read_nexrad_cdm(filename[, field_names, ...])	Read a Common Data Model (CDM) NEXRAD Level 2 file.
read_nexrad_level3(filename[, field_names, ...])	Read a NEXRAD Level 3 product.
read_uf(filename[, field_names, ...])	Read a UF File.

1.2 Writing radar data

write_cfradial(filename, radar[, format, ...])	Write a Radar object to a CF/Radial compliant netCDF file.
write_uf(filename, radar[, uf_field_names, ...])	Write a Radar object to a UF file.

1.3 Reading grid data

read_grid(filename[, exclude_fields, ...])	Read a netCDF grid file produced by Py-ART.
read_grid_mdv(filename[, field_names, ...])	Read a MDV file to a Grid Object.

1.4 Writing grid data

write_grid(filename, grid[, format, ...])	Write a Grid object to a CF-1.5 and ARM standard netCDF file.
write_grid_mdv(filename, grid[, ...])	Write grid object to MDV file.
write_grid_geotiff(grid, filename, field[, ...])	Write a Py-ART Grid object to a GeoTIFF file.

1.5 Reading Sonde data

read_arm_sonde(filename)	Read a ARM sonde file returning a wind profile.
read_arm_sonde_vap(filename[, radar, ...])	Read a ARM interpolated or merged sonde returning a wind profile.

1.6 Special use

prepare_for_read(filename)	Return a file like object read for reading.
make_time_unit_str(dtobj)	Return a time unit string from a datetime object.

pyart.io.**make_time_unit_str**(*dtobj*)
Return a time unit string from a datetime object.

pyart.io.**prepare_for_read**(*filename*)
Return a file like object read for reading.

Open a file for reading in binary mode with transparent decompression of Gzip and BZip2 files. The resulting file-like object should be closed.

Parameters

filename [str or file-like object] Filename or file-like object which will be opened. File-like objects will not be examined for compressed data.

Returns

file_like [file-like object] File like object from which data can be read.

pyart.io.**read**(*filename*, *use_rsl=False*, ***kwargs*)
Read a radar file and return a radar object.

Additional parameters are passed to the underlying read_* function.

Parameters

filename [str] Name of radar file to read.

use_rsl [bool] True will use the TRMM RSL library to read files which are supported both natively and by RSL. False will choose the native read function. RSL will always be used to read a file if it is not supported natively.

Returns

radar [Radar] Radar object. A TypeError is raised if the format cannot be determined.

Other Parameters

field_names [dict, optional] Dictionary mapping file data type names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the metadata configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the metadata configuration file will be used.

file_field_names [bool, optional] True to use the file data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters.

delay_field_loading [bool] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects. Not all file types support this parameter.

pyart.io.**read_arm_sonde**(*filename*)

Read a ARM sonde file returning a wind profile.

Parameters

filename [str] Name of ARM sonde NetCDF file to read data from.

pyart.io.**read_arm_sonde_vap**(*filename*, *radar=None*, *target_datetime=None*)

Read a ARM interpolated or merged sonde returning a wind profile.

Parameters

filename [str] Name of ARM interpolate or merged sonde NetCDF file to read data from.

radar [Radar, optional] If provided the profile returned is that which is closest in time to the first ray collected in this radar. Either radar or target_datetime must be provided.

target_datetime [datetime, optional] If specified the profile returned is that which is closest in time to this datetime.

pyart.io.**read_cfradial**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *delay_field_loading=False*, ***kwargs*)

Read a Cfradial netCDF file.

Parameters

filename [str] Name of CF/Radial netCDF file to read data from.

field_names [dict, optional] Dictionary mapping field names in the file names to radar field names. Unlike other read functions, fields not in this dictionary or having a value of None are still included in the radar.fields dictionary, to exclude them use the *exclude_fields* parameter. Fields which are mapped by this dictionary will be renamed from key to value.

additional_metadata [dict of dicts, optional] This parameter is not used, it is included for uniformity.

file_field_names [bool, optional] True to force the use of the field names from the file in which case the *field_names* parameter is ignored. False will use to *field_names* parameter to rename fields.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

delay_field_loading [bool] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects. Delayed field loading will not provide any speedup in file where the number of gates vary between rays (ngates_vary=True) and is not recommended.

Returns

radar [Radar] Radar object.

Notes

This function has not been tested on "stream" Cfradial files.

pyart.io.**read_cfradial2**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *delay_field_loading=False*, ***kwargs*)

Read a Cfradial2 netCDF file.

Parameters

filename [str] Name of CF/Radial netCDF file to read data from.

field_names [dict, optional] Dictionary mapping field names in the file names to radar field names. Unlike other read functions, fields not in this dictionary or having a value of None are still included in the radar.fields dictionary, to exclude them use the *exclude_fields* parameter. Fields which are mapped by this dictionary will be renamed from key to value.

additional_metadata [dict of dicts, optional] This parameter is not used, it is included for uniformity.

file_field_names [bool, optional] True to force the use of the field names from the file in which case the *field_names* parameter is ignored. False will use to *field_names* parameter to rename fields.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

delay_field_loading [bool] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects. Delayed field loading will not provide any speedup in file where the number of gates vary between rays (ngates_vary=True) and is not recommended.

Returns

radar [Radar] Radar object.

Notes

This function has not been tested on "stream" Cfradial files.

pyart.io.**read_chl**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=None*, *exclude_fields=None*, *include_fields=None*, *use_file_field_attributes=True*, ***kwargs*)

Read a CSU-CHILL CHL file.

Parameters

filename [str] Name of CHL file.

field_names [dict, optional] Dictionary mapping CHL field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the CHL field names for the field names in the radar object. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *field_file_names* and *field_names* parameters. Set to None to include all fields not in exclude_fields.

use_file_field_attributes [bool, optional] True to use information provided by in the file to set the field attribute *long_name*, *units*, *valid_max*, and *valid_min*. False will not set these unless they are defined in the configuration file or in *additional_metadata*.

Returns

radar [Radar] Radar object containing data from CHL file.

pyart.io.**read_grid**(*filename*, *exclude_fields=None*, *include_fields=None*, ***kwargs*)

Read a netCDF grid file produced by Py-ART.

Parameters

filename [str] Filename of netCDF grid file to read. This file must have been produced by *write_grid()* or have identical layout.

Returns

grid [Grid] Grid object containing gridded data.

Other Parameters

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

pyart.io.**read_grid_mdv**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *delay_field_loading=False*, ***kwargs*)

Read a MDV file to a Grid Object.

Parameters

filename [str] Name of MDV file to read or file-like object pointing to the beginning of such a file.

field_names [dict, optional] Dictionary mapping MDV data type names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the grid object. This is applied after the *file_field_names* and *field_names* parameters.

delay_field_loading [bool] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects.

Returns

grid [Grid] Grid object containing data from MDV file.

Notes

This function can only read cartesian MDV files with fields compressed with gzip or zlib. For polar files see *pyart.io.read_mdv()*

MDV files and Grid object are not fully interchangeable. Specific limitation include:

- All fields must have the same shape and dimensions.
- All fields must have the same projection.
- Vlevels types must not vary.
- Projection must not be PROJ_POLAR_RADAR (9) or PROJ_RHI_RADAR (13).
- Correct unit in the Z axis are just availible for 'vlevel_type' equal to VERT_TYPE_Z(4), VERT_TYPE_ELEV(9), VERT_TYPE_AZ(17), VERT_TYPE_PRESSURE(3) and VERT_TYPE_THETA(7).
- The behavior in cases of 2D data is unknown but most likely will not fail.

pyart.io.**read_mdv**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *delay_field_loading=False*, ***kwargs*)

Read a MDV file.

Parameters

filename [str] Name of MDV file to read or file-like object pointing to the beginning of such a file.

field_names [dict, optional] Dictionary mapping MDV data type names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

delay_field_loading [bool] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects. Not all file types support this parameter.

Returns

radar [Radar] Radar object containing data from MDV file.

Notes

Currently this function can only read polar MDV files with fields compressed with gzip or zlib.

pyart.io.**read_nexrad_archive**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *delay_field_loading=False*, *station=None*, *scans=None*, *linear_interp=True*, ***kwargs*)

Read a NEXRAD Level 2 Archive file.

Parameters

filename [str] Filename of NEXRAD Level 2 Archive file. The files hosted by at the NOAA National Climate Data Center [1] as well as on the UCAR THREDDS Data Server [2] have been tested. Other NEXRAD Level 2 Archive files may or may not work. Message type 1 file and message type 31 files are supported.

field_names [dict, optional] Dictionary mapping NEXRAD moments to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the metadata configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata

and the file specific or default metadata as specified by the metadata configuration file will be used.

file_field_names [bool, optional] True to use the NEXRAD field names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

delay_field_loading [bool, optional] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects.

station [str or None, optional] Four letter ICAO name of the NEXRAD station used to determine the location in the returned radar object. This parameter is only used when the location is not contained in the file, which occur in older NEXRAD message 1 files.

scans [list or None, optional] Read only specified scans from the file. None (the default) will read all scans.

linear_interp [bool, optional] True (the default) to perform linear interpolation between valid pairs of gates in low resolution rays in files mixed resolution rays. False will perform a nearest neighbor interpolation. This parameter is not used if the resolution of all rays in the file or requested sweeps is constant.

Returns

radar [Radar] Radar object containing all moments and sweeps/cuts in the volume. Gates not collected are masked in the field data.

References

[1], [2]

pyart.io.**read_nexrad_cdm**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *station=None*, ***kwargs*)

Read a Common Data Model (CDM) NEXRAD Level 2 file.

Parameters

filename [str] File name or URL of a Common Data Model (CDM) NEXRAD Level 2 file. File of in this format can be created using the NetCDF Java Library tools [1]. A URL of a OPeNDAP file on the UCAR THREDDS Data Server [2] is also accepted the netCDF4 library has been compiled with OPeNDAP support.

field_names [dict, optional] Dictionary mapping NEXRAD moments to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the metadata configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata

and the file specific or default metadata as specified by the metadata configuration file will be used.

file_field_names [bool, optional] True to use the NEXRAD field names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

station [str] Four letter ICAO name of the NEXRAD station used to determine the location in the returned radar object. This parameter is only used when the location is not contained in the file, which occur in older NEXRAD files. If the location is not provided in the file and this parameter is set to None the station name will be determined from the filename.

Returns

radar [Radar] Radar object containing all moments and sweeps/cuts in the volume. Gates not collected are masked in the field data.

References

[1], [2]

pyart.io.**read_nexrad_level3**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, ***kwargs*)

Read a NEXRAD Level 3 product.

Parameters

filename [str] Filename of NEXRAD Level 3 product file. The files hosted by at the NOAA National Climate Data Center [1] as well as on the NWS WSR-88D Level III Data Collection and Distribution Network have been tests. Other NEXRAD Level 3 files may or may not work. A file-like object pointing to the beginning of such a file is also supported.

field_names [dict, optional] Dictionary mapping NEXRAD level 3 product number to radar field names. If the product number of the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the metadata configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the metadata configuration file will be used.

file_field_names [bool, optional] True to use the product number for the field name. In this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

Returns

radar [Radar] Radar object containing all moments and sweeps/cuts in the volume. Gates not collected are masked in the field data.

References

[1], [2]

pyart.io.**read_rsl**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *delay_field_loading=False*, *include_fields=None*, *radar_format=None*, *callid=None*, *skip_range_check=False*)

Read a file supported by RSL.

Parameters

filename [str or RSL_radar] Name of file whose format is supported by RSL.

field_names [dict, optional] Dictionary mapping RSL data type names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the RSL data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

delay_field_loading [bool] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects.

radar_format [str or None] Format of the radar file. Must be 'wsr88d' or None.

callid [str or None] Four letter NEXRAD radar Call ID, only used when radar_format is 'wsr88d'.

skip_range_check [bool, optional] True to skip check for uniform range bin location, the reported range locations will only be verified true for the first ray. False will perform the check and raise a IOError when the locations of the gates change between rays.

Returns

radar [Radar] Radar object.

pyart.io.**read_sigmet**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *time_ordered='none'*, *full_xhdr=None*, *noaa_hh_hdr=None*, *debug=False*, *ignore_xhdr=False*, *ignore_sweep_start_ms=None*, ***kwargs*)

Read a Sigmet (IRIS) product file.

Parameters

filename [str] Name of Sigmet (IRIS) product file to read or file-like object pointing to the beginning of such a file.

field_names [dict, optional] Dictionary mapping Sigmet data type names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the metadata configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the metadata configuration file will be used.

file_field_names [bool, optional] True to use the Sigmet data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

time_ordered ['none', 'sequential', 'full', ..., optional] Parameter controlling if and how the rays are re-ordered by time. The default, 'none' keeps the rays ordered in the same manner as they appears in the Sigmet file. 'sequential' will determind and apply an operation which maintains a sequential ray order in elevation or azimuth yet orders the rays according to time. If no operation can be found to accomplish this a warning is issue and the rays are returned in their original order. 'roll', 'reverse', and 'reverse_and_roll' will apply that operation to the rays in order to place them in time order, direct use of these is not recommended. 'full' will order the rays in strictly time increasing order, but the rays will likely become non-sequential, thisoption is not recommended unless strict time increasing order is required.

full_xhdr [bool or None] Flag to read in all extended headers for possible decoding. None will determine if extended headers should be read in automatically by examining the extended header type.

noaa_hh_hdr [bool or None] Flag indicating if the extended header should be decoded as those used by the NOAA Hurricane Hunters aircraft radars. None will determine if the extended header is of this type automatically by examining the header. The *full_xhdr* parameter is set to True when this parameter is True.

ignore_xhdr [bool, optional] True to ignore all data in the extended headers if they exist. False, the default, extracts milliseconds precision times and other parameter from the extended headers if they exists in the file.

ignore_sweep_start_ms [bool or None, optional] True to ignore the millisecond parameter in the start time for each sweep, False will uses this parameter when determining the timing of each ray. None, the default, will ignore the millisecond sweep start timing only when the file

does not contain extended headers or when the extended header has been explicity ignored using the *ignore_xhdr* parameter. The TRMM RSL library ignores these times so setting this parameter to True is required to match the times determined when reading Sigmet files with `pyart.io.read_rsl()`. When there are not extended headers ignoring the millisecond sweep times provides time data which is always prior to the actual collection time with an error from 0 to 2 seconds.

debug [bool, optional] Print debug information during read.

Returns

radar [Radar] Radar object.

pyart.io.**read_uf**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *delay_field_loading=False*, ***kwargs*)

Read a UF File.

Parameters

filename [str or file-like] Name of Universal format file to read data from.

field_names [dict, optional] Dictionary mapping UF data type names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduce any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to force the use of the field names from the file in which case the *field_names* parameter is ignored. False will use to *field_names* parameter to rename fields.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

delay_field_loading [bool] This option is not implemented in the function but included for compatibility.

Returns

radar [Radar] Radar object.

pyart.io.**write_cfradial**(*filename*, *radar*, *format='NETCDF4'*, *time_reference=None*, *arm_time_variables=False*, *physical=True*)

Write a Radar object to a CF/Radial compliant netCDF file.

The files produced by this routine follow the CF/Radial standard. Attempts are also made to to meet many of the standards outlined in the ARM Data File Standards.

To control how the netCDF variables are created, set any of the following keys in the radar attribute dictionaries.

- _Zlib
- _DeflateLevel

- _Shuffle
- _Fletcher32
- _Continguous
- _ChunkSizes
- _Endianness
- _Least_significant_digit
- _FillValue

See the netCDF4 documentation for details on these settings.

Parameters

filename [str] Filename to create.

radar [Radar] Radar object.

format [str, optional] NetCDF format, one of 'NETCDF4', 'NETCDF4_CLASSIC', 'NETCDF3_CLASSIC' or 'NETCDF3_64BIT'. See netCDF4 documentation for details.

time_reference [bool] True to include a time_reference variable, False will not include this variable. The default, None, will include the time_reference variable when the first time value is non-zero.

arm_time_variables [bool] True to create the ARM standard time variables base_time and time_offset, False will not create these variables.

physical [bool] True to store the radar fields as physical numbers, False will store the radar fields as binary if the keyword '_Write_as_dtype' is in the field metadata. The gain and offset can be specified in the keyword 'scale_factor' and 'add_offset' or calculated on the fly.

pyart.io.**write_grid**(*filename*, *grid*, *format='NETCDF4'*, *write_proj_coord_sys=True*, *proj_coord_sys=None*, *arm_time_variables=False*, *arm_alt_lat_lon_variables=False*, *write_point_x_y_z=False*, *write_point_lon_lat_alt=False*)

Write a Grid object to a CF-1.5 and ARM standard netCDF file.

To control how the netCDF variables are created, set any of the following keys in the grid attribute dictionaries.

- _Zlib
- _DeflateLevel
- _Shuffle
- _Fletcher32
- _Continguous
- _ChunkSizes
- _Endianness
- _Least_significant_digit
- _FillValue

See the netCDF4 documentation for details on these settings.

Parameters

filename [str] Filename to save grid to.

grid [Grid] Grid object to write.

format [str, optional] netCDF format, one of 'NETCDF4', 'NETCDF4_CLASSIC', 'NETCDF3_CLASSIC' or 'NETCDF3_64BIT'. See netCDF4 documentation for details.

write_proj_coord_sys bool, optional True to write information on the coordinate transform used in the map projection to the ProjectionCoordinateSystem variable following the CDM Object Model. The resulting file should be interpreted as containing geographic grids by tools which use the Java NetCDF library (THREDDS, toolsUI, etc).

proj_coord_sys [dict or None, optional] Dictionary of parameters which will be written to the ProjectionCoordinateSystem NetCDF variable if write_proj_coord_sys is True. A value of None will attempt to generate an appropriate dictionary by examining the projection attribute of the grid object. If the projection is not understood a warnings will be issued.

arm_time_variables [bool, optional] True to write the ARM standard time variables base_time and time_offset. False will not write these variables.

arm_alt_lat_lon_variables [bool, optional] True to write the ARM standard alt, lat, lon variables. False will not write these variables.

write_point_x_y_z [bool, optional] True to include the point_x, point_y and point_z variables in the written file, False will not write these variables.

write_point_lon_lat_alt [bool, optional] True to include the point_longitude, point_latitude and point_altitude variables in the written file, False will not write these variables.

pyart.io.**write_grid_geotiff**(*grid*, *filename*, *field*, *rgb=False*, *level=None*, *cmap='viridis'*, *vmin=0*, *vmax=75*, *color_levels=None*, *warp=False*, *sld=False*, *use_doublequotes=False*)

Write a Py-ART Grid object to a GeoTIFF file.

The GeoTIFF can be the standard Azimuthal Equidistant projection used in Py-ART, or a lat/lon projection on a WGS84 sphere. The latter is typically more usable in web mapping applications. The GeoTIFF can contain a single float-point raster band, or three RGB byte raster bands. The former will require an SLD file for colorful display using standard GIS or web mapping software, while the latter will show colors "out-of-the-box" but lack actual data values. The function also can output an SLD file based on the user-specified inputs. User can specify the 2D vertical level to be output. If this is not specified, a 2D composite is created. User also can specify the field to output.

This function requires GDAL Python libraries to be installed. These are available via conda; e.g., 'conda install gdal'

Parameters

grid [pyart.core.Grid object] Grid object to write to file.

filename [str] Filename for the GeoTIFF.

field [str] Field name to output to file.

Other Parameters

rbg [bool, optional] True - Output 3-band RGB GeoTIFF

False - Output single-channel, float-valued GeoTIFF. For display, likely will need an SLD file to provide a color table.

level [int or None, optional] Index for z-axis plane to output. None gives composite values (i.e., max in each vertical column).

cmap [str or matplotlib.colors.Colormap object, optional] Colormap to use for RGB output or SLD file.

vmin [int or float, optional] Minimum value to color for RGB output or SLD file.

vmax [int or float, optional] Maximum value to color for RGB output or SLD file.

color_levels [int or None, optional] Number of color levels in cmap. Useful for categorical colormaps with steps << 255 (e.g., hydrometeor ID).

warp [bool, optional]

True - Use gdalwarp (called from command line using os.system) to warp to a lat/lon WGS84 grid.

False - No warping will be performed. Output will be Az. Equidistant.

sld [bool, optional]

True - Create a Style Layer Descriptor file (SLD) mapped to vmin/vmax and cmap. File is named same as output TIFF, except for .sld extension.

False - Don't do this.

use_doublequotes [bool, optional]

True - Use double quotes in the gdalwarp call (requires warp=True), which may help if that command is producing and error like: 'Translating source or target SRS failed'.

False - Use single quotes instead.

pyart.io.**write_grid_mdv**(*filename*, *grid*, *mdv_field_names=None*, *field_write_order=None*)

Write grid object to MDV file.

Create a MDV file containing data from the provided grid instance.

The MDV file will contain parameters from the 'source' key if contained in grid.metadata. If this key or parameters related to the radar location and name are not present in the grid a default or sentinel value. will be written in the MDV file in the place of the parameter.

Grid fields will be saved in float32 unless the *_Write_as_dtype* key is present.

Parameters

filename [str or file-like object.] Filename of MDV file to create. If a file-like object is specified data will be written using the write method.

grid [Grid] Grid object from which to create MDV file.

mdv_field_names [dict or None, optional] Mapping between grid fields and MDV data type names. Field names mapped to None or with no mapping will be excluded from writing. If None, the same field names will be used.

field_write_order [list or None, optional] Order in which grid fields should be written out in the MDV file. None, the default, will determine a valid order automatically.

Notes

Do to limitations of the MDV format, not all grid objects are writable. To write a grid the following conditions must be satisfied:

- XY grid must be regular (equal spacing), Z can be irregular.
- The number of Z levels must not exceed 122.
- Fields can be encoded in the file using the '_Write_as_dtype' key specifying one of 'uint8', 'uint16' or 'float32'. Use the 'scale_factor' and 'add_offset' keys to specify scaling. Field data in the Grid object should be uncompressed, that is to say it has had the scaling applied.

pyart.io.**write_uf**(*filename*, *radar*, *uf_field_names=None*, *radar_field_names=False*, *exclude_fields=None*, *field_write_order=None*, *volume_start=None*, *templates_extra=None*)

Write a Radar object to a UF file.

Create a UF file containing data from the provided radar instance. The UF file will contain instrument parameters from the following dictionaries if they contained in radar.instrument_parameters:

- radar_beam_width_h
- radar_beam_width_v
- radar_receiver_bandwidth
- frequency
- pulse_width
- prt
- polarization_mode
- nyquist_velocity

If any of these parameter are not present a default or sentinel value will be written in the UF file in the place of the parameter. This is also true for the data in the scan_rate attribute.

Radar fields will be scaled and rounded to integer values when writing to UF files. The scale factor for each field can be specified in the *_UF_scale_factor* key for each field dictionary. If not specified the default scaling (100) will be used.

Parameters

filename [str or file-like object.] Filename of UF file to create. If a file-like object is specified data will be written using the write method.

radar [Radar] Radar object from which to create UF file.

uf_field_names [dict or None, optional] Mapping between radar fields and two character UF data type names. Field names mapped to None or with no mapping will be excluded from writing. If None, the default mappings for UF files will be used.

radar_field_names [bool, optional] True to use the radar field names as the field names of the UF fields. False to use the uf_field_names mapping to generate UF field names. The *exclude_fields* argument can still be used to exclude fields from the UF file when this parameter is True. When reading a UF file using *file_field_names=True* set this parameter to True to write a UF file with the same field names.

exclude_fields [list or None, optional] List of radar fields to exclude from writing.

field_write_order [list or None, optional] Order in which radar fields should be written out in the UF file. None, the default, will determine a valid order automatically.

volume_start [datetime, optional] Start of volume used to set UF volume structure elements.

templates_extra [dict of dict or None] Advanced usage parameter for setting UF structure templates. Elements defined in dictionaries with keys 'mandatory_header', 'optional_header', and 'field_header' will be used to build the structure template.

CHAPTER

TWO

AUXILIARY INPUT AND OUTPUT (PYART.AUX_IO)

Additional classes and functions for reading and writing data from a number of file formats.

These auxiliary input/output routines are not as well polished as those in *pyart.io*. They may require addition dependencies beyond those required for a standard Py-ART install, use non-standard function parameter and naming, are not supported by the *pyart.io.read()* function and are not fully tested if tested at all. Please use these at your own risk.

Bugs in these function should be reported but fixing them may not be a priority.

2.1 Reading radar data

read_d3r_gcpex_nc(filename[, field_names, ...])	Read a D3R GCPEX netCDF file.
read_gamic(filename[, field_names, ...])	Read a GAMIC hdf5 file.
read_kazr(filename[, field_names, ...])	Read K-band ARM Zenith Radar (KAZR) NetCDF ingest data.
read_noxp_iphex_nc(filename[, field_names, ...])	Read a NOXP IPHEX netCDF file.
read_odim_h5(filename[, field_names, ...])	Read a ODIM_H5 file.
read_pattern(filename, **kwargs)	Read a netCDF file from a PATTERN project X-band radar.
read_radx(filename[, radx_dir])	Read a file by first converting it to Cf/Radial using RadxConvert.
read_rainbow_wrl(filename[, field_names, ...])	Read a RAINBOW file.
read_metranet(filename[, field_names, rmax, ...])	Read a METRANET file.
read_cartesian_metranet(filename[, ...])	Read a METRANET product file.
read_gif(filename[, additional_metadata, ...])	Read a MeteoSwiss operational radar data gif file.
read_bin(filename[, additional_metadata, ...])	Read a MeteoSwiss operational radar data binary file.
read_iq(filename, filenames_iq[, ...])	Read a rad4alp IQ file.
read_rainbow_psr(filename, filenames_psr[, ...])	Read a PSR file.
read_rainbow_psr_spectra(filename, filenames_psr)	Read a PSR file to get the complex spectra
read_spectra(filename[, field_names, ...])	Read a spectra netCDF file.
read_cf1(filename[, field_names, ...])	Read a CF-1 netCDF file.

2.2 Writing radar data

write_odim_h5(filename, radar[, ...])	Write a Radar object to a EUMETNET OPERA compliant HDF5 file.
write_spectra(filename, radar[, format, ...])	Write a Radar Spectra object to a netCDF file.

pyart.aux_io.**convert_data**(*values*)

Converts an string of values into the corresponding format

Parameters

values: str string containg the values to convert

Returns

values [int, float, str or 1D array of int, float or str] The converted values

pyart.aux_io.**get_library**(*verbose=False*, *momentms=True*)

return the link to C-shared library

Parameters

verbose [Boolean] If true print out extra information

momentsms [Boolean] If true returns the link to the MS library

Returns

metranet_lib [link] loaded METRANET C-library

pyart.aux_io.**read_bin**(*filename*, *additional_metadata=None*, *chy0=255.0*, *chx0=-160.0*, *xres=1.0*, *yres=1.0*, *nx=710*, *ny=640*, *nz=1*, ***kwargs*)

Read a MeteoSwiss operational radar data binary file.

Parameters

filename [str] Name of the file to read.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

chy0, chx0 [float] Swiss coordinates position of the south-western point in the grid

xres, yres [float] resolution of each grid point [km]

nx, ny, nz [int] dimensions of the grid

Returns

grid [Grid] Grid object containing data the data.

pyart.aux_io.**read_cartesian_metranet**(*filename*, *additional_metadata=None*, *chy0=255.0*, *chx0=-160.0*, *reader='C'*, ***kwargs*)

Read a METRANET product file.

Parameters

filename [str] Name of the METRANET file to read.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

chy0, chx0 [float] Swiss coordinates position of the south-western point in the grid

reader [str] The reader library to use. Can be either 'C' or 'python'

Returns

grid [Grid] Grid object containing data from METRANET file.

pyart.aux_io.**read_cf1**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *delay_field_loading=False*, ***kwargs*)

Read a CF-1 netCDF file.

Parameters

filename [str] Name of CF/Radial netCDF file to read data from.

field_names [dict, optional] Dictionary mapping field names in the file names to radar field names. Unlike other read functions, fields not in this dictionary or having a value of None are still included in the radar.fields dictionary, to exclude them use the *exclude_fields* parameter. Fields which are mapped by this dictionary will be renamed from key to value.

additional_metadata [dict of dicts, optional] This parameter is not used, it is included for uniformity.

file_field_names [bool, optional] True to force the use of the field names from the file in which case the *field_names* parameter is ignored. False will use to *field_names* parameter to rename fields.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

delay_field_loading [bool] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects. Delayed field loading will not provide any speedup in file where the number of gates vary between rays (ngates_vary=True) and is not recommended.

Returns

radar [Radar] Radar object.

Notes

This function has not been tested on "stream" Cfradial files.

pyart.aux_io.**read_d3r_gcpex_nc**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *read_altitude_from_nc=False*, ***kwargs*)

Read a D3R GCPEX netCDF file.

Parameters

filename [str] Name of the ODIM_H5 file to read.

field_names [dict, optional] Dictionary mapping ODIM_H5 field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

read_altitude_from_nc [bool, optional] True if you want the altitude value to be read from the provider netCDF file. False will default to the value np.array([295.], dtype='float64')

Returns

radar [Radar] Radar object containing data from ODIM_H5 file.

pyart.aux_io.**read_edge_netcdf**(*filename*, ***kwargs*)

Read a EDGE NetCDF file.

Parameters

filename [str] Name of EDGE NetCDF file to read data from.

Returns

radar [Radar] Radar object.

pyart.aux_io.**read_file_c**(*file*, *moment='ZH'*, *physic_value=False*, *masked_array=False*, *verbose=False*)

Reads a METRANET data file

Parameters

file [str] file name

moment [str] moment name

physic_value [boolean] If true returns the physical value. Otherwise the digital value

masked_array [boolean] If true returns a numpy masked array with NaN values masked. Otherwise returns a regular masked array with NaN values

verbose [boolean] If true prints out extra information

Returns

ret_data [RadarData object] An object containing the information read from the file

pyart.aux_io.**read_file_py**(*file*, *moment='ZH'*, *physic_value=False*, *masked_array=False*, *reorder_angles=True*, *verbose=False*)

Reads a METRANET data file

Parameters

file [str] file name

moment [str] moment name

physic_value [boolean] If true returns the physical value. Otherwise the digital value

masked_array [boolean] If true returns a numpy masked array with NaN values masked. Otherwise returns a regular masked array with NaN values

reorder_angles [boolean] If true angles are reordered

verbose [boolean] If true prints out extra information

Returns

ret_data [RadarData object] An object containing the information read from the file

pyart.aux_io.**read_gamic**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *valid_range_from_file=True*, *units_from_file=True*, *pulse_width=None*, ***kwargs*)

Read a GAMIC hdf5 file.

Parameters

filename [str] Name of GAMIC HDF5 file to read data from.

field_names [dict, optional] Dictionary mapping field names in the file names to radar field names. Unlike other read functions, fields not in this dictionary or having a value of None are still included in the radar.fields dictionary, to exclude them use the *exclude_fields* parameter. Fields which are mapped by this dictionary will be renamed from key to value.

additional_metadata [dict of dicts, optional] This parameter is not used, it is included for uniformity.

file_field_names [bool, optional] True to force the use of the field names from the file in which case the *field_names* parameter is ignored. False will use to *field_names* parameter to rename fields.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

valid_range_from_file [bool, optional] True to extract valid range (valid_min and valid_max) for all field from the file when they are present. False will not extract these parameters.

units_from_file [bool, optional] True to extract the units for all fields from the file when available. False will not extract units using the default units for the fields.

pulse_width [list or None,] Mandatory for gamic radar processors which have pulsewidth enums. pulse_width should contain the pulsewidth' in us.

Returns

radar [Radar] Radar object.

pyart.aux_io.**read_gif**(*filename*, *additional_metadata=None*, *chy0=255.0*, *chx0=-160.0*, *xres=1.0*, *yres=1.0*, *nx=710*, *ny=640*, *nz=1*, ***kwargs*)

Read a MeteoSwiss operational radar data gif file.

Parameters

filename [str] Name of the file to read.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

chy0, chx0 [float] Swiss coordinates position of the south-western point in the grid

xres, yres [float] resolution of each grid point [km]

nx, ny, nz [int] dimensions of the grid

Returns

grid [Grid] Grid object containing the data.

pyart.aux_io.**read_iq**(*filename*, *filenames_iq*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *reader='C'*, *nbytes=4*, *prf=None*, *ang_tol=0.4*, *noise_h=None*, *noise_v=None*, *rconst_h=None*, *rconst_v=None*, *radconst_h=None*, *radconst_v=None*, *mfloss_h=1.0*, *mfloss_v=1.0*, *azi_min=None*, *azi_max=None*, *ele_min=None*, *ele_max=None*, *rng_min=None*, *rng_max=None*, ***kwargs*)

Read a rad4alp IQ file.

Parameters

filename [str] Name of the METRANET file to be used as reference.

filenames_iq [list of str] Name of the IQ files

field_names [dict, optional] Dictionary mapping RAINBOW field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

reader [str] The library used to read the METRANET reference file. Can be either 'C' or 'python'

nbytes [int] The number of bytes used to store the data in numpy arrays, e.g. if nbytes=4 then floats are going to be stored as np.float32

prf [float] The PRF of the read scan

ang_tol [float] Tolerated angle distance between nominal radar angle and angle in PSR files

noise_h, noise_v [float] The estimated H(V) noise power (ADU) of the scan

rconst_h, rconst_v [float] Dynamical factor used in the conversion from dBADU to dBm/dBZ

radconst_h, radconst_v [float] The H(V) radar constant

mfloss_h, mfloss_v [float] The H(V) matched filter losses in the receiver (dB)

azi_min, azi_max, ele_min, ele_max [float or None] The minimum and maximum angles to keep (deg)

rng_min, rng_max [float or None] The minimum and maximum ranges to keep (m)

Returns

radar [Radar] Radar object containing data from PSR file.

pyart.aux_io.**read_kazr**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*)

Read K-band ARM Zenith Radar (KAZR) NetCDF ingest data.

Parameters

filename [str] Name of NetCDF file to read data from.

field_names [dict, optional] Dictionary mapping field names in the file names to radar field names. Unlike other read functions, fields not in this dictionary or having a value of None are still included in the radar.fields dictionary, to exclude them use the *exclude_fields* parameter. Fields which are mapped by this dictionary will be renamed from key to value.

additional_metadata [dict of dicts, optional] This parameter is not used, it is included for uniformity.

file_field_names [bool, optional] True to force the use of the field names from the file in which case the *field_names* parameter is ignored. False will use to *field_names* parameter to rename fields.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

Returns

radar [Radar] Radar object.

pyart.aux_io.**read_metranet**(*filename*, *field_names=None*, *rmax=0.0*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *reader='C'*, *nbytes=4*, ***kwargs*)

Read a METRANET file.

Parameters

filename [str] Name of the METRANET file to read.

field_names [dict, optional] Dictionary mapping METRANET field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

rmax [float, optional] Maximum radar range to store in the radar object [m]. If 0 all data will be stored

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata during this read. This metadata is not used during any successive file reads unless explicitly

included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters.

reader [str] The reader library to use. Can be either 'C' or 'python'

nbytes [int] The number of bytes used to store the data in numpy arrays, e.g. if nbytes=4 then floats are going to be stored as np.float32

Returns

radar [Radar] Radar object containing data from METRANET file.

pyart.aux_io.**read_noxp_iphex_nc**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, ***kwargs*)

Read a NOXP IPHEX netCDF file.

Parameters

filename [str] Name of the netCDF file to read.

field_names [dict, optional] Dictionary mapping netCDF field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the netCDF data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

Returns

radar [Radar] Radar object containing data from netCDF file.

pyart.aux_io.**read_odim_h5**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, ***kwargs*)

Read a ODIM_H5 file.

Parameters

filename [str] Name of the ODIM_H5 file to read.

field_names [dict, optional] Dictionary mapping ODIM_H5 field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

Returns

radar [Radar] Radar object containing data from ODIM_H5 file.

pyart.aux_io.**read_pattern**(*filename*, ***kwargs*)

Read a netCDF file from a PATTERN project X-band radar.

Parameters

filename [str] Name of netCDF file to read data from.

Returns

radar [Radar] Radar object.

pyart.aux_io.**read_product_c**(*radar_file*, *physic_value=False*, *masked_array=False*, *verbose=False*)

Reads a METRANET cartesian data file

Parameters

radar_file [str] file name

physic_value [boolean] If true returns the physical value. Otherwise the digital value

masked_array [boolean] If true returns a numpy masked array with NaN values masked. Otherwise returns a regular masked array with NaN values

verbose [boolean] If true prints out extra information

Returns

ret_data [RadarData object] An object containing the information read from the file. None if the file has not been properly read

pyart.aux_io.**read_product_py**(*radar_file*, *physic_value=False*, *masked_array=False*, *verbose=False*)

Reads a METRANET cartesian data file

Parameters

radar_file [str] file name

physic_value [boolean] If true returns the physical value. Otherwise the digital value

masked_array [boolean] If true returns a numpy masked array with NaN values masked. Otherwise returns a regular masked array with NaN values

verbose [boolean] If true prints out extra information

Returns

ret_data [RadarData object] An object containing the information read from the file. None if the file has not been properly read

pyart.aux_io.**read_psr_cpi_header**(*filename*)
Reads the CPI data headers contained in a PSR file

Parameters

filename [str] Name of the PSR file

Returns

cpi_header, header [dict] Dictionary containing the PSR header data and the CPI headers data

pyart.aux_io.**read_psr_header**(*filename*)
Read a PSR file header.

Parameters

filename [str] Name of the PSR file

Returns

header [dict] Dictionary containing the PSR header data

pyart.aux_io.**read_psr_spectra**(*filename*)
Reads the complex spectral data contained in a PSR file

Parameters

filename [str] Name of the PSR file

Returns

spectra [3D complex ndArray] The complex spectra

pyart.aux_io.**read_radx**(*filename*, *radx_dir=None*, ***kwargs*)
Read a file by first converting it to Cf/Radial using RadxConvert.

Parameters

filename [str] Name of file to read using RadxConvert.

radx_dir [str, optional] path to the radx install

Returns

radar [Radar] Radar object.

pyart.aux_io.**read_rainbow_psr**(*filename*, *filenames_psr*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *undo_txcorr=True*, *cpi='mean'*, *ang_tol=0.5*, *azi_min=None*, *azi_max=None*, *ele_min=None*, *ele_max=None*, *rng_min=None*, *rng_max=None*, ***kwargs*)
Read a PSR file.

Parameters

filename [str] Name of the rainbow file to be used as reference.

filenames_psr [list of str] Name of the PSR files

field_names [dict, optional] Dictionary mapping RAINBOW field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

undo_txcorr: Bool If True the correction of the transmitted power is removed from the noise signal

cpi [str] The CPI to use. Can be 'low_prf', 'intermediate_prf', 'high_prf', 'mean', 'all'. If 'mean' the mean within the angle step is taken

ang_tol [float] Tolerated angle distance between nominal radar angle and angle in PSR files

azi_min, azi_max, ele_min, ele_max [float or None] The minimum and maximum angles to keep (deg)

rng_min, rng_max [float or None] The minimum and maximum ranges to keep (m)

Returns

radar [Radar] Radar object containing data from PSR file.

pyart.aux_io.**read_rainbow_psr_spectra**(*filename*, *filenames_psr*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *undo_txcorr=True*, *fold=True*, *positive_away=True*, *cpi='low_prf'*, *ang_tol=0.5*, *azi_min=None*, *azi_max=None*, *ele_min=None*, *ele_max=None*, *rng_min=None*, *rng_max=None*, ***kwargs*)

Read a PSR file to get the complex spectra

Parameters

filename [str] Name of the rainbow file to be used as reference.

filenames_psr [list of str] list of PSR file names

field_names [dict, optional] Dictionary mapping RAINBOW field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

undo_txcorr: Bool If True the correction of the transmitted power is removed from the noise signal

fold: Bool If True the spectra is folded so that 0-Doppler is in the middle

positive_away: Bool If True the spectra is reversed so that positive velocities are away from the radar

cpi [str] The CPI to use. Can be 'low_prf', 'intermediate_prf', 'high_prf' or 'all'

ang_tol [float] Tolerated angle distance between nominal radar angle and angle in PSR files

azi_min, azi_max, ele_min, ele_max [float or None] The minimum and maximum angles to keep (deg)

rng_min, rng_max [float or None] The minimum and maximum ranges to keep (m)

Returns

radar [Radar] Radar object containing data from PSR file.

pyart.aux_io.**read_rainbow_wrl**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *nbytes=4*, ***kwargs*)

Read a RAINBOW file. This routine has been tested to read rainbow5 files version 5.22.3, 5.34.16 and 5.35.1. Since the rainbow file format is evolving constantly there is no guaranty that it can work with other versions. If necessary, the user should adapt to code according to its own file version and raise an issue upstream.

Data types read by this routine: Reflectivity: dBZ, dBuZ, dBZv, dBuZv Velocity: V, Vu, Vv, Vvu Spectrum width: W, Wu, Wv, Wvu Differential reflectivity: ZDR, ZDRu Co-polar correlation coefficient: RhoHV, RhoHVu Co-polar differential phase: PhiDP, uPhiDP, uPhiDPu Specific differential phase: KDP, uKDP, uKDPu Signal quality parameters: SQI, SQIu, SQIv, SQIvu Temperature: TEMP Position of the range bin respect to the ISO0: ISO0 radar visibility according to Digital Elevation Model (DEM): VIS

Parameters

filename [str] Name of the RAINBOW file to read.

field_names [dict, optional] Dictionary mapping RAINBOW field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata during this read. This metadata is not used during any successive file reads unless explicitly

included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

nbytes [int] The number of bytes used to store the data in numpy arrays, e.g. if nbytes=4 then floats are going to be stored as np.float32

Returns

radar [Radar] Radar object containing data from RAINBOW file.

pyart.aux_io.**read_sinarame_h5**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, ***kwargs*)

Read a SINARAME_H5 file.

Parameters

filename [str] Name of the SINARAME_H5 file to read.

field_names [dict, optional] Dictionary mapping SINARAME_H5 field names to radar field names. If a data type found in the file does not appear in this dictionary or has a value of None it will not be placed in the radar.fields dictionary. A value of None, the default, will use the mapping defined in the Py-ART configuration file.

additional_metadata [dict of dicts, optional] Dictionary of dictionaries to retrieve metadata from during this read. This metadata is not used during any successive file reads unless explicitly included. A value of None, the default, will not introduct any addition metadata and the file specific or default metadata as specified by the Py-ART configuration file will be used.

file_field_names [bool, optional] True to use the MDV data type names for the field names. If this case the field_names parameter is ignored. The field dictionary will likely only have a 'data' key, unless the fields are defined in *additional_metadata*.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

Returns

radar [Radar] Radar object containing data from SINARAME_H5 file.

pyart.aux_io.**read_spectra**(*filename*, *field_names=None*, *additional_metadata=None*, *file_field_names=False*, *exclude_fields=None*, *include_fields=None*, *delay_field_loading=False*, ***kwargs*)

Read a spectra netCDF file.

Parameters

filename [str] Name of CF/Radial netCDF file to read data from.

field_names [dict, optional] Dictionary mapping field names in the file names to radar field names. Unlike other read functions, fields not in this dictionary or having a value of None are still included in the radar.fields dictionary, to exclude them use the *exclude_fields* parameter. Fields which are mapped by this dictionary will be renamed from key to value.

additional_metadata [dict of dicts, optional] This parameter is not used, it is included for uniformity.

file_field_names [bool, optional] True to force the use of the field names from the file in which case the *field_names* parameter is ignored. False will use to *field_names* parameter to rename fields.

exclude_fields [list or None, optional] List of fields to exclude from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields specified by include_fields.

include_fields [list or None, optional] List of fields to include from the radar object. This is applied after the *file_field_names* and *field_names* parameters. Set to None to include all fields not specified by exclude_fields.

delay_field_loading [bool] True to delay loading of field data from the file until the 'data' key in a particular field dictionary is accessed. In this case the field attribute of the returned Radar object will contain LazyLoadDict objects not dict objects. Delayed field loading will not provide any speedup in file where the number of gates vary between rays (ngates_vary=True) and is not recommended.

Returns

radar [Radar] Radar object.

Notes

This function has not been tested on "stream" Cfradial files.

pyart.aux_io.**write_odim_h5**(*filename*, *radar*, *field_names=None*, *physical=True*, *compression='gzip'*, *compression_opts=6*)

Write a Radar object to a EUMETNET OPERA compliant HDF5 file.

The files produced by this routine follow the EUMETNET OPERA information model: http://eumetnet.eu/wp-content/uploads/2017/01/OPERA_hdf_description_2014.pdf

Supported features:

- Writing PPIs: PVOL and SCAN objects - Different sweeps are saved in different dataset groups
- Writing sectorized PPIs and SCANs: AZIM objects
- Writing RHIs: ELEV objects

Not yet supported:

- Mixed datasets (how group always on top level)
- Single ray data (e.g. from fixed staring mode)
- Profiles

Parameters

filename [str] Filename of file to create.

radar [Radar] Radar object to process.

field_names [list of str] The list of fields from the radar object to save. If none all fields in the radar object will be saved.

physical [Bool] If true the physical values are stored. nodata parameter is equal to the _FillValue parameter in the field metadata or the default Py-ART fill value. If false the data is converted into binary values using a linear conversion. The gain and offset are either specified in the metadata of the field with keywords 'scale_factor' and 'add_offset' or calculated on the fly. keyword '_Write_as_dtype' specifies the datatype. It can be either 'uint8' or 'uint16'. The default datatype is uint8. The 'undetect' parameter is not used

compression [str] The type of compression for the datasets. Typical are "gzip" and "lzf".

compression_opts [any] The compression options. In the case of gzip is the level between 0 to 9 (recomended 1 to 6). In the case of lzf there are not options.

pyart.aux_io.**write_sinarame_cfradial**(*path*)

This function takes SINARAME_H5 files (where every file has only one field and one volume) from a folder and writes a CfRadial file for each volume including all fields.

Parameters

path [str] Where the SINARAME_H5 files are.

pyart.aux_io.**write_spectra**(*filename*, *radar*, *format='NETCDF4'*, *time_reference=None*, *arm_time_variables=False*, *physical=True*)

Write a Radar Spectra object to a netCDF file.

The files produced by this routine follow the CF/Radial standard. Attempts are also made to to meet many of the standards outlined in the ARM Data File Standards.

To control how the netCDF variables are created, set any of the following keys in the radar attribute dictionaries.

- _Zlib
- _DeflateLevel
- _Shuffle
- _Fletcher32
- _Continguous
- _ChunkSizes
- _Endianness
- _Least_significant_digit
- _FillValue

See the netCDF4 documentation for details on these settings.

Parameters

filename [str] Filename to create.

radar [Radar] Radar object.

format [str, optional] NetCDF format, one of 'NETCDF4', 'NETCDF4_CLASSIC', 'NETCDF3_CLASSIC' or 'NETCDF3_64BIT'. See netCDF4 documentation for details.

time_reference [bool] True to include a time_reference variable, False will not include this variable. The default, None, will include the time_reference variable when the first time value is non-zero.

arm_time_variables [bool] True to create the ARM standard time variables base_time and time_offset, False will not create these variables.

physical [bool] True to store the radar fields as physical numbers, False will store the radar fields as binary if the keyword '_Write_as_dtype' is in the field metadata. The gain and offset can be specified in the keyword 'scale_factor' and 'add_offset' or calculated on the fly.

CHAPTER

THREE

CORE (PYART.CORE)

Core Py-ART classes and function for interacting with weather radar data.

3.1 Core classes

Radar(time, _range, fields, metadata, ... [, ...])	A class for storing antenna coordinate radar data.
Grid(time, fields, metadata, ... [, ...])	A class for storing rectilinear gridded radar data in Cartesian coordinate.
HorizontalWindProfile(height, speed, direction)	Horizontal wind profile.
RadarSpectra(time, _range, fields, metadata, ...)	A class for storing antenna coordinate radar spectra data.

3.2 Coordinate transformations

antenna_to_cartesian(ranges, azimuths, ...)	Return Cartesian coordinates from antenna coordinates.
antenna_vectors_to_cartesian(ranges, ... [, ...])	Calculate Cartesian coordinate for gates from antenna coordinate vectors.
cartesian_to_geographic(x, y, projparams)	Cartesian to Geographic coordinate transform.
cartesian_vectors_to_geographic(x, y, projparams)	Cartesian vectors to Geographic coordinate transform.
cartesian_to_geographic_aeqd(x, y, lon_0, lat_0)	Azimuthal equidistant Cartesian to geographic coordinate transform.
cartesian_to_antenna(x, y, z)	Returns antenna coordinates from Cartesian coordinates.
geographic_to_cartesian(lon, lat, projparams)	Geographic to Cartesian coordinate transform.
geographic_to_cartesian_aeqd(lon, lat, ...)	Azimuthal equidistant geographic to Cartesian coordinate transform.
wgs84_to_swissCH1903(lon, lat, alt[, ...])	Convert WGS84 coordinates to swiss coordinates (CH1903 / LV03)

class pyart.core.**Grid**(*time*, *fields*, *metadata*, *origin_latitude*, *origin_longitude*, *origin_altitude*, *x*, *y*, *z*, *projection=None*, *radar_latitude=None*, *radar_longitude=None*, *radar_altitude=None*, *radar_time=None*, *radar_name=None*)

Bases: object

A class for storing rectilinear gridded radar data in Cartesian coordinate.

Refer to the attribute section for information on the parameters.

To create a Grid object using legacy parameters present in Py-ART version 1.5 and before, use `from_legacy_parameters()`, grid = Grid.from_legacy_parameters(fields, axes, metadata).

Attributes

time [dict] Time of the grid.

fields [dict of dicts] Moments from radars or other variables.

metadata [dict] Metadata describing the grid.

origin_longitude, origin_latitude, origin_altitude [dict] Geographic coordinate of the origin of the grid.

x, y, z [dict, 1D] Distance from the grid origin for each Cartesian coordinate axis in a one dimensional array. Defines the spacing along the three grid axes which is repeated throughout the grid, making a rectilinear grid.

nx, ny, nz [int] Number of grid points along the given Cartesian dimension.

projection [dic or str] Projection parameters defining the map projection used to transform from Cartesian to geographic coordinates. None will use the default dictionary with the 'proj' key set to 'pyart_aeqd' indicating that the native Py-ART azimuthal equidistant projection is used. Other values should specify a valid pyproj.Proj projparams dictionary or string. The special key '_include_lon_0_lat_0' is removed when interpreting this dictionary. If this key is present and set to True, which is required when proj='pyart_aeqd', then the radar longitude and latitude will be added to the dictionary as 'lon_0' and 'lat_0'. Use the *get_projparams()* method to retrieve a copy of this attribute dictionary with this special key evaluated.

radar_longitude, radar_latitude, radar_altitude [dict or None, optional] Geographic location of the radars which make up the grid.

radar_time [dict or None, optional] Start of collection for the radar which make up the grid.

radar_name [dict or None, optional] Names of the radars which make up the grid.

nradar [int] Number of radars whose data was used to make the grid.

projection_proj [Proj] pyproj.Proj instance for the projection specified by the projection attribute. If the 'pyart_aeqd' projection is specified accessing this attribute will raise a ValueError.

point_x, point_y, point_z [LazyLoadDict] The Cartesian locations of all grid points from the origin in the three Cartesian coordinates. The three dimensional data arrays contained these attributes are calculated from the x, y, and z attributes. If these attributes are changed use :py:func: *init_point_x_y_z* to reset the attributes.

point_longitude, point_latitude [LazyLoadDict] Geographic location of each grid point. The projection parameter(s) defined in the *projection* attribute are used to perform an inverse map projection from the Cartesian grid point locations relative to the grid origin. If these attributes are changed use *init_point_longitude_latitude()* to reset the attributes.

point_altitude [LazyLoadDict] The altitude of each grid point as calculated from the altitude of the grid origin and the Cartesian z location of each grid point. If this attribute is changed use *init_point_altitude()* to reset the attribute.

Methods

add_field(self, field_name, field_dict[, ...])	Add a field to the object.
get_point_longitude_latitude(self[, level, ...])	Return arrays of longitude and latitude for a given grid height level.
get_projparams(self)	Return a projparam dict from the projection attribute.
init_point_altitude(self)	Initialize the point_altitude attribute.
init_point_longitude_latitude(self)	Initialize or reset the point_{longitude, latitudes} attributes.
init_point_x_y_z(self)	Initialize or reset the point_{x, y, z} attributes.
to_xarray(self)	Convert the Grid object to an xarray format.
write(self, filename[, format, ...])	Write the the Grid object to a NetCDF file.

__class__
alias of builtins.type

__delattr__(*self*, *name*, /)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.core.grid', '__doc__': "\n A class for

__dir__(*self*, /)
Default dir() implementation.

__eq__(*self*, *value*, /)
Return self==value.

__format__(*self*, *format_spec*, /)
Default object formatter.

__ge__(*self*, *value*, /)
Return self>=value.

__getattribute__(*self*, *name*, /)
Return getattr(self, name).

__getstate__(*self*)
Return object's state which can be pickled.

__gt__(*self*, *value*, /)
Return self>value.

__hash__(*self*, /)
Return hash(self).

__init__(*self*, *time*, *fields*, *metadata*, *origin_latitude*, *origin_longitude*, *origin_altitude*, *x*, *y*, *z*, *projection=None*, *radar_latitude=None*, *radar_longitude=None*, *radar_altitude=None*, *radar_time=None*, *radar_name=None*)
Initalize object.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, /)
Return self<=value.

__lt__(*self*, *value*, /)
Return self<value.

__module__ = 'pyart.core.grid'

__ne__(*self*, *value*, */*)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, */*)
Helper for pickle.

__reduce_ex__(*self*, *protocol*, */*)
Helper for pickle.

__repr__(*self*, */*)
Return repr(self).

__setattr__(*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__setstate__(*self*, *state*)
Restore unpicklable entries from pickled object.

__sizeof__(*self*, */*)
Size of object in memory, in bytes.

__str__(*self*, */*)
Return str(self).

__subclasshook__()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_find_and_check_nradar(*self*)
Return the number of radars which were used to create the grid.

Examine the radar attributes to determine the number of radars which were used to create the grid. If the size of the radar attributes are inconsistent a ValueError is raised by this method.

add_field(*self*, *field_name*, *field_dict*, *replace_existing=False*)
Add a field to the object.

Parameters

field_name [str] Name of the field to the fields dictionary.

field_dict [dict] Dictionary containing field data and metadata.

replace_existing [bool, optional] True to replace the existing field with key field_name if it exists, overwriting the existing data. If False, a ValueError is raised if field_name already exists.

get_point_longitude_latitude(*self*, *level=0*, *edges=False*)
Return arrays of longitude and latitude for a given grid height level.

Parameters

level [int, optional] Grid height level at which to determine latitudes and longitudes. This is not currently used as all height level have the same layout.

edges [bool, optional] True to calculate the latitude and longitudes of the edges by interpolating between Cartesian coordinates points and extrapolating at the boundaries. False to calculate the locations at the centers.

Returns

longitude, latitude [2D array] Arrays containing the latitude and longitudes, in degrees, of the grid points or edges between grid points for the given height.

get_projparams(*self*)
Return a projparam dict from the projection attribute.

init_point_altitude(*self*)
Initialize the point_altitude attribute.

init_point_longitude_latitude(*self*)
Initialize or reset the point_{longitude, latitudes} attributes.

init_point_x_y_z(*self*)
Initialize or reset the point_{x, y, z} attributes.

projection_proj

to_xarray(*self*)
Convert the Grid object to an xarray format.

Attributes

time [dict] Time of the grid.

fields [dict of dicts] Moments from radars or other variables.

longitude, latitude [dict, 2D] Arrays of latitude and longitude for the grid height level.

x, y, z [dict, 1D] Distance from the grid origin for each Cartesian coordinate axis in a one dimensional array.

write(*self*, *filename*, *format='NETCDF4'*, *arm_time_variables=False*, *arm_alt_lat_lon_variables=False*)
Write the the Grid object to a NetCDF file.

Parameters

filename [str] Filename to save to.

format [str, optional] NetCDF format, one of 'NETCDF4', 'NETCDF4_CLASSIC', 'NETCDF3_CLASSIC' or 'NETCDF3_64BIT'.

arm_time_variables [bool, optional] True to write the ARM standard time variables base_time and time_offset. False will not write these variables.

arm_alt_lat_lon_variables [bool, optional] True to write the ARM standard alt, lat, lon variables. False will not write these variables.

class pyart.core.**HorizontalWindProfile**(*height*, *speed*, *direction*, *latitude=None*, *longitude=None*)

Bases: object

Horizontal wind profile.

Parameters

height [array-like, 1D] Heights in meters above sea level at which horizontal winds were sampled.

speed [array-like, 1D] Horizontal wind speed in meters per second at each height sampled.

direction [array-like, 1D] Horizontal wind direction in degrees at each height sampled.

Other Parameters

latitude [array-like, 1D, optional] Latitude in degrees north at each height sampled.

longitude [array-like, 1D, optional] Longitude in degrees east at each height sampled.

Attributes

height [array, 1D] Heights in meters above sea level at which horizontal winds were sampled.

speed [array, 1D] Horizontal wind speed in meters per second at each height.

direction [array, 1D] Horizontal wind direction in degrees at each height.

u_wind [array, 1D] U component of horizontal wind in meters per second.

v_wind [array, 1D] V component of horizontal wind in meters per second.

Methods

from_u_and_v(height, u_wind, v_wind)	Create a HorizontalWindProfile instance from U and V components.

__class__
alias of builtins.type

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.core.wind_profile', '__doc__': '\n Hor

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *height*, *speed*, *direction*, *latitude=None*, *longitude=None*)
initialize

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__ (*self*, *value*, /)
Return self<=value.

__lt__ (*self*, *value*, /)
Return self<value.

__module__ = 'pyart.core.wind_profile'

__ne__ (*self*, *value*, /)
Return self!=value.

__new__ (**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__ (*self*, /)
Helper for pickle.

__reduce_ex__ (*self*, *protocol*, /)
Helper for pickle.

__repr__ (*self*, /)
Return repr(self).

__setattr__ (*self*, *name*, *value*, /)
Implement setattr(self, name, value).

__sizeof__ (*self*, /)
Size of object in memory, in bytes.

__str__ (*self*, /)
Return str(self).

__subclasshook__ ()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_parse_location_data (*self*, *latitude*, *longitude*)
Parse profile location data.

classmethod from_u_and_v (*height*, *u_wind*, *v_wind*)
Create a HorizontalWindProfile instance from U and V components.

Parameters

height [array-like, 1D] Heights in meters above sea level at which horizontal winds were sampled.

u_wind [array-like, 1D] U component of horizontal wind speed in meters per second.

v_wind [array-like, 1D] V component of horizontal wind speed in meters per second.

u_wind
U component of horizontal wind in meters per second.

v_wind
V component of horizontal wind in meters per second.

class pyart.core.**Radar**(*time*, *_range*, *fields*, *metadata*, *scan_type*, *latitude*, *longitude*, *altitude*, *sweep_number*, *sweep_mode*, *fixed_angle*, *sweep_start_ray_index*, *sweep_end_ray_index*, *azimuth*, *elevation*, *altitude_agl=None*, *target_scan_rate=None*, *rays_are_indexed=None*, *ray_angle_res=None*, *scan_rate=None*, *antenna_transition=None*, *instrument_parameters=None*, *radar_calibration=None*, *rotation=None*, *tilt=None*, *roll=None*, *drift=None*, *heading=None*, *pitch=None*, *georefs_applied=None*)

Bases: object

A class for storing antenna coordinate radar data.

The structure of the Radar class is based on the CF/Radial Data file format. Global attributes and variables (section 4.1 and 4.3) are represented as a dictionary in the metadata attribute. Other required and optional variables are represented as dictionaries in a attribute with the same name as the variable in the CF/Radial standard. When a optional attribute not present the attribute has a value of None. The data for a given variable is stored in the dictionary under the 'data' key. Moment field data is stored as a dictionary of dictionaries in the fields attribute. Sub-convention variables are stored as a dictionary of dictionaries under the meta_group attribute.

Refer to the attribute section for information on the parameters.

Attributes

time [dict] Time at the center of each ray.

range [dict] Range to the center of each gate (bin).

fields [dict of dicts] Moment fields.

metadata [dict] Metadata describing the instrument and data.

scan_type [str] Type of scan, one of 'ppi', 'rhi', 'sector' or 'other'. If the scan volume contains multiple sweep modes this should be 'other'.

latitude [dict] Latitude of the instrument.

longitude [dict] Longitude of the instrument.

altitude [dict] Altitude of the instrument, above sea level.

altitude_agl [dict or None] Altitude of the instrument above ground level. If not provided this attribute is set to None, indicating this parameter not available.

sweep_number [dict] The number of the sweep in the volume scan, 0-based.

sweep_mode [dict] Sweep mode for each mode in the volume scan.

fixed_angle [dict] Target angle for thr sweep. Azimuth angle in RHI modes, elevation angle in all other modes.

sweep_start_ray_index [dict] Index of the first ray in each sweep relative to the start of the volume, 0-based.

sweep_end_ray_index [dict] Index of the last ray in each sweep relative to the start of the volume, 0-based.

rays_per_sweep [LazyLoadDict] Number of rays in each sweep. The data key of this attribute is create upon first access from the data in the sweep_start_ray_index and sweep_end_ray_index attributes. If the sweep locations needs to be modified, do this prior to accessing this attribute or use *init_rays_per_sweep()* to reset the attribute.

target_scan_rate [dict or None] Intended scan rate for each sweep. If not provided this attribute is set to None, indicating this parameter is not available.

rays_are_indexed [dict or None] Indication of whether ray angles are indexed to a regular grid in each sweep. If not provided this attribute is set to None, indicating ray angle spacing is not determined.

ray_angle_res [dict or None] If rays_are_indexed is not None, this provides the angular resolution of the grid. If not provided or available this attribute is set to None.

azimuth [dict] Azimuth of antenna, relative to true North. Azimuth angles are recommended to be expressed in the range of [0, 360], but other representations are not forbidden.

elevation [dict] Elevation of antenna, relative to the horizontal plane. Elevation angles are recommended to be expressed in the range of [-180, 180], but other representations are not forbidden.

gate_x, gate_y, gate_z [LazyLoadDict] Location of each gate in a Cartesian coordinate system assuming a standard atmosphere with a 4/3 Earth's radius model. The data keys of these attributes are create upon first access from the data in the range, azimuth and elevation attributes. If these attributes are changed use *init_gate_x_y_z()* to reset.

gate_longitude, gate_latitude [LazyLoadDict] Geographic location of each gate. The projection parameter(s) defined in the *projection* attribute are used to perform an inverse map projection from the Cartesian gate locations relative to the radar location to longitudes and latitudes. If these attributes are changed use *init_gate_longitude_latitude()* to reset the attributes.

projection [dic or str] Projection parameters defining the map projection used to transform from Cartesian to geographic coordinates. The default dictionary sets the 'proj' key to 'pyart_aeqd' indicating that the native Py-ART azimuthal equidistant projection is used. This can be modified to specify a valid pyproj.Proj projparams dictionary or string. The special key '_include_lon_0_lat_0' is removed when interpreting this dictionary. If this key is present and set to True, which is required when proj='pyart_aeqd', then the radar longitude and latitude will be added to the dictionary as 'lon_0' and 'lat_0'.

gate_altitude [LazyLoadDict] The altitude of each radar gate as calculated from the altitude of the radar and the Cartesian z location of each gate. If this attribute is changed use *init_gate_altitude()* to reset the attribute.

scan_rate [dict or None] Actual antenna scan rate. If not provided this attribute is set to None, indicating this parameter is not available.

antenna_transition [dict or None] Flag indicating if the antenna is in transition, 1 = yes, 0 = no. If not provided this attribute is set to None, indicating this parameter is not available.

rotation [dict or None] The rotation angle of the antenna. The angle about the aircraft longitudinal axis for a vertically scanning radar.

tilt [dict or None] The tilt angle with respect to the plane orthogonal (Z-axis) to aircraft longitudinal axis.

roll [dict or None] The roll angle of platform, for aircraft right wing down is positive.

drift [dict or None] Drift angle of antenna, the angle between heading and track.

heading [dict or None] Heading (compass) angle, clockwise from north.

pitch [dict or None] Pitch angle of antenna, for aircraft nose up is positive.

georefs_applied [dict or None] Indicates whether the variables have had georeference calculation applied. Leading to Earth-centric azimuth and elevation angles.

instrument_parameters [dict of dicts or None] Instrument parameters, if not provided this attribute is set to None, indicating these parameters are not avaiable. This dictionary also includes variables in the radar_parameters CF/Radial subconvention.

radar_calibration [dict of dicts or None] Instrument calibration parameters. If not provided this attribute is set to None, indicating these parameters are not available

ngates [int] Number of gates (bins) in a ray.

nrays [int] Number of rays in the volume.

nsweeps [int] Number of sweep in the volume.

Methods

add_field(self, field_name, dic[, ...])	Add a field to the object.
add_field_like(self, existing_field_name, ...)	Add a field to the object with metadata from a existing field.
check_field_exists(self, field_name)	Check that a field exists in the fields dictionary.
extract_sweeps(self, sweeps)	Create a new radar contains only the data from select sweeps.
get_azimuth(self, sweep[, copy])	Return an array of azimuth angles for a given sweep.
get_elevation(self, sweep[, copy])	Return an array of elevation angles for a given sweep.
get_end(self, sweep)	Return the ending ray for a given sweep.
get_field(self, sweep, field_name[, copy])	Return the field data for a given sweep.
get_gate_lat_lon_alt(self, sweep[, ...])	Return the longitude, latitude and altitude gate locations.
get_gate_x_y_z(self, sweep[, edges, ...])	Return the x, y and z gate locations in meters for a given sweep.
get_nyquist_vel(self, sweep[, check_uniform])	Return the Nyquist velocity in meters per second for a given sweep.
get_slice(self, sweep)	Return a slice for selecting rays for a given sweep.
get_start(self, sweep)	Return the starting ray index for a given sweep.
get_start_end(self, sweep)	Return the starting and ending ray for a given sweep.
info(self[, level, out])	Print information on radar.
init_gate_altitude(self)	Initialize the gate_altitude attribute.
init_gate_longitude_latitude(self)	Initialize or reset the gate_longitude and gate_latitude attributes.
init_gate_x_y_z(self)	Initialize or reset the gate_{x, y, z} attributes.
init_rays_per_sweep(self)	Initialize or reset the rays_per_sweep attribute.
iter_azimuth(self)	Return an iterator which returns sweep azimuth data.
iter_elevation(self)	Return an iterator which returns sweep elevation data.
iter_end(self)	Return an iterator over the sweep end indices.
iter_field(self, field_name)	Return an iterator which returns sweep field data.
iter_slice(self)	Return an iterator which returns sweep slice objects.
iter_start(self)	Return an iterator over the sweep start indices.
iter_start_end(self)	Return an iterator over the sweep start and end indices.

__class__
alias of `builtins.type`

__delattr__(*self*, *name*, */*)

Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.core.radar', '__doc__': "\n A class fo

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__getstate__(*self*)
Return object's state which can be pickled.

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *time*, *_range*, *fields*, *metadata*, *scan_type*, *latitude*, *longitude*, *altitude*, *sweep_number*, *sweep_mode*, *fixed_angle*, *sweep_start_ray_index*, *sweep_end_ray_index*, *azimuth*, *elevation*, *altitude_agl=None*, *target_scan_rate=None*, *rays_are_indexed=None*, *ray_angle_res=None*, *scan_rate=None*, *antenna_transition=None*, *instrument_parameters=None*, *radar_calibration=None*, *rotation=None*, *tilt=None*, *roll=None*, *drift=None*, *heading=None*, *pitch=None*, *georefs_applied=None*)
Initialize self. See help(type(self)) for accurate signature.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, */*)
Return self<=value.

__lt__(*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.core.radar'

__ne__(*self*, *value*, */*)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, */*)
Helper for pickle.

__reduce_ex__(*self*, *protocol*, */*)
Helper for pickle.

__repr__(*self*, */*)
Return repr(self).

__setattr__(*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__setstate__(*self*, *state*)
Restore unpicklable entries from pickled object.

__sizeof__(*self*, */*)
Size of object in memory, in bytes.

__str__(*self*, */*)
Return str(self).

__subclasshook__()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_check_sweep_in_range(*self*, *sweep*)
Check that a sweep number is in range.

_dic_info(*self*, *attr*, *level*, *out*, *dic=None*, *ident_level=0*)
Print information on a dictionary attribute.

add_field(*self*, *field_name*, *dic*, *replace_existing=False*)
Add a field to the object.

Parameters

field_name [str] Name of the field to add to the dictionary of fields.

dic [dict] Dictionary contain field data and metadata.

replace_existing [bool, optional] True to replace the existing field with key field_name if it exists, loosing any existing data. False will raise a ValueError when the field already exists.

add_field_like(*self*, *existing_field_name*, *field_name*, *data*, *replace_existing=False*)
Add a field to the object with metadata from a existing field.

Note that the data parameter is not copied by this method. If data refers to a 'data' array from an existing field dictionary, a copy should be made within or prior to using this method. If this is not done the 'data' key in both field dictionaries will point to the same NumPy array and modification of one will change the second. To copy NumPy arrays use the copy() method. See the Examples section for how to create a copy of the 'reflectivity' field as a field named 'reflectivity_copy'.

Parameters

existing_field_name [str] Name of an existing field to take metadata from when adding the new field to the object.

field_name [str] Name of the field to add to the dictionary of fields.

data [array] Field data. A copy of this data is not made, see the note above.

replace_existing [bool, optional] True to replace the existing field with key field_name if it exists, loosing any existing data. False will raise a ValueError when the field already exists.

Examples

```
>>> radar.add_field_like('reflectivity', 'reflectivity_copy',
...                      radar.fields['reflectivity']['data'].copy())
```

check_field_exists (*self*, *field_name*)
Check that a field exists in the fields dictionary.

If the field does not exist raise a KeyError.

Parameters

field_name [str] Name of field to check.

extract_sweeps (*self*, *sweeps*)
Create a new radar contains only the data from select sweeps.

Parameters

sweeps [array_like] Sweeps (0-based) to include in new Radar object.

Returns

radar [Radar] Radar object which contains a copy of data from the selected sweeps.

get_azimuth (*self*, *sweep*, *copy=False*)
Return an array of azimuth angles for a given sweep.

Parameters

sweep [int] Sweep number to retrieve data for, 0 based.

copy [bool, optional] True to return a copy of the azimuths. False, the default, returns a view of the azimuths (when possible), changing this data will change the data in the underlying Radar object.

Returns

azimuths [array] Array containing the azimuth angles for a given sweep.

get_elevation (*self*, *sweep*, *copy=False*)
Return an array of elevation angles for a given sweep.

Parameters

sweep [int] Sweep number to retrieve data for, 0 based.

copy [bool, optional] True to return a copy of the elevations. False, the default, returns a view of the elevations (when possible), changing this data will change the data in the underlying Radar object.

Returns

azimuths [array] Array containing the elevation angles for a given sweep.

get_end (*self*, *sweep*)
Return the ending ray for a given sweep.

get_field (*self*, *sweep*, *field_name*, *copy=False*)
Return the field data for a given sweep.

When used with *get_gate_x_y_z()* this method can be used to obtain the data needed for plotting a radar field with the correct spatial context.

Parameters

sweep [int] Sweep number to retrieve data for, 0 based.

field_name [str] Name of the field from which data should be retrieved.

copy [bool, optional] True to return a copy of the data. False, the default, returns a view of the data (when possible), changing this data will change the data in the underlying Radar object.

Returns

data [array] Array containing data for the requested sweep and field.

get_gate_lat_lon_alt (*self*, *sweep*, *reset_gate_coords=False*, *filter_transitions=False*)

Return the longitude, latitude and altitude gate locations. Longitude and latitude are in degrees and altitude in meters.

With the default parameter this method returns the same data as contained in the gate_latitude, gate_longitude and gate_altitude attributes but this method performs the gate location calculations only for the specified sweep and therefore is more efficient than accessing this data through these attribute. If coordinates have at all, please use the reset_gate_coords parameter.

Parameters

sweep [int] Sweep number to retrieve gate locations from, 0 based.

reset_gate_coords [bool, optional] Optional to reset the gate latitude, gate longitude and gate altitude attributes before using them in this function. This is useful when the geographic coordinates have changed and gate latitude, gate longitude and gate altitude need to be reset.

filter_transitions [bool, optional] True to remove rays where the antenna was in transition between sweeps. False will include these rays. No rays will be removed if the antenna_transition attribute is not available (set to None).

Returns

lat, lon, alt [2D array] Array containing the latitude, longitude and altitude, for all gates in the sweep.

get_gate_x_y_z (*self*, *sweep*, *edges=False*, *filter_transitions=False*)

Return the x, y and z gate locations in meters for a given sweep.

With the default parameter this method returns the same data as contained in the gate_x, gate_y and gate_z attributes but this method performs the gate location calculations only for the specified sweep and therefore is more efficient than accessing this data through these attribute.

When used with *get_field()* this method can be used to obtain the data needed for plotting a radar field with the correct spatial context.

Parameters

sweep [int] Sweep number to retrieve gate locations from, 0 based.

edges [bool, optional] True to return the locations of the gate edges calculated by interpolating between the range, azimuths and elevations. False (the default) will return the locations of the gate centers with no interpolation.

filter_transitions [bool, optional] True to remove rays where the antenna was in transition between sweeps. False will include these rays. No rays will be removed if the antenna_transition attribute is not available (set to None).

Returns

x, y, z [2D array] Array containing the x, y and z, distances from the radar in meters for the center (or edges) for all gates in the sweep.

get_nyquist_vel(*self*, *sweep*, *check_uniform=True*)
Return the Nyquist velocity in meters per second for a given sweep.

Raises a LookupError if the Nyquist velocity is not available, an Exception is raised if the velocities are not uniform in the sweep unless check_uniform is set to False.

Parameters

sweep [int] Sweep number to retrieve data for, 0 based.

check_uniform [bool] True to check to perform a check on the Nyquist velocities that they are uniform in the sweep, False will skip this check and return the velocity of the first ray in the sweep.

Returns

nyquist_velocity [float] Array containing the Nyquist velocity in m/s for a given sweep.

get_slice(*self*, *sweep*)
Return a slice for selecting rays for a given sweep.

get_start(*self*, *sweep*)
Return the starting ray index for a given sweep.

get_start_end(*self*, *sweep*)
Return the starting and ending ray for a given sweep.

info(*self*, *level='standard'*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Print information on radar.

Parameters

level [{'compact', 'standard', 'full', 'c', 's', 'f'}, optional] Level of information on radar object to print, compact is minimal information, standard more and full everything.

out [file-like, optional] Stream to direct output to, default is to print information to standard out (the screen).

init_gate_altitude(*self*)
Initialize the gate_altitude attribute.

init_gate_longitude_latitude(*self*)
Initialize or reset the gate_longitude and gate_latitude attributes.

init_gate_x_y_z(*self*)
Initialize or reset the gate_{x, y, z} attributes.

init_rays_per_sweep(*self*)
Initialize or reset the rays_per_sweep attribute.

iter_azimuth(*self*)
Return an iterator which returns sweep azimuth data.

iter_elevation(*self*)
Return an iterator which returns sweep elevation data.

iter_end(*self*)
Return an iterator over the sweep end indices.

iter_field(*self*, *field_name*)
Return an iterator which returns sweep field data.

iter_slice(*self*)
Return an iterator which returns sweep slice objects.

iter_start(*self*)
Return an iterator over the sweep start indices.

iter_start_end(*self*)
Return an iterator over the sweep start and end indices.

class pyart.core.**RadarSpectra**(*time*, *_range*, *fields*, *metadata*, *scan_type*, *latitude*, *longitude*, *altitude*, *sweep_number*, *sweep_mode*, *fixed_angle*, *sweep_start_ray_index*, *sweep_end_ray_index*, *azimuth*, *elevation*, *npulses*, *Doppler_velocity=None*, *Doppler_frequency=None*, *altitude_agl=None*, *target_scan_rate=None*, *rays_are_indexed=None*, *ray_angle_res=None*, *scan_rate=None*, *antenna_transition=None*, *instrument_parameters=None*, *radar_calibration=None*, *rotation=None*, *tilt=None*, *roll=None*, *drift=None*, *heading=None*, *pitch=None*, *georefs_applied=None*)

Bases: pyart.core.radar.Radar

A class for storing antenna coordinate radar spectra data. Based on the radar object class

The structure of the Radar class is based on the CF/Radial Data file format. Global attributes and variables (section 4.1 and 4.3) are represented as a dictionary in the metadata attribute. Other required and optional variables are represented as dictionaries in a attribute with the same name as the variable in the CF/Radial standard. When a optional attribute not present the attribute has a value of None. The data for a given variable is stored in the dictionary under the 'data' key. Moment field data is stored as a dictionary of dictionaries in the fields attribute. Sub-convention variables are stored as a dictionary of dictionaries under the meta_group attribute.

Refer to the attribute section for information on the parameters.

Attributes

time [dict] Time at the center of each ray.

range [dict] Range to the center of each gate (bin).

npulses [dict] number of pulses for each ray

Doppler_velocity [dict or None] The Doppler velocity of each Doppler bin. The data has dimensions nrays x npulses_max

Doppler_frequency [dict or None] The Doppler frequency of each Doppler bin. The data has dimensions nrays x npulses_max

fields [dict of dicts] Moment fields. The data has dimensions nrays x ngates x npulses_max

metadata [dict] Metadata describing the instrument and data.

scan_type [str] Type of scan, one of 'ppi', 'rhi', 'sector' or 'other'. If the scan volume contains multiple sweep modes this should be 'other'.

latitude [dict] Latitude of the instrument.

longitude [dict] Longitude of the instrument.

altitude [dict] Altitude of the instrument, above sea level.

altitude_agl [dict or None] Altitude of the instrument above ground level. If not provided this attribute is set to None, indicating this parameter not available.

sweep_number [dict] The number of the sweep in the volume scan, 0-based.

sweep_mode [dict] Sweep mode for each mode in the volume scan.

fixed_angle [dict] Target angle for thr sweep. Azimuth angle in RHI modes, elevation angle in all other modes.

sweep_start_ray_index [dict] Index of the first ray in each sweep relative to the start of the volume, 0-based.

sweep_end_ray_index [dict] Index of the last ray in each sweep relative to the start of the volume, 0-based.

rays_per_sweep [LazyLoadDict] Number of rays in each sweep. The data key of this attribute is create upon first access from the data in the sweep_start_ray_index and sweep_end_ray_index attributes. If the sweep locations needs to be modified, do this prior to accessing this attribute or use *init_rays_per_sweep()* to reset the attribute.

target_scan_rate [dict or None] Intended scan rate for each sweep. If not provided this attribute is set to None, indicating this parameter is not available.

rays_are_indexed [dict or None] Indication of whether ray angles are indexed to a regular grid in each sweep. If not provided this attribute is set to None, indicating ray angle spacing is not determined.

ray_angle_res [dict or None] If rays_are_indexed is not None, this provides the angular resolution of the grid. If not provided or available this attribute is set to None.

azimuth [dict] Azimuth of antenna, relative to true North. Azimuth angles are recommended to be expressed in the range of [0, 360], but other representations are not forbidden.

elevation [dict] Elevation of antenna, relative to the horizontal plane. Elevation angles are recommended to be expressed in the range of [-180, 180], but other representations are not forbidden.

gate_x, gate_y, gate_z [LazyLoadDict] Location of each gate in a Cartesian coordinate system assuming a standard atmosphere with a 4/3 Earth's radius model. The data keys of these attributes are create upon first access from the data in the range, azimuth and elevation attributes. If these attributes are changed use *init_gate_x_y_z()* to reset.

gate_longitude, gate_latitude [LazyLoadDict] Geographic location of each gate. The projection parameter(s) defined in the *projection* attribute are used to perform an inverse map projection from the Cartesian gate locations relative to the radar location to longitudes and latitudes. If these attributes are changed use *init_gate_longitude_latitude()* to reset the attributes.

projection [dic or str] Projection parameters defining the map projection used to transform from Cartesian to geographic coordinates. The default dictionary sets the 'proj' key to 'pyart_aeqd' indicating that the native Py-ART azimuthal equidistant projection is used. This can be modified to specify a valid pyproj.Proj projparams dictionary or string. The special key '_include_lon_0_lat_0' is removed when interpreting this dictionary. If this key is present and set to True, which is required when proj='pyart_aeqd', then the radar longitude and latitude will be added to the dictionary as 'lon_0' and 'lat_0'.

gate_altitude [LazyLoadDict] The altitude of each radar gate as calculated from the altitude of the radar and the Cartesian z location of each gate. If this attribute is changed use *init_gate_altitude()* to reset the attribute.

scan_rate [dict or None] Actual antenna scan rate. If not provided this attribute is set to None, indicating this parameter is not available.

antenna_transition [dict or None] Flag indicating if the antenna is in transition, 1 = yes, 0 = no. If not provided this attribute is set to None, indicating this parameter is not available.

rotation [dict or None] The rotation angle of the antenna. The angle about the aircraft longitudinal axis for a vertically scanning radar.

tilt [dict or None] The tilt angle with respect to the plane orthogonal (Z-axis) to aircraft longitudinal axis.

roll [dict or None] The roll angle of platform, for aircraft right wing down is positive.

drift [dict or None] Drift angle of antenna, the angle between heading and track.

heading [dict or None] Heading (compass) angle, clockwise from north.

pitch [dict or None] Pitch angle of antenna, for aircraft nose up is positive.

georefs_applied [dict or None] Indicates whether the variables have had georeference calculation applied. Leading to Earth-centric azimuth and elevation angles.

instrument_parameters [dict of dicts or None] Instrument parameters, if not provided this attribute is set to None, indicating these parameters are not avaiable. This dictionary also includes variables in the radar_parameters CF/Radial subconvention.

radar_calibration [dict of dicts or None] Instrument calibration parameters. If not provided this attribute is set to None, indicating these parameters are not available

ngates [int] Number of gates (bins) in a ray.

nrays [int] Number of rays in the volume.

npulses_max [int] Maximum number of pulses per ray in the volume.

nsweeps [int] Number of sweep in the volume.

Methods

add_field(self, field_name, dic[, ...])	Add a field to the object.
add_field_like(self, existing_field_name, ...)	Add a field to the object with metadata from a existing field.
check_field_exists(self, field_name)	Check that a field exists in the fields dictionary.
extract_sweeps(self, sweeps)	Create a new radar contains only the data from select sweeps.
get_azimuth(self, sweep[, copy])	Return an array of azimuth angles for a given sweep.
get_elevation(self, sweep[, copy])	Return an array of elevation angles for a given sweep.
get_end(self, sweep)	Return the ending ray for a given sweep.
get_field(self, sweep, field_name[, copy])	Return the field data for a given sweep.
get_gate_lat_lon_alt(self, sweep[, ...])	Return the longitude, latitude and altitude gate locations.
get_gate_x_y_z(self, sweep[, edges, ...])	Return the x, y and z gate locations in meters for a given sweep.
get_nyquist_vel(self, sweep[, check_uniform])	Return the Nyquist velocity in meters per second for a given sweep.
get_slice(self, sweep)	Return a slice for selecting rays for a given sweep.
get_start(self, sweep)	Return the starting ray index for a given sweep.
get_start_end(self, sweep)	Return the starting and ending ray for a given sweep.
info(self[, level, out])	Print information on radar.
init_gate_altitude(self)	Initialize the gate_altitude attribute.
init_gate_longitude_latitude(self)	Initialize or reset the gate_longitude and gate_latitude attributes.

Continued on next page

Table 6 – continued from previous page

init_gate_x_y_z(self)	Initialize or reset the gate_{x, y, z} attributes.
init_rays_per_sweep(self)	Initialize or reset the rays_per_sweep attribute.
iter_azimuth(self)	Return an iterator which returns sweep azimuth data.
iter_elevation(self)	Return an iterator which returns sweep elevation data.
iter_end(self)	Return an iterator over the sweep end indices.
iter_field(self, field_name)	Return an iterator which returns sweep field data.
iter_slice(self)	Return an iterator which returns sweep slice objects.
iter_start(self)	Return an iterator over the sweep start indices.
iter_start_end(self)	Return an iterator over the sweep start and end indices.

__class__
alias of `builtins.type`

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.core.radar_spectra', '__doc__': "\n A

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__getstate__(*self*)
Return object's state which can be pickled.

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *time*, *_range*, *fields*, *metadata*, *scan_type*, *latitude*, *longitude*, *altitude*, *sweep_number*, *sweep_mode*, *fixed_angle*, *sweep_start_ray_index*, *sweep_end_ray_index*, *azimuth*, *elevation*, *npulses*, *Doppler_velocity=None*, *Doppler_frequency=None*, *altitude_agl=None*, *target_scan_rate=None*, *rays_are_indexed=None*, *ray_angle_res=None*, *scan_rate=None*, *antenna_transition=None*, *instrument_parameters=None*, *radar_calibration=None*, *rotation=None*, *tilt=None*, *roll=None*, *drift=None*, *heading=None*, *pitch=None*, *georefs_applied=None*)
Initialize self. See help(type(self)) for accurate signature.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, */*)
Return self<=value.

__lt__ (*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.core.radar_spectra'

__ne__ (*self*, *value*, */*)
Return self!=value.

__new__ (**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__ (*self*, */*)
Helper for pickle.

__reduce_ex__ (*self*, *protocol*, */*)
Helper for pickle.

__repr__ (*self*, */*)
Return repr(self).

__setattr__ (*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__setstate__ (*self*, *state*)
Restore unpicklable entries from pickled object.

__sizeof__ (*self*, */*)
Size of object in memory, in bytes.

__str__ (*self*, */*)
Return str(self).

__subclasshook__ ()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_check_sweep_in_range (*self*, *sweep*)
Check that a sweep number is in range.

_dic_info (*self*, *attr*, *level*, *out*, *dic=None*, *ident_level=0*)
Print information on a dictionary attribute.

add_field (*self*, *field_name*, *dic*, *replace_existing=False*)
Add a field to the object.

Parameters

field_name [str] Name of the field to add to the dictionary of fields.

dic [dict] Dictionary contain field data and metadata.

replace_existing [bool] True to replace the existing field with key field_name if it exists, loosing any existing data. False will raise a ValueError when the field already exists.

add_field_like (*self*, *existing_field_name*, *field_name*, *data*, *replace_existing=False*)
Add a field to the object with metadata from a existing field.

Note that the data parameter is not copied by this method. If data refers to a 'data' array from an existing field dictionary, a copy should be made within or prior to using this method. If this is not done the 'data'

key in both field dictionaries will point to the same NumPy array and modification of one will change the second. To copy NumPy arrays use the copy() method. See the Examples section for how to create a copy of the 'reflectivity' field as a field named 'reflectivity_copy'.

Parameters

existing_field_name [str] Name of an existing field to take metadata from when adding the new field to the object.

field_name [str] Name of the field to add to the dictionary of fields.

data [array] Field data. A copy of this data is not made, see the note above.

replace_existing [bool] True to replace the existing field with key field_name if it exists, loosing any existing data. False will raise a ValueError when the field already exists.

Examples

```
>>> radar.add_field_like('reflectivity', 'reflectivity_copy',
...                      radar.fields['reflectivity']['data'].copy())
```

check_field_exists(*self*, *field_name*)
Check that a field exists in the fields dictionary.

If the field does not exist raise a KeyError.

Parameters

field_name [str] Name of field to check.

extract_sweeps(*self*, *sweeps*)
Create a new radar contains only the data from select sweeps.

Parameters

sweeps [array_like] Sweeps (0-based) to include in new Radar object.

Returns

radar [Radar] Radar object which contains a copy of data from the selected sweeps.

get_azimuth(*self*, *sweep*, *copy=False*)
Return an array of azimuth angles for a given sweep.

Parameters

sweep [int] Sweep number to retrieve data for, 0 based.

copy [bool, optional] True to return a copy of the azimuths. False, the default, returns a view of the azimuths (when possible), changing this data will change the data in the underlying Radar object.

Returns

azimuths [array] Array containing the azimuth angles for a given sweep.

get_elevation(*self*, *sweep*, *copy=False*)
Return an array of elevation angles for a given sweep.

Parameters

sweep [int] Sweep number to retrieve data for, 0 based.

copy [bool, optional] True to return a copy of the elevations. False, the default, returns a view of the elevations (when possible), changing this data will change the data in the underlying Radar object.

Returns

azimuths [array] Array containing the elevation angles for a given sweep.

get_end(*self*, *sweep*)
Return the ending ray for a given sweep.

get_field(*self*, *sweep*, *field_name*, *copy=False*)
Return the field data for a given sweep.

When used with *get_gate_x_y_z()* this method can be used to obtain the data needed for plotting a radar field with the correct spatial context.

Parameters

sweep [int] Sweep number to retrieve data for, 0 based.

field_name [str] Name of the field from which data should be retrieved.

copy [bool, optional] True to return a copy of the data. False, the default, returns a view of the data (when possible), changing this data will change the data in the underlying Radar object.

Returns

data [array] Array containing data for the requested sweep and field.

get_gate_lat_lon_alt(*self*, *sweep*, *reset_gate_coords=False*, *filter_transitions=False*)
Return the longitude, latitude and altitude gate locations. Longitude and latitude are in degrees and altitude in meters.

With the default parameter this method returns the same data as contained in the gate_latitude, gate_longitude and gate_altitude attributes but this method performs the gate location calculations only for the specified sweep and therefore is more efficient than accessing this data through these attribute. If coordinates have at all, please use the reset_gate_coords parameter.

Parameters

sweep [int] Sweep number to retrieve gate locations from, 0 based.

reset_gate_coords [bool, optional] Optional to reset the gate latitude, gate longitude and gate altitude attributes before using them in this function. This is useful when the geographic coordinates have changed and gate latitude, gate longitude and gate altitude need to be reset.

filter_transitions [bool, optional] True to remove rays where the antenna was in transition between sweeps. False will include these rays. No rays will be removed if the antenna_transition attribute is not available (set to None).

Returns

lat, lon, alt [2D array] Array containing the latitude, longitude and altitude, for all gates in the sweep.

get_gate_x_y_z(*self*, *sweep*, *edges=False*, *filter_transitions=False*)
Return the x, y and z gate locations in meters for a given sweep.

With the default parameter this method returns the same data as contained in the gate_x, gate_y and gate_z attributes but this method performs the gate location calculations only for the specified sweep and therefore is more efficient than accessing this data through these attribute.

When used with *get_field()* this method can be used to obtain the data needed for plotting a radar field with the correct spatial context.

Parameters

sweep [int] Sweep number to retrieve gate locations from, 0 based.

edges [bool, optional] True to return the locations of the gate edges calculated by interpolating between the range, azimuths and elevations. False (the default) will return the locations of the gate centers with no interpolation.

filter_transitions [bool, optional] True to remove rays where the antenna was in transition between sweeps. False will include these rays. No rays will be removed if the antenna_transition attribute is not available (set to None).

Returns

x, y, z [2D array] Array containing the x, y and z, distances from the radar in meters for the center (or edges) for all gates in the sweep.

get_nyquist_vel(*self*, *sweep*, *check_uniform=True*)
Return the Nyquist velocity in meters per second for a given sweep.

Raises a LookupError if the Nyquist velocity is not available, an Exception is raised if the velocities are not uniform in the sweep unless check_uniform is set to False.

Parameters

sweep [int] Sweep number to retrieve data for, 0 based.

check_uniform [bool] True to check to perform a check on the Nyquist velocities that they are uniform in the sweep, False will skip this check and return the velocity of the first ray in the sweep.

Returns

nyquist_velocity [float] Array containing the Nyquist velocity in m/s for a given sweep.

get_slice(*self*, *sweep*)
Return a slice for selecting rays for a given sweep.

get_start(*self*, *sweep*)
Return the starting ray index for a given sweep.

get_start_end(*self*, *sweep*)
Return the starting and ending ray for a given sweep.

info(*self*, *level='standard'*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Print information on radar.

Parameters

level [{'compact', 'standard', 'full', 'c', 's', 'f'}] Level of information on radar object to print, compact is minimal information, standard more and full everything.

out [file-like] Stream to direct output to, default is to print information to standard out (the screen).

init_gate_altitude(*self*)
Initialize the gate_altitude attribute.

init_gate_longitude_latitude(*self*)
Initialize or reset the gate_longitude and gate_latitude attributes.

init_gate_x_y_z (*self*)
Initialize or reset the gate_{x, y, z} attributes.

init_rays_per_sweep (*self*)
Initialize or reset the rays_per_sweep attribute.

iter_azimuth (*self*)
Return an iterator which returns sweep azimuth data.

iter_elevation (*self*)
Return an iterator which returns sweep elevation data.

iter_end (*self*)
Return an iterator over the sweep end indices.

iter_field (*self*, *field_name*)
Return an iterator which returns sweep field data.

iter_slice (*self*)
Return an iterator which returns sweep slice objects.

iter_start (*self*)
Return an iterator over the sweep start indices.

iter_start_end (*self*)
Return an iterator over the sweep start and end indices.

pyart.core.**antenna_to_cartesian** (*ranges*, *azimuths*, *elevations*)
Return Cartesian coordinates from antenna coordinates.

Parameters

ranges [array] Distances to the center of the radar gates (bins) in kilometers.

azimuths [array] Azimuth angle of the radar in degrees.

elevations [array] Elevation angle of the radar in degrees.

Returns

x, y, z [array] Cartesian coordinates in meters from the radar.

Notes

The calculation for Cartesian coordinate is adapted from equations 2.28(b) and 2.28(c) of Doviak and Zrnic [1] assuming a standard atmosphere (4/3 Earth's radius model).

$$z = \sqrt{r^2 + R^2 + 2 * r * R * sin(\theta_e)} - R$$
$$s = R * arcsin(\frac{r * cos(\theta_e)}{R + z})$$
$$x = s * sin(\theta_a)$$
$$y = s * cos(\theta_a)$$

Where r is the distance from the radar to the center of the gate, θ_a is the azimuth angle, θ_e is the elevation angle, s is the arc length, and R is the effective radius of the earth, taken to be 4/3 the mean radius of earth (6371 km).

References

[1]

pyart.core.**antenna_vectors_to_cartesian**(*ranges*, *azimuths*, *elevations*, *edges=False*)
Calculate Cartesian coordinate for gates from antenna coordinate vectors.

Calculates the Cartesian coordinates for the gate centers or edges for all gates from antenna coordinate vectors assuming a standard atmosphere (4/3 Earth's radius model). See `pyart.util.antenna_to_cartesian()` for details.

Parameters

ranges [array, 1D.] Distances to the center of the radar gates (bins) in meters.

azimuths [array, 1D.] Azimuth angles of the rays in degrees.

elevations [array, 1D.] Elevation angles of the rays in degrees.

edges [bool, optional] True to calculate the coordinates of the gate edges by interpolating between gates and extrapolating at the boundaries. False to calculate the gate centers.

Returns

x, y, z [array, 2D] Cartesian coordinates in meters from the center of the radar to the gate centers or edges.

pyart.core.**cartesian_to_antenna**(*x*, *y*, *z*)
Returns antenna coordinates from Cartesian coordinates.

Parameters

x, y, z [array] Cartesian coordinates in meters from the radar.

Returns

ranges [array] Distances to the center of the radar gates (bins) in m.

azimuths [array] Azimuth angle of the radar in degrees. [-180., 180]

elevations [array] Elevation angle of the radar in degrees.

pyart.core.**cartesian_to_geographic**(*x*, *y*, *projparams*)
Cartesian to Geographic coordinate transform.

Transform a set of Cartesian/Cartographic coordinates (x, y) to a geographic coordinate system (lat, lon) using pyproj or a build in Azimuthal equidistant projection.

Parameters

x, y [array-like] Cartesian coordinates in meters unless R is defined in different units in the projparams parameter.

projparams [dict or str] Projection parameters passed to pyproj.Proj. If this parameter is a dictionary with a 'proj' key equal to 'pyart_aeqd' then a azimuthal equidistant projection will be used that is native to Py-ART and does not require pyproj to be installed. In this case a non-default value of R can be specified by setting the 'R' key to the desired value.

Returns

lon, lat [array] Longitude and latitude of the Cartesian coordinates in degrees.

pyart.core.**cartesian_to_geographic_aeqd**(*x*, *y*, *lon_0*, *lat_0*, *R=6370997.0*)
Azimuthal equidistant Cartesian to geographic coordinate transform.

Transform a set of Cartesian/Cartographic coordinates (x, y) to geographic coordinate system (lat, lon) using a

azimuthal equidistant map projection [1].

$$lat = \arcsin(\cos(c) * \sin(lat_0) + (y * \sin(c) * \cos(lat_0)/\rho))$$
$$lon = lon_0 + \arctan 2(x * \sin(c), \rho * \cos(lat_0) * \cos(c) - y * \sin(lat_0) * \sin(c))$$
$$\rho = \sqrt{(x^2 + y^2)}$$
$$c = \rho/R$$

Where x, y are the Cartesian position from the center of projection; lat, lon the corresponding latitude and longitude; lat_0, lon_0 are the latitude and longitude of the center of the projection; R is the radius of the earth (defaults to ~6371 km). lon is adjusted to be between -180 and 180.

Parameters

x, y [array-like] Cartesian coordinates in the same units as R, typically meters.

lon_0, lat_0 [float] Longitude and latitude, in degrees, of the center of the projection.

R [float, optional] Earth radius in the same units as x and y. The default value is in units of meters.

Returns

lon, lat [array] Longitude and latitude of Cartesian coordinates in degrees.

References

[1]

pyart.core.**cartesian_vectors_to_geographic**(*x*, *y*, *projparams*, *edges=False*)

Cartesian vectors to Geographic coordinate transform.

Transform a set of Cartesian/Cartographic coordinate vectors (x, y) to a geographic coordinate system (lat, lon) using pyproj or a build in Azimuthal equidistant projection finding the coordinates edges in Cartesian space if requested.

Parameters

x, y [array 1D.] Cartesian coordinate vectors in meters unless R is defined in different units in the projparams parameter.

projparams [dict or str] Projection parameters passed to pyproj.Proj. If this parameter is a dictionary with a 'proj' key equal to 'pyart_aeqd' then a azimuthal equidistant projection will be used that is native to Py-ART and does not require pyproj to be installed. In this case a non-default value of R can be specified by setting the 'R' key to the desired value.

edges [bool, optional] True to calculate the coordinates of the geographic edges by interpolating between Cartesian points and extrapolating at the boundaries. False to calculate the coordinate centers.

Returns

lon, lat [array] Longitude and latitude of the Cartesian coordinates in degrees.

pyart.core.**geographic_to_cartesian**(*lon*, *lat*, *projparams*)

Geographic to Cartesian coordinate transform.

Transform a set of Geographic coordinate (lat, lon) to a Cartesian/Cartographic coordinate (x, y) using pyproj or a build in Azimuthal equidistant projection.

Parameters

lon, lat [array-like] Geographic coordinates in degrees.

projparams [dict or str] Projection parameters passed to pyproj.Proj. If this parameter is a dictionary with a 'proj' key equal to 'pyart_aeqd' then a azimuthal equidistant projection will be used that is native to Py-ART and does not require pyproj to be installed. In this case a non-default value of R can be specified by setting the 'R' key to the desired value.

Returns

x, y [array-like] Cartesian coordinates in meters unless projparams defines a value for R in different units.

pyart.core.**geographic_to_cartesian_aeqd**(*lon*, *lat*, *lon_0*, *lat_0*, *R=6370997.0*)

Azimuthal equidistant geographic to Cartesian coordinate transform.

Transform a set of geographic coordinates (lat, lon) to Cartesian/Cartographic coordinates (x, y) using a azimuthal equidistant map projection [1].

$$x = R * k * \cos(lat) * \sin(lon - lon_0)$$
$$y = R * k * [\cos(lat_0) * \sin(lat) - \sin(lat_0) * \cos(lat) * \cos(lon - lon_0)]$$
$$k = c/\sin(c)$$
$$c = \arccos(\sin(lat_0) * \sin(lat) + \cos(lat_0) * \cos(lat) * \cos(lon - lon_0))$$

Where x, y are the Cartesian position from the center of projection; lat, lon the corresponding latitude and longitude; lat_0, lon_0 are the latitude and longitude of the center of the projection; R is the radius of the earth (defaults to ~6371 km).

Parameters

lon, lat [array-like] Longitude and latitude coordinates in degrees.

lon_0, lat_0 [float] Longitude and latitude, in degrees, of the center of the projection.

R [float, optional] Earth radius in the same units as x and y. The default value is in units of meters.

Returns

x, y [array] Cartesian coordinates in the same units as R, typically meters.

References

[1]

pyart.core.**wgs84_to_swissCH1903**(*lon*, *lat*, *alt*, *no_altitude_transform=False*)

Convert WGS84 coordinates to swiss coordinates (CH1903 / LV03)

The formulas for the coordinates transformation are taken from: "Formeln und Konstanten für die Berechnung der Schweizerischen schiefachsigen Zylinderprojektion und der Transformation zwischen Koordinatensystemen", chapter 4. "Näherungslösungen CH1903 <=> WGS84" Bundesamt für Landestopografie swisstopo (http://www.swisstopo.admin.ch), Oktober 2008

Parameters

lon, lat [array-like] Geographic coordinates WGS84 in degrees.

alt [array-like] Altitude in m

no_altitude_transform [bool] If set, do not convert altitude

Returns

chy, chx, chh [array-like] Coordinates in swiss CH1903 coordinates in meter

CHAPTER

FOUR

BRIDGING TO OTHER TOOLKITS (`PYART.BRIDGE`)

Py-ART can act as bridge to other community software projects.

The functionality in this namespace is available in other pyart namespaces.

4.1 Phase functions

`texture_of_complex_phase`(radar[, ...])	Calculate the texture of the differential phase field.

CHAPTER

FIVE

FILTERS (PYART.FILTERS)

Classes for specifying what gates are included and excluded from routines.

5.1 Filtering radar data

GateFilter(radar[, exclude_based])	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.
moment_based_gate_filter(radar[, ncp_field, ...])	Create a filter which removes undesired gates based on moments.
moment_and_texture_based_gate_filter(radar[, ...])	Create a filter which removes undesired gates based on texture of moments.
snr_based_gate_filter(radar[, snr_field, ...])	Create a filter which removes undesired gates based on SNR.
visibility_based_gate_filter(radar[, ...])	Create a filter which removes undesired gates based on visibility.
class_based_gate_filter(radar[, field, ...])	Create a filter which removes undesired gates based on class values
temp_based_gate_filter(radar[, temp_field, ...])	Create a filter which removes undesired gates based on temperature.
iso0_based_gate_filter(radar[, iso0_field, ...])	Create a filter which removes undesired gates based height over the iso0.
birds_gate_filter(radar[, zdr_field, ...])	Create a filter which removes data not suspected of being birds

class pyart.filters.**GateFilter**(*radar*, *exclude_based=True*)

Bases: object

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. These filter can be used in various algorithms and calculations within Py-ART.

See *pyart.correct.GateFilter.exclude_below()* for method parameter details.

Parameters

radar [Radar] Radar object from which gate filter will be build.

exclude_based [bool, optional] True, the default and suggested method, will begin with all gates included and then use the exclude methods to exclude gates based on conditions. False will begin with all gates excluded from which a set of gates to include should be set using the include methods.

Examples

```
>>> import pyart
>>> radar = pyart.io.read('radar_file.nc')
>>> gatefilter = pyart.correct.GateFilter(radar)
>>> gatefilter.exclude_below('reflectivity', 10)
>>> gatefilter.exclude_below('normalized_coherent_power', 0.75)
```

Attributes

gate_excluded [array, dtype=bool] Return a copy of the excluded gates.

gate_included [array, dtype=bool] Return a copy of the included gates.

Methods

copy(self)	Return a copy of the gatefilter.
exclude_above(self, field, value[, ...])	Exclude gates where a given field is above a given value.
exclude_all(self)	Exclude all gates.
exclude_below(self, field, value[, ...])	Exclude gates where a given field is below a given value.
exclude_equal(self, field, value[, ...])	Exclude gates where a given field is equal to a value.
exclude_gates(self, mask[, exclude_masked, op])	Exclude gates where a given mask is equal True.
exclude_inside(self, field, v1, v2[, ...])	Exclude gates where a given field is inside a given interval.
exclude_invalid(self, field[, ...])	Exclude gates where an invalid value occurs in a field (NaNs or infs).
exclude_masked(self, field[, exclude_masked, op])	Exclude gates where a given field is masked.
exclude_none(self)	Exclude no gates, include all gates.
exclude_not_equal(self, field, value[, ...])	Exclude gates where a given field is not equal to a value.
exclude_outside(self, field, v1, v2[, ...])	Exclude gates where a given field is outside a given interval.
exclude_transition(self[, trans_value, ...])	Exclude all gates in rays marked as in transition between sweeps.
include_above(self, field, value[, ...])	Include gates where a given field is above a given value.
include_all(self)	Include all gates.
include_below(self, field, value[, ...])	Include gates where a given field is below a given value.
include_equal(self, field, value[, ...])	Include gates where a given field is equal to a value.
include_gates(self, mask[, exclude_masked, op])	Include gates where a given mask is equal True.
include_inside(self, field, v1, v2[, ...])	Include gates where a given field is inside a given interval.
include_none(self)	Include no gates, exclude all gates.
include_not_equal(self, field, value[, ...])	Include gates where a given field is not equal to a value.

Continued on next page

Table 2 – continued from previous page

include_not_masked(self, field[, ...])	Include gates where a given field in not masked.
include_not_transition(self[, trans_value, ...])	Include all gates in rays not marked as in transition between sweeps.
include_outside(self, field, v1, v2[, ...])	Include gates where a given field is outside a given interval.
include_valid(self, field[, exclude_masked, op])	Include gates where a valid value occurs in a field (not NaN or inf).

__class__
alias of `builtins.type`

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.filters.gatefilter', '__doc__': "\n A

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *radar*, *exclude_based=True*)
initialize

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, */*)
Return self<=value.

__lt__(*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.filters.gatefilter'

__ne__(*self*, *value*, */*)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, */*)
Helper for pickle.

__reduce_ex__ (*self*, *protocol*, */*)
Helper for pickle.

__repr__ (*self*, */*)
Return repr(self).

__setattr__ (*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__sizeof__ (*self*, */*)
Size of object in memory, in bytes.

__str__ (*self*, */*)
Return str(self).

__subclasshook__ ()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_get_fdata (*self*, *field*)
Check that the field exists and retrieve field data.

_merge (*self*, *marked*, *op*, *exclude_masked*)
Merge an array of marked gates with the exclude array.

copy (*self*)
Return a copy of the gatefilter.

exclude_above (*self*, *field*, *value*, *exclude_masked=True*, *op='or'*, *inclusive=False*)
Exclude gates where a given field is above a given value.

exclude_all (*self*)
Exclude all gates.

exclude_below (*self*, *field*, *value*, *exclude_masked=True*, *op='or'*, *inclusive=False*)
Exclude gates where a given field is below a given value.

Parameters

field [str] Name of field compared against the value.

value [float] Gates with a value below this value in the specified field will be marked for exclusion in the filter.

exclude_masked [bool, optional] True to filter masked values in the specified field if the data is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'and' method MAY results in including gates which have previously been excluded because they were masked or invalid.

inclusive [bool] Indicates whether the specified value should also be excluded.

exclude_equal(*self*, *field*, *value*, *exclude_masked=True*, *op='or'*)
Exclude gates where a given field is equal to a value.

exclude_gates(*self*, *mask*, *exclude_masked=True*, *op='or'*)
Exclude gates where a given mask is equal True.

Parameters

mask [numpy array] Boolean numpy array with same shape as a field array.

exclude_masked [bool, optional] True to filter masked values in the specified mask if it is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'and' method MAY results in including gates which have previously been excluded because they were masked or invalid.

exclude_inside(*self*, *field*, *v1*, *v2*, *exclude_masked=True*, *op='or'*, *inclusive=True*)
Exclude gates where a given field is inside a given interval.

exclude_invalid(*self*, *field*, *exclude_masked=True*, *op='or'*)
Exclude gates where an invalid value occurs in a field (NaNs or infs).

exclude_masked(*self*, *field*, *exclude_masked=True*, *op='or'*)
Exclude gates where a given field is masked.

exclude_none(*self*)
Exclude no gates, include all gates.

exclude_not_equal(*self*, *field*, *value*, *exclude_masked=True*, *op='or'*)
Exclude gates where a given field is not equal to a value.

exclude_outside(*self*, *field*, *v1*, *v2*, *exclude_masked=True*, *op='or'*, *inclusive=False*)
Exclude gates where a given field is outside a given interval.

exclude_transition(*self*, *trans_value=1*, *exclude_masked=True*, *op='or'*)
Exclude all gates in rays marked as in transition between sweeps.

Exclude all gates in rays marked as "in transition" by the antenna_transition attribute of the radar used to construct the filter. If no antenna transition information is available no gates are excluded.

Parameters

trans_value [int, optional] Value used in the antenna transition data to indicate that the instrument was between sweeps (in transition) during the collection of a specific ray. Typically a value of 1 is used to indicate this transition and the default can be used in these cases.

exclude_masked [bool, optional] True to filter masked values in antenna_transition if the data is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when

building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'and' method MAY results in including gates which have previously been excluded because they were masked or invalid.

gate_excluded
Return a copy of the excluded gates.

gate_included
Return a copy of the included gates.

include_above(*self*, *field*, *value*, *exclude_masked=True*, *op='and'*, *inclusive=False*)
Include gates where a given field is above a given value.

include_all(*self*)
Include all gates.

include_below(*self*, *field*, *value*, *exclude_masked=True*, *op='and'*, *inclusive=False*)
Include gates where a given field is below a given value.

include_equal(*self*, *field*, *value*, *exclude_masked=True*, *op='and'*)
Include gates where a given field is equal to a value.

include_gates(*self*, *mask*, *exclude_masked=True*, *op='and'*)
Include gates where a given mask is equal True.

Parameters

mask [numpy array] Boolean numpy array with same shape as a field array.

exclude_masked [bool, optional] True to filter masked values in the specified mask if it is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'or' method MAY results in excluding gates which have previously been included.

include_inside(*self*, *field*, *v1*, *v2*, *exclude_masked=True*, *op='and'*, *inclusive=True*)
Include gates where a given field is inside a given interval.

include_none(*self*)
Include no gates, exclude all gates.

include_not_equal(*self*, *field*, *value*, *exclude_masked=True*, *op='and'*)
Include gates where a given field is not equal to a value.

include_not_masked(*self*, *field*, *exclude_masked=True*, *op='and'*)
Include gates where a given field in not masked.

include_not_transition(*self*, *trans_value=0*, *exclude_masked=True*, *op='and'*)
Include all gates in rays not marked as in transition between sweeps.

Include all gates in rays not marked as "in transition" by the antenna_transition attribute of the radar used to construct the filter. If no antenna transition information is available all gates are included.

Parameters

trans_value [int, optional] Value used in the antenna transition data to indicate that the instrument is not between sweeps (in transition) during the collection of a specific ray. Typically a value of 0 is used to indicate no transition and the default can be used in these cases.

exclude_masked [bool, optional] True to filter masked values in antenna_transition if the data is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'or' method MAY results in excluding gates which have previously been included.

include_outside(*self*, *field*, *v1*, *v2*, *exclude_masked=True*, *op='and'*, *inclusive=False*)
Include gates where a given field is outside a given interval.

include_valid(*self*, *field*, *exclude_masked=True*, *op='and'*)
Include gates where a valid value occurs in a field (not NaN or inf).

pyart.filters.**birds_gate_filter**(*radar*, *zdr_field=None*, *rhv_field=None*, *refl_field=None*, *vel_field=None*, *max_zdr=3.0*, *max_rhv=0.9*, *min_refl=0.0*, *max_refl=20.0*, *vel_lim=1.0*, *rmin=2000.0*, *rmax=25000.0*, *elmin=1.0*, *elmax=85.0*)
Create a filter which removes data not suspected of being birds

Creates a gate filter in which the following gates are excluded:

- Gates where the instrument is transitioning between sweeps.
- Gates where the reflectivity is beyond min_refl and max_refl
- Gates where the co-polar correlation coefficient is above max_rhv
- Gates where the differential reflectivity is above max_zdr
- Gates where the Doppler velocity is within the interval given by +-vel_lim
- Gates where any of the above fields are masked or contain invalid values (NaNs or infs).
- Gates outside the range given by range min and range max
- If any of these three fields do not exist in the radar that fields filter criteria is not applied.

Parameters

radar [Radar] Radar object from which the gate filter will be built.

refl_field, zdr_field, rhv_field, vel_field [str] Names of the radar fields which contain the reflectivity, differential reflectivity, co-polar correlation coefficient, and Doppler velocity from which the gate filter will be created using the above criteria. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

max_zdr, max_rhv [float] Maximum values for the differential reflectivity and co-polar correlation coefficient. Gates in these fields above these limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the given field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value above the highest value in the field.

min_refl, max_refl [float] Minimum and maximum values for the reflectivity. Gates outside of this interval as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use this filter. A value or None for one of these parameters will disable the minimum or maximum filtering but retain the other. A value of None for both of these values will disable all filtering based upon the reflectivity including removing masked or gates with an invalid value. To disable the interval filtering but retain the masked and invalid filter set the parameters to values above and below the lowest and greatest values in the reflectivity field.

rmin, rmax [float] Minimum and maximum ranges [m]

elmin, elmax [float] Minimum and maximum elevations [deg]

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

pyart.filters.**class_based_gate_filter**(*radar*, *field=None*, *kept_values=None*)

Create a filter which removes undesired gates based on class values

Parameters

radar [Radar] Radar object from which the gate filter will be built.

field [str] Name of the radar field which contains the classification. A value of None for will use the default field name for the hydrometeor classification as defined in the Py-ART configuration file.

kept_values [list of ints or none] The class values to keep

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

pyart.filters.**iso0_based_gate_filter**(*radar*, *iso0_field=None*, *max_h_iso0=0.0*, *thickness=400.0*, *beamwidth=None*)

Create a filter which removes undesired gates based height over the iso0. Used primarily to filter out the melting layer and gates above it.

Parameters

radar [Radar] Radar object from which the gate filter will be built.

iso0_field [str] Name of the radar field which contains the height relative to the iso0. A value of None for will use the default field name as defined in the Py-ART configuration file.

max_h_iso0 [float] Maximum height relative to the iso0 in m. Gates below this limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value below the lowest value in the field.

thickness [float] The estimated thickness of the melting layer in m.

beamwidth [float] The radar antenna 3 dB beamwidth [deg].

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

pyart.filters.**moment_and_texture_based_gate_filter**(*radar*, *zdr_field=None*, *rhv_field=None*, *phi_field=None*, *refl_field=None*, *textzdr_field=None*, *textrhv_field=None*, *textphi_field=None*, *textrefl_field=None*, *wind_size=7*, *max_textphi=20.0*, *max_textrhv=0.3*, *max_textzdr=2.85*, *max_textrefl=8.0*, *min_rhv=0.6*)

Create a filter which removes undesired gates based on texture of moments.

Creates a gate filter in which the following gates are excluded: * Gates where the instrument is transitioning between sweeps. * Gates where RhoHV is below min_rhv * Gates where the PhiDP texture is above max_textphi. * Gates where the RhoHV texture is above max_textrhv. * Gates where the ZDR texture is above max_textzdr * Gates where the reflectivity texture is above max_textrefl * If any of the thresholds is not set or the field (RhoHV, ZDR, PhiDP, reflectivity) do not exist in the radar the filter is not applied.

Parameters

radar [Radar] Radar object from which the gate filter will be built.

zdr_field, rhv_field, phi_field, refl_field [str] Names of the radar fields which contain the differential reflectivity, cross correlation ratio, differential phase and reflectivity from which the textures will be computed. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

textzdr_field, textrhv_field, textphi_field, textrefl_field [str] Names of the radar fields given to the texture of the differential reflectivity, texture of the cross correlation ratio, texture of differential phase and texture of reflectivity. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

wind_size [int] Size of the moving window used to compute the ray texture.

max_textphi, max_textrhv, max_textzdr, max_textrefl [float] Maximum value for the texture of the differential phase, texture of RhoHV, texture of Zdr and texture of reflectivity. Gates in these fields above these limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the given field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value above the highest value in the field.

min_rhv [float] Minimum value for the RhoHV. Gates below this limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the given field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value below the lowest value in the field.

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

pyart.filters.**moment_based_gate_filter**(*radar*, *ncp_field=None*, *rhv_field=None*, *refl_field=None*, *min_ncp=0.5*, *min_rhv=None*, *min_refl=-20.0*, *max_refl=100.0*)

Create a filter which removes undesired gates based on moments.

Creates a gate filter in which the following gates are excluded:

- Gates where the instrument is transitioning between sweeps.
- Gates where the reflectivity is outside the interval min_refl, max_refl.
- Gates where the normalized coherent power is below min_ncp.
- Gates where the cross correlation ratio is below min_rhi. Using the default parameter this filtering is disabled.
- Gates where any of the above three fields are masked or contain invalid values (NaNs or infs).
- If any of these three fields do not exist in the radar that fields filter criteria is not applied.

Parameters

radar [Radar] Radar object from which the gate filter will be built.

refl_field, ncp_field, rhv_field [str] Names of the radar fields which contain the reflectivity, normalized coherent power (signal quality index) and cross correlation ratio (RhoHV) from which the gate filter will be created using the above criteria. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

min_ncp, min_rhv [float] Minimum values for the normalized coherence power and cross correlation ratio. Gates in these fields below these limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the given field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value below the lowest value in the field.

min_refl, max_refl [float] Minimum and maximum values for the reflectivity. Gates outside of this interval as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use this filter. A value or None for one of these parameters will disable the minimum or maximum filtering but retain the other. A value of None for both of these values will disable all filtering based upon the reflectivity including removing masked or gates with an invalid value. To disable the interval filtering but retain the masked and invalid filter set the parameters to values above and below the lowest and greatest values in the reflectivity field.

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

pyart.filters.**snr_based_gate_filter**(*radar*, *snr_field=None*, *min_snr=10.0*, *max_snr=None*)

Create a filter which removes undesired gates based on SNR.

Parameters

radar [Radar] Radar object from which the gate filter will be built.

snr_field [str] Name of the radar field which contains the signal to noise ratio. A value of None for will use the default field name as defined in the Py-ART configuration file.

min_snr [float] Minimum value for the SNR. Gates below this limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value below the lowest value in the field.

max_snr [float] Maximum value for the SNR

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

pyart.filters.**temp_based_gate_filter**(*radar*, *temp_field=None*, *min_temp=0.0*, *thickness=400.0*, *beamwidth=None*)

Create a filter which removes undesired gates based on temperature. Used primarily to filter out the melting layer and gates above it.

Parameters

radar [Radar] Radar object from which the gate filter will be built.

temp_field [str] Name of the radar field which contains the temperature. A value of None for will use the default field name as defined in the Py-ART configuration file.

min_temp [float] Minimum value for the temperature in degrees. Gates below this limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value below the lowest value in the field.

thickness [float] The estimated thickness of the melting layer in m.

beamwidth [float] The radar antenna 3 dB beamwidth [deg].

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

pyart.filters.**visibility_based_gate_filter**(*radar*, *vis_field=None*, *min_vis=10.0*)

Create a filter which removes undesired gates based on visibility.

Parameters

radar [Radar] Radar object from which the gate filter will be built.

vis_field [str] Name of the radar field which contains the visibility. A value of None for will use the default field name as defined in the Py-ART configuration file.

min_vis [float] Minimum value for the visibility. Gates below this limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value below the lowest value in the field.

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

CHAPTER

SIX

RADAR CORRECTIONS (pyart.correct)

Correct radar fields.

6.1 Velocity unfolding

dealias_fourdd(radar[, last_radar, ...])	Dealias Doppler velocities using the 4DD algorithm.
dealias_unwrap_phase(radar[, unwrap_unit, ...])	Dealias Doppler velocities using multi-dimensional phase unwrapping.
dealias_region_based(radar[, ref_vel_field, ...])	Dealias Doppler velocities using a region based algorithm.

6.2 Other corrections

calculate_attenuation(radar, z_offset[, ...])	Calculate the attenuation from a polarimetric radar using Z-PHI method.
calculate_attenuation_zphi(radar[, doc, ...])	Calculate the attenuation and the differential attenuation from a polarimetric radar using Z-PHI method..
calculate_attenuation_philinear(radar[, ...])	Calculate the attenuation and the differential attenuation from a polarimetric radar using linear dependece with PhiDP.
phase_proc_lp(radar, offset[, debug, ...])	Phase process using a LP method [1].
det_sys_phase_ray(radar[, ind_rmin, ...])	Public method Alternative determination of the system phase.
correct_sys_phase(radar[, ind_rmin, ...])	correction of the system offset.
smooth_phidp_single_window(radar[, ...])	correction of the system offset and smoothing using one window
smooth_phidp_double_window(radar[, ...])	correction of the system offset and smoothing using two window
despeckle_field(radar, field[, label_dict, ...])	Despeckle a radar volume by identifying small objects in each scan and masking them out.
correct_noise_rhohv(radar[, urhohv_field, ...])	Corrects RhoHV for noise according to eq.
correct_bias(radar[, bias, field_name])	Corrects a radar data bias.
correct_visibility(radar[, vis_field, ...])	Corrects the reflectivity according to visibility.
est_rhohv_rain(radar[, ind_rmin, ind_rmax, ...])	Estimates the quantiles of RhoHV in rain for each sweep

Continued on next page

Table 2 – continued from previous page

est_zdr_precip(radar[, ind_rmin, ind_rmax, ...])	Filters out all undesired data to be able to estimate ZDR bias, either in moderate rain or from vertically pointing scans
est_zdr_snow(radar[, ind_rmin, ind_rmax, ...])	Filters out all undesired data to be able to estimate ZDR bias in snow
selfconsistency_bias(radar, zdr_kdpzh_dict)	Estimates reflectivity bias at each ray using the self-consistency algorithm by Gourley
selfconsistency_kdp_phidp(radar, zdr_kdpzh_dict)	Estimates KDP and PhiDP in rain from Zh and ZDR using a selfconsistency relation between ZDR, Zh and KDP.
get_sun_hits(radar[, delev_max, dazim_max, ...])	get data from suspected sun hits
sun_retrieval(az_rad, az_sun, el_rad, ...[, ...])	Estimates sun parameters from sun hits
phase_proc_lp_gf(radar[, gatefilter, debug, ...])	Phase process using a LP method [1] using Py-ART's Gatefilter.

6.3 Helper functions

find_objects(radar, field, threshold[, ...])	Find objects (i.e., contiguous gates) in one or more sweeps that match thresholds.
get_mask_fzl(radar[, fzl, doc, min_temp, ...])	Constructs a mask to mask data placed thickness m below data at min_temp and beyond.
sun_power(solar_flux, pulse_width, wavelen, ...)	computes the theoretical sun power detected at the antenna [dBm] as it would be without atmospheric attenuation (sun power at top of the atmosphere) for a given solar flux and radar characteristics
ptoa_to_sf(ptoa, pulse_width, wavelen, ...)	Converts the sun power at the top of the atmosphere (in dBm) into solar flux.
solar_flux_lookup(solar_flux, wavelen)	Given the observed solar flux at 10.7 cm wavelength, returns the solar flux at the given radar wavelength
scanning_losses(angle_step, beamwidth)	Given the antenna beam width and the integration angle, compute the losses due to the fact that the sun is not a point target and the antenna is scanning
smooth_masked(raw_data[, wind_len, ...])	smoothes the data using a rolling window.

class pyart.correct.**GateFilter**(*radar*, *exclude_based=True*)

Bases: object

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. These filter can be used in various algorithms and calculations within Py-ART.

See *pyart.correct.GateFilter.exclude_below()* for method parameter details.

Parameters

radar [Radar] Radar object from which gate filter will be build.

exclude_based [bool, optional] True, the default and suggested method, will begin with all gates included and then use the exclude methods to exclude gates based on conditions. False will begin with all gates excluded from which a set of gates to include should be set using the include methods.

Examples

```
>>> import pyart
>>> radar = pyart.io.read('radar_file.nc')
>>> gatefilter = pyart.correct.GateFilter(radar)
>>> gatefilter.exclude_below('reflectivity', 10)
>>> gatefilter.exclude_below('normalized_coherent_power', 0.75)
```

Attributes

gate_excluded [array, dtype=bool] Return a copy of the excluded gates.

gate_included [array, dtype=bool] Return a copy of the included gates.

Methods

copy(self)	Return a copy of the gatefilter.
exclude_above(self, field, value[, ...])	Exclude gates where a given field is above a given value.
exclude_all(self)	Exclude all gates.
exclude_below(self, field, value[, ...])	Exclude gates where a given field is below a given value.
exclude_equal(self, field, value[, ...])	Exclude gates where a given field is equal to a value.
exclude_gates(self, mask[, exclude_masked, op])	Exclude gates where a given mask is equal True.
exclude_inside(self, field, v1, v2[, ...])	Exclude gates where a given field is inside a given interval.
exclude_invalid(self, field[, ...])	Exclude gates where an invalid value occurs in a field (NaNs or infs).
exclude_masked(self, field[, exclude_masked, op])	Exclude gates where a given field is masked.
exclude_none(self)	Exclude no gates, include all gates.
exclude_not_equal(self, field, value[, ...])	Exclude gates where a given field is not equal to a value.
exclude_outside(self, field, v1, v2[, ...])	Exclude gates where a given field is outside a given interval.
exclude_transition(self[, trans_value, ...])	Exclude all gates in rays marked as in transition between sweeps.
include_above(self, field, value[, ...])	Include gates where a given field is above a given value.
include_all(self)	Include all gates.
include_below(self, field, value[, ...])	Include gates where a given field is below a given value.
include_equal(self, field, value[, ...])	Include gates where a given field is equal to a value.
include_gates(self, mask[, exclude_masked, op])	Include gates where a given mask is equal True.
include_inside(self, field, v1, v2[, ...])	Include gates where a given field is inside a given interval.
include_none(self)	Include no gates, exclude all gates.
include_not_equal(self, field, value[, ...])	Include gates where a given field is not equal to a value.

Continued on next page

Table 4 – continued from previous page

include_not_masked(self, field[, ...])	Include gates where a given field in not masked.
include_not_transition(self[, trans_value, ...])	Include all gates in rays not marked as in transition between sweeps.
include_outside(self, field, v1, v2[, ...])	Include gates where a given field is outside a given interval.
include_valid(self, field[, exclude_masked, op])	Include gates where a valid value occurs in a field (not NaN or inf).

__class__
alias of `builtins.type`

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.filters.gatefilter', '__doc__': "\n A

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *radar*, *exclude_based=True*)
initialize

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, */*)
Return self<=value.

__lt__(*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.filters.gatefilter'

__ne__(*self*, *value*, */*)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, */*)
Helper for pickle.

__reduce_ex__ (*self*, *protocol*, */*)
Helper for pickle.

__repr__ (*self*, */*)
Return repr(self).

__setattr__ (*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__sizeof__ (*self*, */*)
Size of object in memory, in bytes.

__str__ (*self*, */*)
Return str(self).

__subclasshook__ ()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_get_fdata (*self*, *field*)
Check that the field exists and retrieve field data.

_merge (*self*, *marked*, *op*, *exclude_masked*)
Merge an array of marked gates with the exclude array.

copy (*self*)
Return a copy of the gatefilter.

exclude_above (*self*, *field*, *value*, *exclude_masked=True*, *op='or'*, *inclusive=False*)
Exclude gates where a given field is above a given value.

exclude_all (*self*)
Exclude all gates.

exclude_below (*self*, *field*, *value*, *exclude_masked=True*, *op='or'*, *inclusive=False*)
Exclude gates where a given field is below a given value.

Parameters

field [str] Name of field compared against the value.

value [float] Gates with a value below this value in the specified field will be marked for exclusion in the filter.

exclude_masked [bool, optional] True to filter masked values in the specified field if the data is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'and' method MAY results in including gates which have previously been excluded because they were masked or invalid.

inclusive [bool] Indicates whether the specified value should also be excluded.

exclude_equal(*self*, *field*, *value*, *exclude_masked=True*, *op='or'*)
Exclude gates where a given field is equal to a value.

exclude_gates(*self*, *mask*, *exclude_masked=True*, *op='or'*)
Exclude gates where a given mask is equal True.

Parameters

mask [numpy array] Boolean numpy array with same shape as a field array.

exclude_masked [bool, optional] True to filter masked values in the specified mask if it is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'and' method MAY results in including gates which have previously been excluded because they were masked or invalid.

exclude_inside(*self*, *field*, *v1*, *v2*, *exclude_masked=True*, *op='or'*, *inclusive=True*)
Exclude gates where a given field is inside a given interval.

exclude_invalid(*self*, *field*, *exclude_masked=True*, *op='or'*)
Exclude gates where an invalid value occurs in a field (NaNs or infs).

exclude_masked(*self*, *field*, *exclude_masked=True*, *op='or'*)
Exclude gates where a given field is masked.

exclude_none(*self*)
Exclude no gates, include all gates.

exclude_not_equal(*self*, *field*, *value*, *exclude_masked=True*, *op='or'*)
Exclude gates where a given field is not equal to a value.

exclude_outside(*self*, *field*, *v1*, *v2*, *exclude_masked=True*, *op='or'*, *inclusive=False*)
Exclude gates where a given field is outside a given interval.

exclude_transition(*self*, *trans_value=1*, *exclude_masked=True*, *op='or'*)
Exclude all gates in rays marked as in transition between sweeps.

Exclude all gates in rays marked as "in transition" by the antenna_transition attribute of the radar used to construct the filter. If no antenna transition information is available no gates are excluded.

Parameters

trans_value [int, optional] Value used in the antenna transition data to indicate that the instrument was between sweeps (in transition) during the collection of a specific ray. Typically a value of 1 is used to indicate this transition and the default can be used in these cases.

exclude_masked [bool, optional] True to filter masked values in antenna_transition if the data is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when

building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'and' method MAY results in including gates which have previously been excluded because they were masked or invalid.

gate_excluded
Return a copy of the excluded gates.

gate_included
Return a copy of the included gates.

include_above(*self*, *field*, *value*, *exclude_masked=True*, *op='and'*, *inclusive=False*)
Include gates where a given field is above a given value.

include_all(*self*)
Include all gates.

include_below(*self*, *field*, *value*, *exclude_masked=True*, *op='and'*, *inclusive=False*)
Include gates where a given field is below a given value.

include_equal(*self*, *field*, *value*, *exclude_masked=True*, *op='and'*)
Include gates where a given field is equal to a value.

include_gates(*self*, *mask*, *exclude_masked=True*, *op='and'*)
Include gates where a given mask is equal True.

Parameters

mask [numpy array] Boolean numpy array with same shape as a field array.

exclude_masked [bool, optional] True to filter masked values in the specified mask if it is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'or' method MAY results in excluding gates which have previously been included.

include_inside(*self*, *field*, *v1*, *v2*, *exclude_masked=True*, *op='and'*, *inclusive=True*)
Include gates where a given field is inside a given interval.

include_none(*self*)
Include no gates, exclude all gates.

include_not_equal(*self*, *field*, *value*, *exclude_masked=True*, *op='and'*)
Include gates where a given field is not equal to a value.

include_not_masked(*self*, *field*, *exclude_masked=True*, *op='and'*)
Include gates where a given field in not masked.

include_not_transition(*self*, *trans_value=0*, *exclude_masked=True*, *op='and'*)
Include all gates in rays not marked as in transition between sweeps.

Include all gates in rays not marked as "in transition" by the antenna_transition attribute of the radar used to construct the filter. If no antenna transition information is available all gates are included.

Parameters

trans_value [int, optional] Value used in the antenna transition data to indicate that the instrument is not between sweeps (in transition) during the collection of a specific ray. Typically a value of 0 is used to indicate no transition and the default can be used in these cases.

exclude_masked [bool, optional] True to filter masked values in antenna_transition if the data is a masked array, False to include any masked values.

op [{'and', 'or', 'new'}] Operation to perform when merging the existing set of excluded gates with the excluded gates from the current operation. 'and' will perform a logical AND operation, 'or' a logical OR, and 'new' will replace the existing excluded gates with the one generated here. 'or', the default for exclude methods, is typically desired when building up a set of conditions for excluding gates where the desired effect is to exclude gates which meet any of the conditions. 'and', the default for include methods, is typically desired when building up a set of conditions where the desired effect is to include gates which meet any of the conditions. Note that the 'or' method MAY results in excluding gates which have previously been included.

include_outside(*self*, *field*, *v1*, *v2*, *exclude_masked=True*, *op='and'*, *inclusive=False*)
Include gates where a given field is outside a given interval.

include_valid(*self*, *field*, *exclude_masked=True*, *op='and'*)
Include gates where a valid value occurs in a field (not NaN or inf).

pyart.correct.**calculate_attenuation**(*radar*, *z_offset*, *debug=False*, *doc=15*, *fzl=4000.0*, *rhv_min=0.8*, *ncp_min=0.5*, *a_coef=0.06*, *beta=0.8*, *refl_field=None*, *ncp_field=None*, *rhv_field=None*, *phidp_field=None*, *spec_at_field=None*, *corr_refl_field=None*)
Calculate the attenuation from a polarimetric radar using Z-PHI method. Parameters ———- radar : Radar

Radar object to use for attenuation calculations. Must have copol_coeff, norm_coherent_power, proc_dp_phase_shift, reflectivity_horizontal fields.

z_offset [float] Horizontal reflectivity offset in dBZ.

debug [bool, optional] True to print debugging information, False supressed this printing.

doc [float, optional] Number of gates at the end of each ray to to remove from the calculation.

fzl [float, optional] Freezing layer, gates above this point are not included in the correction.

rhv_min [float, optional] Minimum copol_coeff value to consider valid.

ncp_min [float, optional] Minimum norm_coherent_power to consider valid.

a_coef [float, optional] A coefficient in attenuation calculation.

beta [float, optional] Beta parameter in attenuation calculation.

refl_field [str, optional] Name of the reflectivity field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

phidp_field [str, optional] Name of the differential phase field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

ncp_field [str, optional] Name of the normalized coherent power field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

zdr_field [str, optional] Name of the differential reflectivity field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file. This will only be used if it is available.

spec_at_field [str, optional] Name of the specific attenuation field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

corr_refl_field [str, optional] Name of the corrected reflectivity field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

Returns

spec_at [dict] Field dictionary containing the specific attenuation.

cor_z [dict] Field dictionary containing the corrected reflectivity.

References

Gu et al. Polarimetric Attenuation Correction in Heavy Rain at C Band, JAMC, 2011, 50, 39-58.

pyart.correct.**calculate_attenuation_philinear**(*radar*, *doc=None*, *fzl=None*, *pia_coef=None*, *gatefilter=None*, *pida_coef=None*, *refl_field=None*, *phidp_field=None*, *zdr_field=None*, *temp_field=None*, *iso0_field=None*, *spec_at_field=None*, *pia_field=None*, *corr_refl_field=None*, *spec_diff_at_field=None*, *pida_field=None*, *corr_zdr_field=None*, *temp_ref='temperature'*)

Calculate the attenuation and the differential attenuation from a polarimetric radar using linear dependece with PhiDP. The attenuation is computed up to a user defined freezing level height, where temperatures in a temperature field are positive or where the height relative to the iso0 is 0. The coefficients are either user-defined or radar frequency dependent.

Parameters

radar [Radar] Radar object to use for attenuation calculations. Must have phidp and refl fields.

doc [float, optional] Number of gates at the end of each ray to to remove from the calculation.

fzl [float, optional] Freezing layer, gates above this point are not included in the correction.

gatefilter [GateFilter, optional] The gates to exclude from the calculation. This, combined with the gates above fzl, will be excluded from the correction. Set to None to not use a gatefilter.

pia_coef [float, optional] Coefficient in path integrated attenuation calculation

pida_coeff [float, optional] Coefficient in path integrated differential attenuation calculation

refl_field [str, optional] Name of the reflectivity field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

phidp_field [str, optional] Name of the differential phase field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

zdr_field [str, optional] Name of the differential reflectivity field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file. This will only be used if it is available.

temp_field [str, optional] Name of the temperature field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

iso0_field [str, optional] Name of the field for the height above the 0C isotherm for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file. This will only be used if it is available.

spec_at_field [str, optional] Name of the specific attenuation field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

pia_field [str, optional] Name of the path integrated attenuation field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

corr_refl_field [str, optional] Name of the corrected reflectivity field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

spec_diff_at_field [str, optional] Name of the specific differential attenuation field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file. This will only be calculated if ZDR is available.

corr_zdr_field [str, optional] Name of the corrected differential reflectivity field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file. This will only be calculated if ZDR is available.

temp_ref [str, optional] The field use as reference for temperature. Can be either temperature, height_over_iso0 or fixed_fzl.

Returns

spec_at [dict] Field dictionary containing the specific attenuation.

pia_dict [dict] Field dictionary containing the path integrated attenuation.

cor_z [dict] Field dictionary containing the corrected reflectivity.

spec_diff_at [dict] Field dictionary containing the specific differential attenuation.

pida_dict [dict] Field dictionary containing the path integrated differential attenuation.

cor_zdr [dict] Field dictionary containing the corrected differential reflectivity.

pyart.correct.**calculate_attenuation_zphi**(*radar*, *doc=None*, *fzl=None*, *smooth_window_len=5*, *gatefilter=None*, *a_coef=None*, *beta=None*, *c=None*, *d=None*, *refl_field=None*, *phidp_field=None*, *zdr_field=None*, *temp_field=None*, *iso0_field=None*, *spec_at_field=None*, *pia_field=None*, *corr_refl_field=None*, *spec_diff_at_field=None*, *pida_field=None*, *corr_zdr_field=None*, *temp_ref='temperature'*)

Calculate the attenuation and the differential attenuation from a polarimetric radar using Z-PHI method.. The

attenuation is computed up to a user defined freezing level height or up to where temperatures in a temperature field are positive. The coefficients are either user-defined or radar frequency dependent.

Parameters

radar [Radar] Radar object to use for attenuation calculations. Must have phidp and refl fields.

doc [float, optional] Number of gates at the end of each ray to to remove from the calculation.

fzl [float, optional] Freezing layer, gates above this point are not included in the correction.

gatefilter [GateFilter, optional] The gates to exclude from the calculation. This, combined with the gates above fzl, will be excluded from the correction. Set to None to not use a gatefilter.

smooth_window_len [int, optional] Size, in range bins, of the smoothing window

a_coef [float, optional] A coefficient in attenuation calculation.

beta [float, optional] Beta parameter in attenuation calculation.

c, d [float, optional] coefficient and exponent of the power law that relates attenuation with differential attenuation

refl_field [str, optional] Name of the reflectivity field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

phidp_field [str, optional] Name of the differential phase field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

zdr_field [str, optional] Name of the differential reflectivity field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file. This will only be used if it is available.

temp_field [str, optional] Name of the temperature field used for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

iso0_field [str, optional] Name of the field for the height above the 0C isotherm for the attenuation correction. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file. This will only be used if it is available.

spec_at_field [str, optional] Name of the specific attenuation field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

pia_field [str, optional] Name of the path integrated attenuation field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

corr_refl_field [str, optional] Name of the corrected reflectivity field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

spec_diff_at_field [str, optional] Name of the specific differential attenuation field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file. This will only be calculated if ZDR is available.

pida_field [str, optional] Name of the path integrated differential attenuation field that will be used to fill in the metadata for the returned fields. A value of None for any of these param-

eters will use the default field names as defined in the Py-ART configuration file. This will only be calculated if ZDR is available.

corr_zdr_field [str, optional] Name of the corrected differential reflectivity field that will be used to fill in the metadata for the returned fields. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file. This will only be calculated if ZDR is available.

temp_ref [str, optional] the field use as reference for temperature. Can be either temperature, height_over_iso0 or fixed_fzl

Returns

spec_at [dict] Field dictionary containing the specific attenuation.

pia_dict [dict] Field dictionary containing the path integrated attenuation.

cor_z [dict] Field dictionary containing the corrected reflectivity.

spec_diff_at [dict] Field dictionary containing the specific differential attenuation.

pida_dict [dict] Field dictionary containing the path integrated differential attenuation.

cor_zdr [dict] Field dictionary containing the corrected differential reflectivity.

References

Gu et al. Polarimetric Attenuation Correction in Heavy Rain at C Band, JAMC, 2011, 50, 39-58.

Ryzhkov et al. Potential Utilization of Specific Attenuation for Rainfall Estimation, Mitigation of Partial Beam Blockage, and Radar Networking, JAOT, 2014, 31, 599-619.

pyart.correct.**correct_bias**(*radar*, *bias=0.0*, *field_name=None*)

Corrects a radar data bias. If field name is none the correction is applied to horizontal reflectivity by default.

Parameters

radar [Radar] Radar object.

bias [float, optional] The bias magnitude.

field_name: str, optional Names of the field to be corrected.

Returns

corrected_field [dict] The corrected field

pyart.correct.**correct_noise_rhohv**(*radar*, *urhohv_field=None*, *snr_field=None*, *zdr_field=None*, *nh_field=None*, *nv_field=None*, *rhohv_field=None*)

Corrects RhoHV for noise according to eq. 6 in Gourley et al. 2006. This correction should only be performed if noise has not been subtracted from the signal during the moments computation.

Parameters

radar [Radar] Radar object.

urhohv_field [str, optional] Name of the RhoHV uncorrected for noise field.

snr_field, zdr_field, nh_field, nv_field [str, optional] Names of the SNR, ZDR, horizontal channel noise in dBZ and vertical channel noise in dBZ used to correct RhoHV.

rhohv_field [str, optional] Name of the rhohv field to output.

Returns

rhohv [dict] Noise corrected RhoHV field.

References

Gourley et al. Data Quality of the Meteo-France C-Band Polarimetric Radar, JAOT, 23, 1340-1356

pyart.correct.**correct_sys_phase**(*radar*, *ind_rmin=10*, *ind_rmax=500*, *min_rcons=11*, *zmin=20.0*, *zmax=40.0*, *psidp_field=None*, *refl_field=None*, *phidp_field=None*)

correction of the system offset. Public method

Parameters

radar [Radar] Radar object for which to determine the system phase.

ind_rmin, ind_rmax [int] Min and max range index where to look for continuous precipitation

min_rcons [int] The minimum number of consecutive gates to consider it a rain cell.

zmin, zmax [float] Minimum and maximum reflectivity to consider it a rain cell

psidp_field [str] Field name within the radar object which represent the differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

refl_field [str] Field name within the radar object which represent the reflectivity. A value of None will use the default field name as defined in the Py-ART configuration file.

phidp_field [str] Field name within the radar object which represent the corrected differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

Returns

phidp_dict [dict] The corrected phidp field

pyart.correct.**correct_visibility**(*radar*, *vis_field=None*, *field_name=None*)

Corrects the reflectivity according to visibility. Applied to horizontal reflectivity by default

Parameters

radar [Radar] radar object

vis_field [str] the name of the visibility field

field_name: str names of the field to be corrected

Returns

corrected_field [dict] The corrected field

pyart.correct.**dealias_fourdd**(*radar*, *last_radar=None*, *sonde_profile=None*, *gatefilter=False*, *filt=1*, *rsl_badval=131072.0*, *keep_original=False*, *set_limits=True*, *vel_field=None*, *corr_vel_field=None*, *last_vel_field=None*, *debug=False*, *max_shear=0.05*, *sign=1*, ***kwargs*)

Dealias Doppler velocities using the 4DD algorithm.

Dealias the Doppler velocities field using the University of Washington 4DD algorithm utilizing information from a previous volume scan and/or sounding data. Either last_radar or sonde_profile must be provided. For best results provide both a previous volume scan and sounding data. Radar and last_radar must contain the same number of rays per sweep.

Additional arguments are passed to `_fourdd_interface.fourdd_dealias()`. These can be used to fine tune the behavior of the FourDD algorithm. See the documentation of Other Parameters for details. For the default values of these parameters see the documentation of `_fourdd_interface.fourdd_dealias()`.

Parameters

radar [Radar] Radar object to use for dealiasing. Must have a Nyquist defined in the instrument_parameters attribute and have a reflectivity_horizontal and mean_doppler_velocity fields.

last_radar [Radar, optional] The previous radar volume, which has been successfully dealiased. Using a previous volume as an initial condition can greatly improve the dealiasing, and represents the final dimension in the 4DD algorithm.

sonde_profile [HorizontalWindProfile, optional] Profile of horizontal winds from a sonding used for the initial condition of the dealiasing.

Returns

vr_corr [dict] Field dictionary containing dealiased Doppler velocities. Dealiased array is stored under the 'data' key.

Other Parameters

gatefilter [GateFilter, optional.] A GateFilter instance which specifies which gates should be ignored when performing velocity dealiasing. A value of None will create this filter from the radar moments using any additional arguments by passing them to *moment_based_gate_filter()*. The default value assumes all gates are valid.

filt [int, optional] Flag controlling Bergen and Albers filter, 1 = yes, 0 = no.

rsl_badval [float, optional] Value which represents a bad value in RSL.

keep_original [bool, optional] True to keep original doppler velocity values when the dealiasing procedure fails, otherwise these gates will be masked. NaN values are still masked.

set_limits [bool, optional] True to set valid_min and valid_max elements in the returned dictionary. False will not set these dictionary elements.

vel_field [str, optional] Field in radar to use as the Doppler velocities during dealiasing. None will use the default field name from the Py-ART configuration file.

corr_vel_field [str, optional] Name to use for the dealiased Doppler velocity field metadata. None will use the default field name from the Py-ART configuration file.

last_vel_field [str, optional] Name to use for the dealiased Doppler velocity field metadata in last_radar. None will use the corr_vel_field name.

maxshear [float, optional] Maximum vertical shear which will be incorporated into the created volume from the sounding data. Parameter not used when no sounding data is provided.

sign [int, optional] Sign convention which the radial velocities in the volume created from the sounding data will will. This should match the convention used in the radar data. A value of 1 represents when positive values velocities are towards the radar, -1 represents when negative velocities are towards the radar.

compthresh [float, optional] Fraction of the Nyquist velocity to use as a threshold when performing continuity (initial) dealiasing. Velocities differences above this threshold will not be marked as gate from which to begin unfolding during spatial dealiasing.

compthresh2 [float, optional] The same as compthresh but the value used during the second pass of dealiasing. This second pass is only performed in both a sounding and last volume are provided.

thresh [float, optional] Fraction of the Nyquist velocity to use as a threshold when performing spatial dealiasing. Horizontally adjacent gates with velocities above this threshold will count against assigning the gate in question the velocity value being tested.

ckval [float, optional] When the absolute value of the velocities are below this value they will not be marked as gates from which to begin unfolding during spatial dealiasing.

stdthresh [float, optional] Fraction of the Nyquist velocity to use as a standard deviation threshold in the window dealiasing portion of the algorithm.

epsilon [float, optional] Difference used when comparing a value to missing value, changing this from the default is not recommended.

maxcount [int, optional] Maximum allowed number of fold allowed when unfolding velocities.

pass2 [int, optional] Controls weather unfolded gates should be removed (a value of 0) or retained for unfolding during the second pass (a value of 1) when both a sounding volume and last volume are provided.

rm [int, optional] Determines what should be done with gates that are left unfolded after the first pass of dealiasing. A value of 1 will remove these gates, a value of 0 sets these gates to their initial velocity. If both a sounding volume and last volume are provided this parameter is ignored.

proximity [int, optional] Number of gates and rays to include of either side of the current gate during window dealiasing. This value may be doubled in cases where a standard sized window does not capture a sufficient number of good valued gates.

mingood [int, optional] Number of good valued gates required within the window before the current gate will be unfolded.

ba_mincount [int, optional] Number of neighbors required during Bergen and Albers filter for a given gate to be included, must be between 1 and 8, 5 recommended.

ba_edgecount [int, optional] Same as ba_mincount but used at ray edges, must be between 1 and 5, 3 recommended.

debug [bool, optional] Set True to return RSL Volume objects for debugging: usuccess, radialVelVolume, lastVelVolume, unfoldedVolume, sondVolume

Notes

Due to limitations in the C code do not call with sounding arrays over 999 elements long.

References

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

pyart.correct.**dealias_region_based**(*radar*, *ref_vel_field=None*, *interval_splits=3*, *interval_limits=None*, *skip_between_rays=100*, *skip_along_ray=100*, *centered=True*, *nyquist_vel=None*, *check_nyquist_uniform=True*, *gatefilter=False*, *rays_wrap_around=None*, *keep_original=False*, *set_limits=True*, *vel_field=None*, *corr_vel_field=None*, ***kwargs*)

Dealias Doppler velocities using a region based algorithm.

Performs Doppler velocity dealiasing by finding regions of similar velocities and unfolding and merging pairs of regions until all regions are unfolded. Unfolding and merging regions is accomplished by modeling the problem as a dynamic network reduction.

Parameters

radar [Radar] Radar object containing Doppler velocities to dealias.

ref_vel_field [str or None, optional] Field in radar containing a reference velocity field used to anchor the unfolded velocities once the algorithm completes. Typically this field is created by simulating the radial velocities from wind data from an atmospheric sonding using *pyart.util.simulated_vel_from_profile()*.

interval_splits [int, optional] Number of segments to split the nyquist interval into when finding regions of similar velocity. More splits creates a larger number of initial regions which takes longer to process but may result in better dealiasing. The default value of 3 seems to be a good compromise between performance and artifact free dealiasing. This value is not used if the interval_limits parameter is not None.

interval_limits [array like or None, optional] Velocity limits used for finding regions of similar velocity. Should cover the entire nyquist interval. None, the default value, will split the Nyquist interval into interval_splits equal sized intervals.

skip_between_rays, skip_along_ray [int, optional] Maximum number of filtered gates to skip over when joining regions, gaps between region larger than this will not be connected. Parameters specify the maximum number of filtered gates between and along a ray. Set these parameters to 0 to disable unfolding across filtered gates.

centered [bool, optional] True to apply centering to each sweep after the dealiasing algorithm so that the average number of unfolding is near 0. False does not apply centering which may results in individual sweeps under or over folded by the nyquist interval.

nyquist_velocity [array like or float, optional] Nyquist velocity in unit identical to those stored in the radar's velocity field, either for each sweep or a single value which will be used for all sweeps. None will attempt to determine this value from the Radar object.

check_nyquist_uniform [bool, optional] True to check if the Nyquist velocities are uniform for all rays within a sweep, False will skip this check. This parameter is ignored when the nyquist_velocity parameter is not None.

gatefilter [GateFilter, None or False, optional.] A GateFilter instance which specified which gates should be ignored when performing de-aliasing. A value of None created this filter from the radar moments using any additional arguments by passing them to *moment_based_gate_filter()*. False, the default, disables filtering including all gates in the dealiasing.

rays_wrap_around [bool or None, optional] True when the rays at the beginning of the sweep and end of the sweep should be interpreted as connected when de-aliasing (PPI scans). False if they edges should not be interpreted as connected (other scan types). None will determine the correct value from the radar scan type.

keep_original [bool, optional] True to retain the original Doppler velocity values at gates where the dealiasing procedure fails or was not applied. False does not replacement and these gates will be masked in the corrected velocity field.

set_limits [bool, optional] True to set valid_min and valid_max elements in the returned dictionary. False will not set these dictionary elements.

vel_field [str, optional] Field in radar to use as the Doppler velocities during dealiasing. None will use the default field name from the Py-ART configuration file.

corr_vel_field [str, optional] Name to use for the dealiased Doppler velocity field metadata. None will use the default field name from the Py-ART configuration file.

Returns

corr_vel [dict] Field dictionary containing dealiased Doppler velocities. Dealiased array is stored under the 'data' key.

pyart.correct.**dealias_unwrap_phase**(*radar*, *unwrap_unit='sweep'*, *nyquist_vel=None*, *check_nyquist_uniform=True*, *gatefilter=False*, *rays_wrap_around=None*, *keep_original=False*, *set_limits=True*, *vel_field=None*, *corr_vel_field=None*, *skip_checks=False*, ***kwargs*)

Dealias Doppler velocities using multi-dimensional phase unwrapping.

Parameters

radar [Radar] Radar object containing Doppler velocities to dealias.

unwrap_unit [{'ray', 'sweep', 'volume'}, optional] Unit to unwrap independently. 'ray' will unwrap each ray individually, 'sweep' each sweep, and 'volume' will unwrap the entire volume in a single pass. 'sweep', the default, often gives superior results when the lower sweeps of the radar volume are contaminated by clutter. 'ray' does not use the gatefilter parameter and rays where gates ared masked will result in poor dealiasing for that ray.

nyquist_velocity [array like or float, optional] Nyquist velocity in unit identical to those stored in the radar's velocity field, either for each sweep or a single value which will be used for all sweeps. None will attempt to determine this value from the Radar object. The Nyquist velocity of the first sweep is used for all dealiasing unless the unwrap_unit is 'sweep' when the velocities of each sweep are used.

check_nyquist_uniform [bool, optional] True to check if the Nyquist velocities are uniform for all rays within a sweep, False will skip this check. This parameter is ignored when the nyquist_velocity parameter is not None.

gatefilter [GateFilter, None or False, optional.] A GateFilter instance which specified which gates should be ignored when performing de-aliasing. A value of None created this filter from the radar moments using any additional arguments by passing them to *moment_based_gate_filter()*. False, the default, disables filtering including all gates in the dealiasing.

rays_wrap_around [bool or None, optional] True when the rays at the beginning of the sweep and end of the sweep should be interpreted as connected when de-aliasing (PPI scans). False if they edges should not be interpreted as connected (other scan types). None will determine the correct value from the radar scan type.

keep_original [bool, optional] True to retain the original Doppler velocity values at gates where the dealiasing procedure fails or was not applied. False does not replacement and these gates will be masked in the corrected velocity field.

set_limits [bool, optional] True to set valid_min and valid_max elements in the returned dictionary. False will not set these dictionary elements.

vel_field [str, optional] Field in radar to use as the Doppler velocities during dealiasing. None will use the default field name from the Py-ART configuration file.

corr_vel_field [str, optional] Name to use for the dealiased Doppler velocity field metadata. None will use the default field name from the Py-ART configuration file.

skip_checks [bool] True to skip checks verifing that an appropiate unwrap_unit is selected, False retains these checked. Setting this parameter to True is not recommended and is only offered as an option for extreme cases.

Returns

corr_vel [dict] Field dictionary containing dealiased Doppler velocities. Dealiased array is stored under the 'data' key.

References

[1], [2]

pyart.correct.**despeckle_field**(*radar*, *field*, *label_dict=None*, *threshold=-100*, *size=10*, *gatefilter=None*, *delta=5.0*)

Despeckle a radar volume by identifying small objects in each scan and masking them out. User can define which field to investigate, as well as various thresholds to use on that field and any objects found within. Requires scipy to be installed, and returns a GateFilter object.

Parameters

radar [pyart.core.Radar object] Radar object to query.

field [str] Name of field to investigate for speckles.

Returns

gatefilter [pyart.filters.GateFilter object] Py-ART GateFilter object that includes the despeckling mask

Other Parameters

label_dict [dict or None, optional] Dictionary that is produced by find_objects. If None, find_objects will be called to produce it.

threshold [int or float, or 2-element tuple of ints or floats] Threshold values above (if single value) or between (if tuple) for objects to be identified. Default value assumes reflectivity.

size [int, optional] Number of contiguous gates in an object, below which it is a speckle.

gatefilter [None or pyart.filters.GateFilter object] Py-ART GateFilter object to which to add the despeckling mask. The GateFilter object will be permanently modified with the new filtering. If None, creates a new GateFilter.

delta [int or float, optional] Size of allowable gap near PPI edges, in deg, to consider it full 360. If gap is small, then PPI edges will be checked for matching objects.

pyart.correct.**det_sys_phase_ray**(*radar*, *ind_rmin=10*, *ind_rmax=500*, *min_rcons=11*, *zmin=20.0*, *zmax=40.0*, *phidp_field=None*, *refl_field=None*)

Public method Alternative determination of the system phase. Assumes that the valid gates of phidp are only precipitation. A system phase value is found for each ray.

Parameters

radar [Radar] Radar object for which to determine the system phase.

ind_rmin, ind_rmax [int] Min and max range index where to look for continuous precipitation

min_rcons [int] The minimum number of consecutive gates to consider it a rain cell.

zmin, zmax [float] The minimum and maximum reflectivity to consider the radar bin suitable precipitation

phidp_field [str] Field name within the radar object which represent the differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

refl_field [str] Field name within the radar object which represent the reflectivity. A value of None will use the default field name as defined in the Py-ART configuration file.

Returns

phidp0_dict [dict] Estimate of the system phase at each ray and metadata

first_gates_dict [dict] The first gate where PhiDP is valid and metadata

pyart.correct.**est_rhohv_rain**(*radar*, *ind_rmin=10*, *ind_rmax=500*, *zmin=20.0*, *zmax=40.0*, *thickness=700.0*, *doc=None*, *fzl=None*, *rhohv_field=None*, *temp_field=None*, *iso0_field=None*, *refl_field=None*, *temp_ref='temperature'*)

Estimates the quantiles of RhoHV in rain for each sweep

Parameters

radar [Radar] radar object

ind_rmin, ind_rmax [int] Min and max range index where to look for rain

zmin, zmax [float] The minimum and maximum reflectivity to consider the radar bin suitable rain

thickness [float] Assumed thickness of the melting layer

doc [float] Number of gates at the end of each ray to to remove from the calculation.

fzl [float] Freezing layer, gates above this point are not included in the correction.

temp_field, iso0_field, rhohv_field, refl_field [str] Field names within the radar object which represent the temperature, the height over the iso0, co-polar correlation and reflectivity fields. A value of None will use the default field name as defined in the Py-ART configuration file.

temp_ref [str] the field use as reference for temperature. Can be either temperature or height_over_iso0

Returns

rhohv_rain_dict [dict] The estimated RhoHV in rain for each sweep and metadata

pyart.correct.**est_zdr_precip**(*radar*, *ind_rmin=10*, *ind_rmax=500*, *zmin=20.0*, *zmax=22.0*, *rhohvmin=0.97*, *phidpmax=10.0*, *elmax=None*, *thickness=700.0*, *doc=None*, *fzl=None*, *zdr_field=None*, *rhohv_field=None*, *phidp_field=None*, *temp_field=None*, *iso0_field=None*, *refl_field=None*, *temp_ref='temperature'*)

Filters out all undesired data to be able to estimate ZDR bias, either in moderate rain or from vertically pointing scans

Parameters

radar [Radar] radar object

ind_rmin, ind_rmax [int] Min and max range index where to look for rain

zmin, zmax [float] The minimum and maximum reflectivity to consider the radar bin suitable rain

rhohvmin [float] Minimum RhoHV to consider the radar bin suitable rain

phidpmax [float] Maximum PhiDP to consider the radar bin suitable rain

elmax [float] Maximum elevation

thickness [float] Assumed thickness of the melting layer

doc [float] Number of gates at the end of each ray to to remove from the calculation.

fzl [float] Freezing layer, gates above this point are not included in the correction.

zdr_field, rhohv_field, refl_field, phidp_field, temp_field, iso0_field : str Field names within the radar object which represent the differential reflectivity, co-polar correlation, reflectivity, differential phase, temperature and height relative to the iso0 fields. A value of None will use the default field name as defined in the Py-ART configuration file.

temp_ref [str] the field use as reference for temperature. Can be either temperature, height_over_iso0, fixed_fzl or None

Returns

zdr_prec_dict [dict] The ZDR data complying with specifications and metadata

pyart.correct.**est_zdr_snow**(*radar, ind_rmin=10, ind_rmax=500, zmin=0.0, zmax=30.0, snrmin=10.0, snrmax=50.0, rhohvmin=0.97, kept_values=[2], phidpmax=10.0, kdpmax=None, tempmin=None, tempmax=None, elmax=None, zdr_field=None, rhohv_field=None, phidp_field=None, temp_field=None, snr_field=None, hydro_field=None, kdp_field=None, refl_field=None*)

Filters out all undesired data to be able to estimate ZDR bias in snow

Parameters

radar [Radar] radar object

ind_rmin, ind_rmax [int] Min and max range index where to look for snow

zmin, zmax [float] The minimum and maximum reflectivity to consider the radar bin suitable snow

snrmin, snrmax [float] The minimum and maximum SNR to consider the radar bin suitable snow

rhohvmin [float] Minimum RhoHV to consider the radar bin suitable snow

kept_values [list of int] The hydrometeor classification values to keep

phidpmax [float] Maximum PhiDP to consider the radar bin suitable snow

kdpmax [float or None] Maximum KDP. If not none this is the maximum KDP value to consider the radar bin suitable snow

tempmin, tempmax [float or None] If not None, the minimum and maximum temperature to consider the radar bin suitable snow

elmax [float] Maximum elevation

zdr_field, rhohv_field, refl_field, phidp_field, kdp_field, temp_field,

snr_field, hydro_field [str] Field names within the radar object which represent the differential reflectivity, co-polar correlation, reflectivity, differential phase, specific differnetial phase, signal to noise ratio, hydrometeor classification and temperature fields. A value of None will use the default field name as defined in the Py-ART configuration file.

Returns

zdr_snow_dict [dict] The ZDR data complying with specifications and metadata

pyart.correct.**find_objects**(*radar*, *field*, *threshold*, *sweeps=None*, *smooth=None*, *gatefilter=None*, *delta=5.0*)

Find objects (i.e., contiguous gates) in one or more sweeps that match thresholds. Filtering & smoothing are available prior to labeling objects. In addition, periodic boundaries are accounted for if they exist (e.g., 360-deg PPIs). Requires scipy to be installed.

Parameters

radar [pyart.core.Radar object] Radar object to query.

field [str] Name of field to investigate for objects.

threshold [int or float, or 2-element tuple of ints or floats] Threshold values above (if single value) or between (if tuple) for objects to be identified.

Returns

label_dict [dict] Dictionary that contains all the labeled objects. If this function is performed on the full Radar object, then the dict is ready to be added as a field.

Other Parameters

sweeps [int or array of ints or None, optional] Sweep numbers to examine. If None, all sweeps are examined.

smooth [int or None, optional] Number of gates included in a smoothing box filter along a ray. If None, no smoothing is done prior to labeling objects.

gatefilter [None or pyart.filters.GateFilter object, optional] Py-ART GateFilter object to apply before labeling objects. If None, no filtering will be performed. Note: Filtering always occurs before smoothing.

delta [int or float, optional] Size of allowable gap near PPI edges, in deg, to consider it full 360. If gap is small, then PPI edges will be checked for matching objects along the periodic boundary.

pyart.correct.**get_mask_fzl**(*radar*, *fzl=None*, *doc=None*, *min_temp=0.0*, *max_h_iso0=0.0*, *thickness=None*, *beamwidth=None*, *temp_field=None*, *iso0_field=None*, *temp_ref='temperature'*)

Constructs a mask to mask data placed thickness m below data at min_temp and beyond.

Parameters

radar [Radar] The radar object.

fzl [float, optional] Freezing layer, gates above this point are not included in the correction.

doc [float, optional] Number of gates at the end of each ray to to remove from the calculation.

min_temp [float, optional] Minimum temperature below which the data is mask in degrees.

max_h_iso0 [float, optional] Maximum height relative to the iso0 below which the data is mask in meters.

thickness [float, optional] Extent of the layer below the first gate where min_temp is reached that is going to be masked.

beamwidth [float, optional] The radar antenna 3 dB beamwidth.

temp_field: str, optional The temperature field. A value of None will use the default field name as defined in the Py-ART configuration file. It is going to be used only if available.

iso0_field: str, optional The field containing the height over the 0C isotherm. A value of None will use the default field name as defined in the Py-ART configuration file. It is going to be used only if available.

temp_ref [str, optional] The field use as reference for temperature. Can be either temperature, height_over_iso0 or fixed_fzl.

Returns

mask_fzl [2D array] The values that should be masked.

end_gate_arr [1D array] The index of the last valid gate in the ray.

pyart.correct.**get_sun_hits**(*radar*, *delev_max=2.0*, *dazim_max=2.0*, *elmin=1.0*, *rmin=50000.0*, *hmin=10000.0*, *nbins_min=20*, *attg=None*, *max_std_pwr=1.0*, *max_std_zdr=1.5*, *pwrh_field=None*, *pwrv_field=None*, *zdr_field=None*)

get data from suspected sun hits

Parameters

radar [Radar] radar object

delev_max, dazim_max [float] maximum difference in elevation and azimuth between sun position and antenna pointing

elmin [float] minimum radar elevation angle

rmin [float] minimum range from which we can look for noise [m]

hmin [float] minimum altitude from which we can look for noise [m]. The actual range min will be the minimum between rmin and the range bin higher than hmin.

nbins_min [int] Minimum number of bins with valid data to consider a ray as potentially sun hit

attg [float] gas attenuation coefficient (1-way)

max_std_pwr [float] Maximum standard deviation of the estimated sun power to consider the sun signal valid [dB]

max_std_zdr [float] Maximum standard deviation of the estimated sun ZDR to consider the sun signal valid [dB]

pwrh_field, pwrv_field, zdr_field [str] names of the signal power in dBm for the H and V polarizations and the differential reflectivity

Returns

sun_hits [dict] a dictionary containing information of the sun hits

new_radar [radar object] radar object containing sweeps that contain sun hits

pyart.correct.**moment_based_gate_filter**(*radar*, *ncp_field=None*, *rhv_field=None*, *refl_field=None*, *min_ncp=0.5*, *min_rhv=None*, *min_refl=-20.0*, *max_refl=100.0*)

Create a filter which removes undesired gates based on moments.

Creates a gate filter in which the following gates are excluded:

- Gates where the instrument is transitioning between sweeps.
- Gates where the reflectivity is outside the interval min_refl, max_refl.
- Gates where the normalized coherent power is below min_ncp.
- Gates where the cross correlation ratio is below min_rhi. Using the default parameter this filtering is disabled.
- Gates where any of the above three fields are masked or contain invalid values (NaNs or infs).
- If any of these three fields do not exist in the radar that fields filter criteria is not applied.

Parameters

radar [Radar] Radar object from which the gate filter will be built.

refl_field, ncp_field, rhv_field [str] Names of the radar fields which contain the reflectivity, normalized coherent power (signal quality index) and cross correlation ratio (RhoHV) from which the gate filter will be created using the above criteria. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

min_ncp, min_rhv [float] Minimum values for the normalized coherence power and cross correlation ratio. Gates in these fields below these limits as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use the filter. A value of None will disable filtering based upon the given field including removing masked or gates with an invalid value. To disable the thresholding but retain the masked and invalid filter set the parameter to a value below the lowest value in the field.

min_refl, max_refl [float] Minimum and maximum values for the reflectivity. Gates outside of this interval as well as gates which are masked or contain invalid values will be excluded and not used in calculation which use this filter. A value or None for one of these parameters will disable the minimum or maximum filtering but retain the other. A value of None for both of these values will disable all filtering based upon the reflectivity including removing masked or gates with an invalid value. To disable the interval filtering but retain the masked and invalid filter set the parameters to values above and below the lowest and greatest values in the reflectivity field.

Returns

gatefilter [*GateFilter*] A gate filter based upon the described criteria. This can be used as a gatefilter parameter to various functions in pyart.correct.

pyart.correct.**phase_proc_lp**(*radar*, *offset*, *debug=False*, *self_const=60000.0*, *low_z=10.0*, *high_z=53.0*, *min_phidp=0.01*, *min_ncp=0.5*, *min_rhv=0.8*, *fzl=4000.0*, *sys_phase=0.0*, *overide_sys_phase=False*, *nowrap=None*, *really_verbose=False*, *LP_solver='cylp'*, *refl_field=None*, *ncp_field=None*, *rhv_field=None*, *phidp_field=None*, *kdp_field=None*, *unf_field=None*, *window_len=35*, *proc=1*, *coef=0.914*)

Phase process using a LP method [1].

Parameters

radar [Radar] Input radar.

offset [float] Reflectivity offset in dBz.

debug [bool, optional] True to print debugging information.

self_const [float, optional] Self consistency factor.

low_z [float, optional] Low limit for reflectivity. Reflectivity below this value is set to this limit.

high_z [float, optional] High limit for reflectivity. Reflectivity above this value is set to this limit.

min_phidp [float, optional] Minimum Phi differential phase.

min_ncp [float, optional] Minimum normal coherent power.

min_rhv [float, optional] Minimum copolar coefficient.

fzl [float, optional] Maximum altitude.

sys_phase [float, optional] System phase in degrees.

overide_sys_phase [bool, optional] True to use *sys_phase* as the system phase. False will calculate a value automatically.

nowrap [int or None, optional] Gate number to begin phase unwrapping. None will unwrap all phases.

really_verbose [bool, optional] True to print LPX messaging. False to suppress.

LP_solver ['pyglpk' or 'cvxopt', 'cylp', or 'cylp_mp', optional] Module to use to solve LP problem. Default is 'cylp'.

refl_field, ncp_field, rhv_field, phidp_field, kdp_field [str, optional] Name of field in radar which contains the horizonal reflectivity, normal coherent power, copolar coefficient, differential phase shift, and differential phase. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

unf_field [str, optional] Name of field which will be added to the radar object which will contain the unfolded differential phase. Metadata for this field will be taken from the phidp_field. A value of None will use the default field name as defined in the Py-ART configuration file.

window_len [int, optional] Length of Sobel window applied to PhiDP field when prior to calculating KDP.

proc [int, optional] Number of worker processes, only used when *LP_solver* is 'cylp_mp'.

coef [float, optional] Exponent linking Z to KDP in self consistency. kdp=(10**(0.1z))*coef

Returns

reproc_phase [dict] Field dictionary containing processed differential phase shifts.

sob_kdp [dict] Field dictionary containing recalculated differential phases.

References

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

pyart.correct.**phase_proc_lp_gf**(*radar*, *gatefilter=None*, *debug=False*, *self_const=60000.0*, *low_z=10.0*, *high_z=53.0*, *min_phidp=0.01*, *fzl=4000.0*, *system_phase=None*, *nowrap=None*, *really_verbose=False*, *LP_solver='cylp'*, *refl_field=None*, *phidp_field=None*, *kdp_field=None*, *unf_field=None*, *window_len=35*, *proc=1*, *coef=0.914*, *ncpts=None*, *first_gate_sysp=None*, *offset=0.0*, *doc=0*)

Phase process using a LP method [1] using Py-ART's Gatefilter.

Parameters

radar [Radar] Input radar.

gatefilter [Gatefilter, optional] Py-ART gatefilter object indicating where processing should be carried out

debug [bool, optional] True to print debugging information.

self_const [float, optional] Self consistency factor.

low_z [float, optional] Low limit for reflectivity. Reflectivity below this value is set to this limit.

high_z [float, optional] High limit for reflectivity. Reflectivity above this value is set to this limit.

fzl [float, optional] Maximum altitude.

system_phase [float, optional] System phase in degrees.

nowrap [int or None, optional] Gate number to begin phase unwrapping. None will unwrap all phases.

really_verbose [bool, optional] True to print LPX messaging. False to suppress.

LP_solver ['pyglpk' or 'cvxopt', 'cylp', or 'cylp_mp', optional] Module to use to solve LP problem. Default is 'cylp'.

refl_field, ncp_field, rhv_field, phidp_field, kdp_field [str, optional] Name of field in radar which contains the horizonal reflectivity, normal coherent power, copolar coefficient, differential phase shift, and differential phase. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

unf_field [str, optional] Name of field which will be added to the radar object which will contain the unfolded differential phase. Metadata for this field will be taken from the phidp_field. A value of None will use the default field name as defined in the Py-ART configuration file.

window_len [int, optional] Length of Sobel window applied to PhiDP field when prior to calculating KDP.

proc [int, optional] Number of worker processes, only used when *LP_solver* is 'cylp_mp'.

coef [float, optional] Exponent linking Z to KDP in self consistency. kdp=(10**(0.1z))*coef

ncpts [int, optional] Minimum number of points in a ray. Regions within a ray smaller than this or beginning before this gate number are excluded from unfolding.

offset [float, optional] Reflectivity offset to add in dBz.

doc [int, optional] Number of gates to "doc" off the end of a ray.

Returns

reproc_phase [dict] Field dictionary containing processed differential phase shifts.

sob_kdp [dict] Field dictionary containing recalculated differential phases.

References

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

pyart.correct.**ptoa_to_sf**(*ptoa*, *pulse_width*, *wavelen*, *antenna_gain*, *coeff_band=1.2*)

Converts the sun power at the top of the atmosphere (in dBm) into solar flux.

Parameters

ptoa [float] sun power at the top of the amosphere. It already takes into account the correction for antenna polarization

pulse_width [float] pulse width [s]

wavelen [float] radar wavelength [m]

antenna_gain [float] the antenna gain [dB]

coeff_band [float] multiplicative coefficient applied to the inverse of the pulse width to get the effective bandwidth

Returns

s0 [float] solar flux [10e-22 W/(m2 Hz)]

References

Altube P., J. Bech, O. Argemi, T. Rigo, 2015: Quality Control of Antenna Alignment and Receiver Calibration Using the Sun: Adaptation to Midrange Weather Radar Observations at Low Elevation Angles

pyart.correct.**scanning_losses**(*angle_step*, *beamwidth*)

Given the antenna beam width and the integration angle, compute the losses due to the fact that the sun is not a point target and the antenna is scanning

Parameters

angle_step [float] integration angle [deg]

beamwidth [float] 3 dB-beamwidth [deg]

Returns

la [float] The losses due to the scanning of the antenna [dB positive]

References

Altube P., J. Bech, O. Argemi, T. Rigo, 2015: Quality Control of Antenna Alignment and Receiver Calibration Using the Sun: Adaptation to Midrange Weather Radar Observations at Low Elevation Angles

pyart.correct.**selfconsistency_bias**(*radar*, *zdr_kdpzh_dict*, *min_rhohv=0.92*, *filter_rain=True*, *max_phidp=20.0*, *smooth_wind_len=5*, *doc=None*, *fzl=None*, *thickness=700.0*, *min_rcons=20*, *dphidp_min=2*, *dphidp_max=16*, *parametrization='None'*, *refl_field=None*, *phidp_field=None*, *zdr_field=None*, *temp_field=None*, *iso0_field=None*, *hydro_field=None*, *rhohv_field=None*, *temp_ref='temperature'*, *check_wet_radome=True*, *wet_radome_refl=25.0*, *wet_radome_len_min=4*, *wet_radome_len_max=8*, *wet_radome_ngates_min=180*, *valid_gates_only=False*, *keep_points=False*, *kdp_wind_len=12*)

Estimates reflectivity bias at each ray using the self-consistency algorithm by Gourley

Parameters

radar [Radar] radar object

zdr_kdpzh_dict [dict] dictionary containing a look up table relating ZDR with KDP/Zh for different elevations

min_rhohv [float] minimum RhoHV value to consider the data valid

filter_rain [bool] If True the hydrometeor classification is going to be used to filter out all gates that are not rain

max_phidp [float] maximum PhiDP value to consider the data valid

smooth_wind_len [int] length of the smoothing window

doc [float] Number of gates at the end of each ray to to remove from the calculation.

fzl [float] Freezing layer, gates above this point are not included in the correction.

thickness [float] assumed melting layer thickness [m]

min_rcons [int] minimum number of consecutive gates to consider a valid segment of PhiDP

dphidp_min [float] minimum differential phase shift in a segment

dphidp_max [float] maximum differential phase shift in a segment

parametrization [str] The type of parametrization for the self-consistency curves. Can be 'None', 'Gourley', 'Wolfensberger', 'Louf', 'Gorgucci' or 'Vaccarono'. 'None' will use tables contained in zdr_kdpzh_dict.

refl_field, phidp_field, zdr_field [str] Field names within the radar object which represent the reflectivity, differential phase and differential reflectivity fields. A value of None will use the default field name as defined in the Py-ART configuration file.

temp_field, iso0_field, hydro_field, rhohv_field [str] Field names within the radar object which represent the temperature, the height relative to the iso0, the hydrometeor classification and the co-polar correlation fields. A value of None will use the default field name as defined in the Py-ART configuration file. They are going to be used only if available.

kdpsim_field, phidpsim_field [str] Field names which represent the estimated specific differential phase and differential phase. A value of None will use the default field name as defined in the Py-ART configuration file.

temp_ref [str] the field use as reference for temperature. Can be either temperature, height_over_iso0 or fixed_fzl

check_wet_radome [Bool] if True the average reflectivity of the closest gates to the radar is going to be check to find out whether there is rain over the radome. If there is rain no bias will be computed

wet_radome_refl [Float] Average reflectivity of the gates close to the radar to consider the radome as wet

wet_radome_len_min, wet_radome_len_max [int] Mim and max gate indices of the disk around the radome used to decide whether the radome is wet

wet_radome_ngates_min [int] Minimum number of valid gates to consider that the radome is wet

valid_gates_only [Bool] If True the reflectivity bias obtained for each valid ray is going to be assigned only to gates of the segment used. That will give more weight to longer segments when computing the total bias.

keep_points [Bool] If True the ZDR, ZH and KDP of the gates used in the self- consistency algorithm are going to be stored for further analysis

kdp_wind_len [int] The length of the window used to compute KDP with the single window least square method

Returns

refl_bias_dict [dict] the bias at each ray field and metadata

pyart.correct.**selfconsistency_kdp_phidp**(*radar*, *zdr_kdpzh_dict*, *min_rhohv=0.92*, *filter_rain=True*, *max_phidp=20.0*, *smooth_wind_len=5*, *doc=None*, *fzl=None*, *thickness=700.0*, *parametrization='None'*, *refl_field=None*, *phidp_field=None*, *zdr_field=None*, *temp_field=None*, *iso0_field=None*, *hydro_field=None*, *rhohv_field=None*, *kdpsim_field=None*, *phidpsim_field=None*, *temp_ref='temperature'*)

Estimates KDP and PhiDP in rain from Zh and ZDR using a selfconsistency relation between ZDR, Zh and KDP. Private method

Parameters

radar [Radar] radar object

zdr_kdpzh_dict [dict] dictionary containing a look up table relating ZDR with KDP/Zh for different elevations

min_rhohv [float] minimum RhoHV value to consider the data valid

filter_rain [bool] If True the hydrometeor classification is going to be used to filter out all gates that are not rain

max_phidp [float] maximum PhiDP value to consider the data valid

smooth_wind_len [int] length of the smoothing window

doc [float] Number of gates at the end of each ray to to remove from the calculation.

fzl [float] Freezing layer, gates above this point are not included in the correction.

thickness [float] assumed melting layer thickness [m]

parametrization [str] The type of parametrization for the self-consistency curves. Can be 'None', 'Gourley', 'Wolfensberger', 'Louf', 'Gorgucci' or 'Vaccarono'. 'None' will use tables contained in zdr_kdpzh_dict.

refl_field, phidp_field, zdr_field [str] Field names within the radar object which represent the reflectivity, differential phase and differential reflectivity fields. A value of None will use the default field name as defined in the Py-ART configuration file.

temp_field, iso0_field, hydro_field, rhohv_field [str] Field names within the radar object which represent the temperature, the height relative to the iso0, the hydrometeor classification and the co-polar correlation fields. A value of None will use the default field name as defined in the Py-ART configuration file. They are going to be used only if available.

kdpsim_field, phidpsim_field [str] Field names which represent the estimated specific differential phase and differential phase. A value of None will use the default field name as defined in the Py-ART configuration file.

temp_ref [str] the field use as reference for temperature. Can be either temperature, height_over_iso0 or fixed_fzl

Returns

kdp_sim_dict, phidp_sim_dict [dict] the KDP and PhiDP estimated fields and metadata

pyart.correct.**smooth_masked**(*raw_data*, *wind_len=11*, *min_valid=6*, *wind_type='median'*)
smoothes the data using a rolling window. data with less than n valid points is masked.

Parameters

raw_data [float masked array] The data to smooth.

window_len [float] Length of the moving window

min_valid [float] Minimum number of valid points for the smoothing to be valid

wind_type [str] type of window. Can be median or mean

Returns

data_smooth [float masked array] smoothed data

pyart.correct.**smooth_phidp_double_window**(*radar*, *ind_rmin=10*, *ind_rmax=500*, *min_rcons=11*, *zmin=20.0*, *zmax=40*, *swind_len=11*, *smin_valid=6*, *lwind_len=31*, *lmin_valid=16*, *zthr=40.0*, *psidp_field=None*, *refl_field=None*, *phidp_field=None*)

correction of the system offset and smoothing using two window

Parameters

radar [Radar] Radar object for which to determine the system phase.

ind_rmin, ind_rmax [int] Min and max range index where to look for continuous precipitation

min_rcons [int] The minimum number of consecutive gates to consider it a rain cell.

zmin, zmax [float] Minimum and maximum reflectivity to consider it a rain cell

swind_len [int] Length of the short moving window used to smooth

smin_valid [int] Minimum number of valid bins to consider the short window smooth data valid

lwind_len [int] Length of the long moving window used to smooth

lmin_valid [int] Minimum number of valid bins to consider the long window smooth data valid

zthr [float] reflectivity value above which the short window is used

psidp_field [str] Field name within the radar object which represent the differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

refl_field [str] Field name within the radar object which represent the reflectivity. A value of None will use the default field name as defined in the Py-ART configuration file.

phidp_field [str] Field name within the radar object which represent the corrected differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

Returns

phidp_dict [dict] The corrected phidp field

pyart.correct.**smooth_phidp_single_window**(*radar*, *ind_rmin=10*, *ind_rmax=500*, *min_rcons=11*, *zmin=20.0*, *zmax=40*, *wind_len=11*, *min_valid=6*, *psidp_field=None*, *refl_field=None*, *phidp_field=None*)

correction of the system offset and smoothing using one window

Parameters

radar [Radar] Radar object for which to determine the system phase.

ind_rmin, ind_rmax [int] Min and max range index where to look for continuous precipitation

min_rcons [int] The minimum number of consecutive gates to consider it a rain cell.

zmin, zmax [float] Minimum and maximum reflectivity to consider it a rain cell

wind_len [int] Length of the moving window used to smooth

min_valid [int] Minimum number of valid bins to consider the smooth data valid

psidp_field [str] Field name within the radar object which represent the differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

refl_field [str] Field name within the radar object which represent the reflectivity. A value of None will use the default field name as defined in the Py-ART configuration file.

phidp_field [str] Field name within the radar object which represent the corrected differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

Returns

phidp_dict [dict] The corrected phidp field

pyart.correct.**solar_flux_lookup**(*solar_flux*, *wavelen*)

Given the observed solar flux at 10.7 cm wavelength, returns the solar flux at the given radar wavelength

Parameters

solar_flux [float array] the solar fluxes measured at 10.7 cm wavelength [10e-22 W/(m2 Hz)]

wavelen [float] radar wavelength [m]

Returns

s0 [float] the radar flux at the radar wavelength [10e-22 W/(m2 Hz)]

References

Altube P., J. Bech, O. Argemi, T. Rigo, 2015: Quality Control of Antenna Alignment and Receiver Calibration Using the Sun: Adaptation to Midrange Weather Radar Observations at Low Elevation Angles

pyart.correct.**sun_power**(*solar_flux*, *pulse_width*, *wavelen*, *antenna_gain*, *angle_step*, *beamwidth*, *coeff_band=1.2*)

computes the theoretical sun power detected at the antenna [dBm] as it would be without atmospheric attenuation (sun power at top of the atmosphere) for a given solar flux and radar characteristics

Parameters

solar_flux [float array] the solar fluxes measured at 10.7 cm wavelength [10e-22 W/(m2 Hz)]

pulse_width [float] pulse width [s]

wavelen [float] radar wavelength [m]

antenna_gain [float] the antenna gain [dB]

angle_step [float] integration angle [deg]

beamwidth [float] 3 dB-beamwidth [deg]

coeff_band [float] multiplicative coefficient applied to the inverse of the pulse width to get the effective bandwidth

Returns

pwr_det [float array] the detected power

References

Altube P., J. Bech, O. Argemi, T. Rigo, 2015: Quality Control of Antenna Alignment and Receiver Calibration Using the Sun: Adaptation to Midrange Weather Radar Observations at Low Elevation Angles

pyart.correct.**sun_retrieval**(*az_rad*, *az_sun*, *el_rad*, *el_sun*, *sun_hit*, *sun_hit_std*, *az_width_co=None*, *el_width_co=None*, *az_width_cross=None*, *el_width_cross=None*, *is_zdr=False*)

Estimates sun parameters from sun hits

Parameters

az_rad, az_sun, el_rad, el_sun [float array] azimuth and elevation values of the sun and the radar

sun_hit [float array] sun hit value. Either power in dBm or ZDR in dB

sun_hit_std [float array] standard deviation of the sun hit value in dB

az_width_co, el_width_co, az_width_cross, el_width_cross [float] azimuth and elevation antenna width for each channel

is_zdr [boolean] boolean to signal that is ZDR data

Returns

val, val_std [float] retrieved value and its standard deviation

az_bias, el_bias [float] retrieved azimuth and elevation antenna bias respect to the sun position

az_width, el_width [float] retrieved azimuth and elevation antenna widths

nhits [int] number of sun hits used in the retrieval

par [float array] and array with the 5 parameters of the Gaussian fit

---

CHAPTER

SEVEN

RADAR RETRIEVALS (PYART.RETRIEVE)

Radar retrievals.

7.1 Radar retrievals

kdp_maesaka(radar[, gatefilter, method, ...])	Compute the specific differential phase (KDP) from corrected (e.g., unfolded) total differential phase data based on the variational method outlined in Maesaka et al.
kdp_schneebeli(radar[, gatefilter, ...])	Estimates Kdp with the Kalman filter method by Schneebeli and al.
kdp_vulpiani(radar[, gatefilter, ...])	Estimates Kdp with the Vulpiani method for a 2D array of psidp measurements with the first dimension being the distance from radar and the second dimension being the angles (azimuths for PPI, elev for RHI).The input psidp is assumed to be pre-filtered (for ex.
kdp_leastsquare_single_window(radar[, ...])	Compute the specific differential phase (KDP) from differential phase data using a piecewise least square method.
kdp_leastsquare_double_window(radar[, ...])	Compute the specific differential phase (KDP) from differential phase data using a piecewise least square method.
calculate_snr_from_reflectivity(radar[, ...])	Calculate the signal to noise ratio, in dB, from the reflectivity field.
calculate_velocity_texture(radar[, ...])	Derive the texture of the velocity field.
compute_ccor(radar[, filt_field, ...])	Computes the clutter correction ratio (CCOR), i.e.
compute_snr(radar[, refl_field, ...])	Computes SNR from a reflectivity field and the noise in dBZ.
compute_l(radar[, rhohv_field, l_field])	Computes Rhohv in logarithmic scale according to L=-log10(1-RhoHV).
compute_cdr(radar[, rhohv_field, zdr_field, ...])	Computes the Circular Depolarization Ratio.
compute_noisedBZ(nrays, noisedBZ_val, ...[, ...])	Computes noise in dBZ from reference noise value.
compute_radial_noise(radar[, ind_rmin, ...])	Computes radial noise in dBm from signal power
compute_signal_power(radar[, lmf, attg, ...])	Computes received signal power OUTSIDE THE RADOME in dBm from a reflectivity field.
compute_rcs(radar[, kw2, pulse_width, ...])	Computes the radar cross-section (assuming a point target) from radar reflectivity.

Continued on next page

Table 1 – continued from previous page

compute_rcs_from_pr(radar[, lmf, attg, ...])	Computes the radar cross-section (assuming a point target) from radar reflectivity by first computing the received power and then the RCS from it.
compute_vol_refl(radar[, kw, freq, ...])	Computes the volumetric reflectivity from the effective reflectivity factor
compute_bird_density(radar[, sigma_bird, ...])	Computes the bird density from the volumetric reflectivity
fetch_radar_time_profile(sonde_dset, radar)	Extract the correct profile from a interpolated sonde.
map_profile_to_gates(profile, heights, radar)	Given a profile of a variable map it to the gates of radar assuming 4/3Re.
steiner_conv_strat(grid[, dx, dy, intense, ...])	Partition reflectivity into convective-stratiform using the Steiner et al.
hydroclass_semisupervised(radar[, ...])	Classifies precipitation echoes following the approach by Besic et al (2016).
get_freq_band(freq)	Returns the frequency band name (S, C, X, ...).
texture_of_complex_phase(radar[, ...])	Calculate the texture of the differential phase field.
grid_displacement_pc(grid1, grid2, field, level)	Calculate the grid displacement using phase correlation.
grid_shift(grid, advection[, trim_edges, ...])	Shift a grid by a certain number of pixels.
est_rain_rate_zpoly(radar[, refl_field, ...])	Estimates rainfall rate from reflectivity using a polynomial Z-R relation developed at McGill University.
est_rain_rate_z(radar[, alpha, beta, ...])	Estimates rainfall rate from reflectivity using a power law.
est_rain_rate_kdp(radar[, alpha, beta, ...])	Estimates rainfall rate from kdp using alpha power law.
est_rain_rate_a(radar[, alpha, beta, ...])	Estimates rainfall rate from specific attenuation using alpha power law.
est_rain_rate_zkdp(radar[, alphaz, betaz, ...])	Estimates rainfall rate from a blending of power law r-kdp and r-z relations.
est_rain_rate_za(radar[, alphaz, betaz, ...])	Estimates rainfall rate from a blending of power law r-alpha and r-z relations.
est_rain_rate_hydro(radar[, alphazr, ...])	Estimates rainfall rate using different relations between R and the polarimetric variables depending on the hydrometeor type.
est_wind_vel(radar[, vert_proj, vel_field, ...])	Estimates wind velocity.
est_vertical_windshear(radar[, az_tol, ...])	Estimates wind shear.
atmospheric_gas_att(freq, elev, rng)	Computes the one-way atmospheric gas attenuation [dB] according to the empirical formula in Doviak and Zrnic (1993) pp 44.
get_coeff_attg(freq)	get the 1-way gas attenuation for a particular frequency
est_wind_profile(radar[, npoints_min, ...])	Estimates the vertical wind profile using VAD techniques
detect_ml(radar[, gatefilter, fill_value, ...])	Detects the melting layer (ML) using the reflectivity and copolar correlation coefficient.
melting_layer_giangrande(radar[, nVol, ...])	Detects the melting layer following the approach by Giangrande et al (2008)
melting_layer_hydroclass(radar[, ...])	Using the results of the hydrometeor classification by Besic et al.
_get_res_vol_sides(radar)	Computes the height of the lower left and upper right points of the range resolution volume.
velocity_azimuth_display(radar[, vel_field, ...])	Velocity azimuth display.
quasi_vertical_profile(radar[, ...])	Quasi Vertical Profile.

Continued on next page

Table 1 – continued from previous page

compute_qvp(radar, field_names[, ref_time, ...])	Computes quasi vertical profiles.
compute_rqvp(radar, field_names[, ref_time, ...])	Computes range-defined quasi vertical profiles.
compute_evp(radar, field_names, lon, lat[, ...])	Computes enhanced vertical profiles.
compute_svp(radar, field_names, lon, lat, angle)	Computes slanted vertical profiles.
compute_vp(radar, field_names, lon, lat[, ...])	Computes vertical profiles.
compute_ts_along_coord(radar, field_name[, ...])	Computes time series along a particular antenna coordinate, i.e.
compute_iq(spectra, fields_in_list, ...[, ...])	Computes the IQ data from the spectra through an inverse Fourier transform
compute_spectral_power(spectra[, units, ...])	Computes the spectral power from the complex spectra in ADU.
compute_spectral_noise(spectra[, units, ...])	Computes the spectral noise power from the complex spectra in ADU.
compute_spectral_phase(spectra[, signal_field])	Computes the spectral phase from the complex spectra in ADU
compute_spectral_reflectivity(spectra[, ...])	Computes the spectral reflectivity from the complex spectra in ADU or from the signal power in ADU.
compute_spectral_differential_reflectivity(spectra)	Computes the spectral differential reflectivity from the complex spectras or the power in ADU
compute_spectral_differential_phase(spectra)	Computes the spectral differential reflectivity from the complex spectras in ADU or sRhoHV
compute_spectral_rhohv(spectra[, ...])	Computes the spectral RhoHV from the complex spectras in ADU
compute_pol_variables(spectra, fields_list)	Computes the polarimetric variables from the complex spectra in ADU or the spectral powers and spectral RhoHV
compute_noise_power(spectra[, units, navg, ...])	Computes the noise power from the complex spectra in ADU.
compute_reflectivity(spectra[, sdBZ_field])	Computes the reflectivity from the spectral reflectivity
compute_differential_reflectivity(spectra[, ...])	Computes the differential reflectivity from the horizontal and vertical spectral reflectivity
compute_differential_phase(spectra[, ...])	Computes the differential phase from the spectral differential phase and the spectral reflectivity
compute_rhohv(spectra[, use_rhohv, ...])	Computes RhoHV from the horizontal and vertical spectral reflectivity or from sRhoHV and the spectral powers
compute_Doppler_velocity(spectra[, sdBZ_field])	Computes the Doppler velocity from the spectral reflectivity
compute_Doppler_width(spectra[, sdBZ_field])	Computes the Doppler width from the spectral reflectivity
compute_reflectivity_iq(radar[, ...])	Computes the reflectivity from the IQ signal data
compute_st1_iq(radar[, signal_field])	Computes the statistical test one lag fluctuation from the horizontal or vertical channel IQ data
compute_st2_iq(radar[, signal_field])	Computes the statistical test two lag fluctuation from the horizontal or vertical channel IQ data
compute_wbn_iq(radar[, signal_field])	Computes the wide band noise from the horizontal or vertical channel IQ data
compute_differential_reflectivity_iq(radar)	Computes the differential reflectivity from the horizontal and vertical IQ data
compute_mean_phase_iq(radar[, signal_field])	Computes the differential phase from the horizontal or vertical channel IQ data

Continued on next page

Table 1 – continued from previous page

compute_differential_phase_iq(radar[, ...])	Computes the differential phase from the horizontal and vertical channels IQ data
compute_rhohv_iq(radar[, subtract_noise, ...])	Computes RhoHV from the horizontal and vertical channels IQ data
compute_Doppler_velocity_iq(radar[, ...])	Computes the Doppler velocity from the IQ data
compute_Doppler_width_iq(radar[, ...])	Computes the Doppler width from the IQ data
compute_pol_variables_iq(radar, fields_list)	Computes the polarimetric variables from the IQ signals in ADU
compute_spectra(radar, fields_in_list, ...)	Computes the spectra from IQ data through a Fourier transform

pyart.retrieve.**atmospheric_gas_att**(*freq*, *elev*, *rng*)

Computes the one-way atmospheric gas attenuation [dB] according to the empirical formula in Doviak and Zrnic (1993) pp 44. This formula is valid for elev < 10 deg and rng < 200 km so values above these will be saturated to 10 deg and 200 km respectively

Parameters

freq [float] radar frequency [Hz]

elev [float or array of floats] elevation angle [deg]

rng [float or array of floats. If array must have the same size as elev] range [km]

Returns

latm [float or array of floats] 1-way gas attenuation [dB]

pyart.retrieve.**calculate_snr_from_reflectivity**(*radar*, *refl_field=None*, *snr_field=None*, *toa=25000.0*)

Calculate the signal to noise ratio, in dB, from the reflectivity field.

Parameters

radar [Radar] Radar object from which to retrieve reflectivity field.

refl_field [str, optional] Name of field in radar which contains the reflectivity. None will use the default field name in the Py-ART configuration file.

snr_field [str, optional] Name to use for snr metadata. None will use the default field name in the Py-ART configuration file.

toa [float, optional] Height above which to take noise floor measurements, in meters.

Returns

snr [field dictionary] Field dictionary containing the signal to noise ratio.

pyart.retrieve.**calculate_velocity_texture**(*radar*, *vel_field=None*, *wind_size=4*, *nyq=None*, *check_nyq_uniform=True*)

Derive the texture of the velocity field.

Parameters

radar: Radar Radar object from which velocity texture field will be made.

vel_field [str, optional] Name of the velocity field. A value of None will force Py-ART to automatically determine the name of the velocity field.

wind_size [int, optional] The size of the window to calculate texture from. The window is defined to be a square of size wind_size by wind_size.

nyq [float, optional] The nyquist velocity of the radar. A value of None will force Py-ART to try and determine this automatically.

check_nyquist_uniform [bool, optional] True to check if the Nyquist velocities are uniform for all rays within a sweep, False will skip this check. This parameter is ignored when the nyq parameter is not None.

Returns

vel_dict: dict A dictionary containing the field entries for the radial velocity texture.

pyart.retrieve.**compute_Doppler_velocity**(*spectra*, *sdBZ_field=None*)
Computes the Doppler velocity from the spectral reflectivity

Parameters

spectra [Radar spectra object] Object containing the required fields

sdBZ_field [str] Name of the field that contains the spectral reflectivity. None will use the default field name in the Py-ART configuration file.

Returns

vel_dict [field dictionary] Field dictionary containing the Doppler velocity

pyart.retrieve.**compute_Doppler_velocity_iq**(*radar*, *signal_field=None*, *direction='negative_away'*)
Computes the Doppler velocity from the IQ data

Parameters

radar [IQ radar object] Object containing the required fields

signal_field [str] Name of the field in the radar which contains the signal. None will use the default field name in the Py-ART configuration file.

direction [str] The convention used in the Doppler mean field. Can be negative_away or negative_towards

Returns

vel_dict [field dictionary] Field dictionary containing the Doppler velocity

pyart.retrieve.**compute_Doppler_width**(*spectra*, *sdBZ_field=None*)
Computes the Doppler width from the spectral reflectivity

Parameters

spectra [Radar spectra object] Object containing the required fields

sdBZ_field [str] Name of the field that contains the spectral reflectivity. None will use the default field name in the Py-ART configuration file.

Returns

width_dict [field dictionary] Field dictionary containing the Doppler spectrum width

pyart.retrieve.**compute_Doppler_width_iq**(*radar*, *subtract_noise=True*, *signal_field=None*, *noise_field=None*, *lag=1*)
Computes the Doppler width from the IQ data

Parameters

radar [Radar radar object] Object containing the required fields

subtract_noise [Bool] If True noise will be subtracted from the signals

lag [int] Time lag used in the denominator of the computation

signal_field, noise_field [str] Name of the field in the radar which contains the signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

width_dict [field dictionary] Field dictionary containing the Doppler spectrum width

pyart.retrieve.**compute_bird_density**(*radar*, *sigma_bird=11*, *vol_refl_field=None*, *bird_density_field=None*)

Computes the bird density from the volumetric reflectivity

Parameters

radar [Radar] radar object

sigma_bird [float] Estimated bird radar cross-section

vol_refl_field [str] name of the volumetric reflectivity used for the calculations

bird_density_field [str] name of the bird density field

Returns

bird_density_dict [dict] bird density data and metadata [birds/km^3]

pyart.retrieve.**compute_ccor**(*radar*, *filt_field=None*, *unfilt_field=None*, *ccor_field=None*)

Computes the clutter correction ratio (CCOR), i.e. the ratio between the signal without Doppler filtering and the signal with Doppler filtering

Parameters

radar [Radar] Radar object

filt_field, unfilt_field [str] Name of Doppler filtered and unfiltered fields

ccor_field [str] Name of the CCOR field

Returns

ccor_dict [field dictionary] Field dictionary containing the CCOR

pyart.retrieve.**compute_cdr**(*radar*, *rhohv_field=None*, *zdr_field=None*, *cdr_field=None*)

Computes the Circular Depolarization Ratio.

Parameters

radar [Radar] Radar object.

rhohv_field [str, optional] Name of the RhoHV field.

zdr_field [str, optional] Name of the ZDR field.

cdr_field [str, optional] Name of the CDR field.

Returns

cdr [dict] CDR field.

pyart.retrieve.**compute_differential_phase**(*spectra*, *sdBZ_field=None*, *sPhiDP_field=None*)

Computes the differential phase from the spectral differential phase and the spectral reflectivity

Parameters

spectra [Radar spectra object] Object containing the required fields

sdBZ_field, sPhiDP_field [str] Name of the fields that contain the spectral reflectivity and the spectral differential phase. None will use the default field name in the Py-ART configuration file.

Returns

PhiDP_dict [field dictionary] Field dictionary containing the differential phase

pyart.retrieve.**compute_differential_phase_iq**(*radar*, *phase_offset=0.0*, *signal_h_field=None*, *signal_v_field=None*)

Computes the differential phase from the horizontal and vertical channels IQ data

Parameters

radar [IQ radar object] Object containing the required fields

phase_offset [float] system phase offset to add

signal_h_field, signal_v_field [str] Name of the fields that contain the H and V IQ data. None will use the default field name in the Py-ART configuration file.

Returns

phidp_dict [field dictionary] Field dictionary containing the differential phase

pyart.retrieve.**compute_differential_reflectivity**(*spectra*, *sdBZ_field=None*, *sdBZv_field=None*)

Computes the differential reflectivity from the horizontal and vertical spectral reflectivity

Parameters

spectra [Radar spectra object] Object containing the required fields

sdBZ_field, sdBZv_field [str] Name of the fields that contain the spectral reflectivity. None will use the default field name in the Py-ART configuration file.

Returns

ZDR_dict [field dictionary] Field dictionary containing the differential reflectivity

pyart.retrieve.**compute_differential_reflectivity_iq**(*radar*, *subtract_noise=False*, *lag=0*, *signal_h_field=None*, *signal_v_field=None*, *noise_h_field=None*, *noise_v_field=None*)

Computes the differential reflectivity from the horizontal and vertical IQ data

Parameters

radar [IQ radar object] Object containing the required fields

subtract_noise [Bool] If true the noise is subtracted from the power

lag [int] Time lag used to compute the differential reflectivity

signal_h_field, signal_v_field, noise_h_field, noise_v_field [str] Name of the signal and noise fields. None will use the default field name in the Py-ART configuration file.

Returns

zdr_dict [field dictionary] Field dictionary containing the differential reflectivity

pyart.retrieve.**compute_evp**(*radar*, *field_names*, *lon*, *lat*, *ref_time=None*, *latlon_tol=0.0005*, *delta_rng=15000.0*, *delta_azi=10*, *hmax=10000.0*, *hres=250.0*, *avg_type='mean'*, *nvalid_min=1*, *interp_kind='none'*, *qvp=None*)

Computes enhanced vertical profiles.

Parameters

radar [Radar] Radar object used.

field_names [list of str] list of field names to add to the QVP

lat, lon [float] latitude and longitude of the point of interest [deg]

ref_time [datetime object] reference time for current radar volume

latlon_tol [float] tolerance in latitude and longitude in deg.

delta_rng, delta_azi [float] maximum range distance [m] and azimuth distance [degree] from the central point of the evp containing data to average.

hmax [float] The maximum height to plot [m].

hres [float] The height resolution [m].

avg_type [str] The type of averaging to perform. Can be either "mean" or "median"

nvalid_min [int] Minimum number of valid points to accept average.

interp_kind [str] type of interpolation when projecting to vertical grid: 'none', or 'nearest', etc. 'none' will select from all data points within the regular grid height bin the closest to the center of the bin. 'nearest' will select the closest data point to the center of the height bin regardless if it is within the height bin or not. Data points can be masked values If another type of interpolation is selected masked values will be eliminated from the data points before the interpolation

qvp [QVP object or None] If it is not None this is the QVP object where to store the data from the current time step. Otherwise a new QVP object will be created

Returns

qvp [qvp object] The computed enhanced vertical profile

pyart.retrieve.**compute_iq**(*spectra*, *fields_in_list*, *fields_out_list*, *window=None*)
Computes the IQ data from the spectra through an inverse Fourier transform

Parameters

spectra [Spectra radar object] Object containing the spectra

fields_in_list [list of str] list of input spectra fields names

fields_out_list [list of str] list with the output IQ fields names obtained from the input fields

window [string, tupple or None] Parameters of the window used to obtain the spectra. The parameters are the ones corresponding to function scipy.signal.windows.get_window. If it is not None the inverse will be used to multiply the IQ data obtained by the IFFT

Returns

radar [IQ radar object] radar object containing the IQ fields

pyart.retrieve.**compute_l**(*radar*, *rhohv_field=None*, *l_field=None*)
Computes Rhohv in logarithmic scale according to L=-log10(1-RhoHV).

Parameters

radar [Radar] Radar object.

rhohv_field [str, optional] Name of the RhoHV field to use.

l_field [str, optional] Name of the L field.

Returns

l [dict] L field.

pyart.retrieve.**compute_mean_phase_iq**(*radar*, *signal_field=None*)
Computes the differential phase from the horizontal or vertical channel IQ data

Parameters

radar [IQ radar object] Object containing the required fields

signal_field [str] Name of the field that contain the H or V IQ data. None will use the default field name in the Py-ART configuration file.

Returns

mph_dict [field dictionary] Field dictionary containing the mean phase

pyart.retrieve.**compute_noise_power**(*spectra*, *units='dBADU'*, *navg=1*, *rmin=0.0*, *nnoise_min=1*, *signal_field=None*)

Computes the noise power from the complex spectra in ADU. Requires key dBADU_to_dBm_hh or dBADU_to_dBm_vv in radar_calibration if the units are to be dBm. The noise is computed using the method described in Hildebrand and Sehkon, 1974.

Parameters

spectra [Radar spectra object] Object containing the required fields

units [str] The units of the returned signal. Can be 'ADU', 'dBADU' or 'dBm'

navg [int] Number of spectra averaged

rmin [int] Range from which the data is used to estimate the noise

nnoise_min [int] Minimum number of samples to consider the estimated noise power valid

signal_field [str, optional] Name of the field in radar which contains the signal. None will use the default field name in the Py-ART configuration file.

Returns

noise_dict [field dictionary] Field dictionary containing the noise power

References

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

pyart.retrieve.**compute_noisedBZ**(*nrays*, *noisedBZ_val*, *_range*, *ref_dist*, *noise_field=None*)

Computes noise in dBZ from reference noise value.

Parameters

nrays [int] Number of rays in the reflectivity field.

noisedBZ_val [float] Estimated noise value in dBZ at reference distance.

_range [np array of floats] Range vector in m.

ref_dist [float] Reference distance in Km.

noise_field [str, optional] Name of the noise field.

Returns

noisedBZ [dict] The noise field.

pyart.retrieve.**compute_pol_variables**(*spectra*, *fields_list*, *use_pwr=False*, *subtract_noise=False*, *smooth_window=None*, *srhohv_field=None*, *pwr_h_field=None*, *pwr_v_field=None*, *signal_h_field=None*, *signal_v_field=None*, *noise_h_field=None*, *noise_v_field=None*)

Computes the polarimetric variables from the complex spectra in ADU or the spectral powers and spectral RhoHV

Parameters

spectra [Radar spectra object] Object containing the required fields

fields_list [list of str] list of fields to compute

use_pwr [Bool] If True the polarimetric variables will be computed from the spectral power and the spectral RhoHV. Otherwise from the complex spectra

subtract_noise [Bool] If True noise will be subtracted from the signals

smooth_window [int or None] Size of the moving Gaussian smoothing window. If none no smoothing will be applied

srhohv_field, pwr_h_field, pwr_v_field, signal_h_field, signal_v_field,

noise_h_field, noise_v_field [str] Name of the fields in radar which contains the signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

radar [radar object] Object containing the computed fields

pyart.retrieve.**compute_pol_variables_iq**(*radar*, *fields_list*, *subtract_noise=False*, *lag=0*, *direction='negative_away'*, *phase_offset=0.0*, *signal_h_field=None*, *signal_v_field=None*, *noise_h_field=None*, *noise_v_field=None*)

Computes the polarimetric variables from the IQ signals in ADU

Parameters

radar [IQ radar object] Object containing the required fields

fields_list [list of str] list of fields to compute

subtract_noise [Bool] If True noise will be subtracted from the signals

lag [int] The time lag to use in the estimators

direction [str] The convention used in the Doppler mean field. Can be negative_away or negative_towards

phase_offset [float. Dataset keyword] The system differential phase offset to remove

signal_h_field, signal_v_field, noise_h_field, noise_v_field [str] Name of the fields in radar which contains the signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

radar [radar object] Object containing the computed fields

pyart.retrieve.**compute_qvp**(*radar*, *field_names*, *ref_time=None*, *angle=0.0*, *ang_tol=1.0*, *hmax=10000.0*, *hres=50.0*, *avg_type='mean'*, *nvalid_min=30*, *interp_kind='none'*, *qvp=None*)

Computes quasi vertical profiles.

Parameters

radar [Radar] Radar object used.

field_names [list of str] list of field names to add to the QVP

ref_time [datetime object] reference time for current radar volume

angle [int or float] If the radar object contains a PPI volume, the sweep number to use, if it contains an RHI volume the elevation angle.

ang_tol [float] If the radar object contains an RHI volume, the tolerance in the elevation angle for the conversion into PPI

hmax [float] The maximum height to plot [m].

hres [float] The height resolution [m].

avg_type [str] The type of averaging to perform. Can be either "mean" or "median"

nvalid_min [int] Minimum number of valid points to accept average.

interp_kind [str] type of interpolation when projecting to vertical grid: 'none', or 'nearest', etc. 'none' will select from all data points within the regular grid height bin the closest to the center of the bin. 'nearest' will select the closest data point to the center of the height bin regardless if it is within the height bin or not. Data points can be masked values If another type of interpolation is selected masked values will be eliminated from the data points before the interpolation

qvp [QVP object or None] If it is not None this is the QVP object where to store the data from the current time step. Otherwise a new QVP object will be created

Returns

qvp [qvp object] The computed QVP object

pyart.retrieve.**compute_radial_noise**(*radar*, *ind_rmin=0*, *nbins_min=1*, *max_std_pwr=2.0*, *pwr_field=None*, *noise_field=None*)

Computes radial noise in dBm from signal power

Parameters

radar: radar object radar object containing the signal power in dBm

ind_rmin: int index of the gate nearest to the radar where start looking for noisy gates

nbins_min: int min number of noisy gates to consider the estimation valid

max_std_pwr: float max standard deviation of the noise power to consider the noise valid

pwr_field: str Name of the input signal power field

noise_field: str name of the noise field to use

Returns

noise_dict [dict] the noise field in dBm

pyart.retrieve.**compute_rcs**(*radar*, *kw2=0.93*, *pulse_width=None*, *beamwidth=None*, *freq=None*, *refl_field=None*, *rcs_field=None*)

Computes the radar cross-section (assuming a point target) from radar reflectivity.

Parameters

radar [Radar] radar object

kw2 [float] water constant

pulse_width [float] pulse width [s]

beamwidth [float] beamwidth [degree]

freq [float] radar frequency [Hz]. If none it will be obtained from the radar metadata

refl_field [str] name of the reflectivity used for the calculations

rcs_field [str] name of the RCS field

Returns

rcs_dict [dict] RCS field and metadata

pyart.retrieve.**compute_rcs_from_pr**(*radar*, *lmf=None*, *attg=None*, *radconst=None*, *tx_pwr=None*, *antenna_gain=None*, *lrx=0.0*, *ltx=0.0*, *lradome=0.0*, *freq=None*, *refl_field=None*, *rcs_field=None*, *neglect_gas_att=False*)

Computes the radar cross-section (assuming a point target) from radar reflectivity by first computing the received power and then the RCS from it.

Parameters

radar [Radar] radar object

lmf [float] matched filter losses. If None it will be obtained from the attribute radar_calibration of the radar object

attg [float] 1-way gas attenuation

radconst [float] radar constant

tx_pwr [float] radar transmitted power [dBm]

antenna_gain [float] antenna gain [dB]. If None it will be obtain from the instrument_parameters attribute of the radar object

lrx [float] receiver losses from the antenna feed to the reference point (positive value) [dB]

lradome [float] 1-way losses due to the radome (positive value) [dB]

freq [float] radar frequency [Hz]. If none it will be obtained from the radar metadata

refl_field [str] name of the reflectivity used for the calculations

rcs_field [str] name of the RCS field

neglect_gas_att [bool] Whether to neglect or not gas attenuation in the estimation of the RCS

Returns

rcs_dict [dict] RCS field and metadata

pyart.retrieve.**compute_reflectivity**(*spectra*, *sdBZ_field=None*)

Computes the reflectivity from the spectral reflectivity

Parameters

spectra [Radar spectra object] Object containing the required fields

sdBZ_field [str] Name of the field that contains the spectral reflectivity. None will use the default field name in the Py-ART configuration file.

Returns

dBZ_dict [field dictionary] Field dictionary containing the reflectivity

pyart.retrieve.**compute_reflectivity_iq**(*radar*, *subtract_noise=False*, *signal_field=None*, *noise_field=None*)

Computes the reflectivity from the IQ signal data

Parameters

radar [IQ radar object] Object containing the required fields

subtract_noise [Bool] If true the noise is subtracted from the power

signal_field, noise_field [str] Name of the signal and noise fields. None will use the default field name in the Py-ART configuration file.

Returns

dBZ_dict [field dictionary] Field dictionary containing the reflectivity

pyart.retrieve.**compute_rhohv**(*spectra*, *use_rhohv=False*, *subtract_noise=False*, *srhohv_field=None*, *pwr_h_field=None*, *pwr_v_field=None*, *signal_h_field=None*, *signal_v_field=None*, *noise_h_field=None*, *noise_v_field=None*)

Computes RhoHV from the horizontal and vertical spectral reflectivity or from sRhoHV and the spectral powers

Parameters

spectra [Radar spectra object] Object containing the required fields

use_rhohv [Bool] If true the RhoHV will be computed from sRho_hv. Otherwise it will be computed using the complex spectra

subtract_noise [Bool] If True noise will be subtracted from the signals

srhohv_field, pwr_h_field, pwr_v_field, signal_h_field, signal_v_field,

noise_h_field, noise_v_field [str] Name of the fields in radar which contains the signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

RhoHV_dict [field dictionary] Field dictionary containing the RhoHV

pyart.retrieve.**compute_rhohv_iq**(*radar*, *subtract_noise=False*, *lag=0*, *signal_h_field=None*, *signal_v_field=None*, *noise_h_field=None*, *noise_v_field=None*)

Computes RhoHV from the horizontal and vertical channels IQ data

Parameters

radar [IQ radar object] Object containing the required fields

subtract_noise [Bool] If True noise will be subtracted from the signals

lag [int] Time lag used in the computation

signal_h_field, signal_v_field, noise_h_field, noise_v_field [str] Name of the fields in radar which contains the signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

rhohv_dict [field dictionary] Field dictionary containing the RhoHV

pyart.retrieve.**compute_rqvp**(*radar*, *field_names*, *ref_time=None*, *hmax=10000.0*, *hres=2.0*, *avg_type='mean'*, *nvalid_min=30*, *interp_kind='nearest'*, *rmax=50000.0*, *weight_power=2.0*, *qvp=None*)

Computes range-defined quasi vertical profiles.

Parameters

radar [Radar] Radar object used.

field_names [list of str] list of field names to add to the QVP

ref_time [datetime object] reference time for current radar volume

hmax [float] The maximum height to plot [m].

hres [float] The height resolution [m].

avg_type [str] The type of averaging to perform. Can be either "mean" or "median"

nvalid_min [int] Minimum number of valid points to accept average.

interp_kind [str] type of interpolation when projecting to vertical grid: 'none', or 'nearest', etc. 'none' will select from all data points within the regular grid height bin the closest to the center of the bin. 'nearest' will select the closest data point to the center of the height bin regardless if it is within the height bin or not. Data points can be masked values If another type of interpolation is selected masked values will be eliminated from the data points before the interpolation

rmax [float] ground range up to which the data is intended for use [m].

weight_power [float] Power p of the weighting function 1/abs(grng-(rmax-1))**p given to the data outside the desired range. -1 will set the weight to 0.

qvp [QVP object or None] If it is not None this is the QVP object where to store the data from the current time step. Otherwise a new QVP object will be created

Returns

qvp [qvp object] The computed range defined quasi vertical profile

pyart.retrieve.**compute_signal_power**(*radar*, *lmf=None*, *attg=None*, *radconst=None*, *lrx=0.0*, *lradome=0.0*, *refl_field=None*, *pwr_field=None*)

Computes received signal power OUTSIDE THE RADOME in dBm from a reflectivity field.

Parameters

radar [Radar] radar object

lmf [float] matched filter losses

attg [float] 1-way gas attenuation

radconst [float] radar constant

lrx [float] receiver losses from the antenna feed to the reference point (positive value) [dB]

lradome [float] 1-way losses due to the radome (positive value) [dB]

refl_field [str] name of the reflectivity used for the calculations

pwr_field [str] name of the signal power field

Returns

s_pwr_dict [dict] power field and metadata

pyart.retrieve.**compute_snr**(*radar*, *refl_field=None*, *noise_field=None*, *snr_field=None*)

Computes SNR from a reflectivity field and the noise in dBZ.

Parameters

radar [Radar] Radar object

refl_field [str, optional] Name of the reflectivity field to use.

noise_field [str, optional] Name of the noise field to use.

snr_field [str, optional] Name of the SNR field.

Returns

snr [dict] The SNR field.

pyart.retrieve.**compute_spectra**(*radar*, *fields_in_list*, *fields_out_list*, *window=None*)

Computes the spectra from IQ data through a Fourier transform

Parameters

radar [radar object] Object containing the IQ data

fields_in_list [list of str] list of input IQ data fields names

fields_out_list [list of str] list with the output spectra fields names obtained from the input fields

window [string, tupple or None] Parameters of the window used to obtain the spectra. The parameters are the ones corresponding to function scipy.signal.windows.get_window. If None no window will be used

Returns

spectra [spectra radar object] radar object containing the spectra fields

pyart.retrieve.**compute_spectral_differential_phase**(*spectra*, *use_rhohv=False*, *srhohv_field=None*, *signal_h_field=None*, *signal_v_field=None*)

Computes the spectral differential reflectivity from the complex spectras in ADU or sRhoHV

Parameters

spectra [Radar spectra object] Object containing the required fields

use_rhohv [Bool] If true sRhoHV is going to be used to compute the differential phase. Otherwise the complex signals are used

signal_h_field, signal_v_field [str] Name of the fields in radar which contains the signal. None will use the default field name in the Py-ART configuration file.

Returns

sPhiDP_dict [field dictionary] Field dictionary containing the spectral differential phase

pyart.retrieve.**compute_spectral_differential_reflectivity**(*spectra*, *compute_power=True*, *subtract_noise=False*, *smooth_window=None*, *pwr_h_field=None*, *pwr_v_field=None*, *signal_h_field=None*, *signal_v_field=None*, *noise_h_field=None*, *noise_v_field=None*)

Computes the spectral differential reflectivity from the complex spectras or the power in ADU

Parameters

spectra [Radar spectra object] Object containing the required fields

compute_power [Bool] If True the signal power will be computed. Otherwise the field given by the user will be used

subtract_noise [Bool] If True noise will be subtracted from the signals

smooth_window [int or None] Size of the moving Gaussian smoothing window. If none no smoothing will be applied

pwr_h_field, pwr_v_field, signal_h_field, signal_v_field, noise_h_field,

noise_v_field [str] Name of the fields in radar which contains the signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

sZDR_dict [field dictionary] Field dictionary containing the spectral differential reflectivity

pyart.retrieve.**compute_spectral_noise**(*spectra*, *units='dBADU'*, *navg=1*, *rmin=0.0*, *nnoise_min=1*, *signal_field=None*)

Computes the spectral noise power from the complex spectra in ADU. Requires key dBADU_to_dBm_hh or dBADU_to_dBm_vv in radar_calibration if the units are to be dBm. The noise is computed using the method described in Hildebrand and Sehkon, 1974.

Parameters

spectra [Radar spectra object] Object containing the required fields

units [str] The units of the returned signal. Can be 'ADU', 'dBADU' or 'dBm'

navg [int] Number of spectra averaged

rmin [int] Range from which the data is used to estimate the noise

nnoise_min [int] Minimum number of samples to consider the estimated noise power valid

signal_field [str, optional] Name of the field in radar which contains the signal. None will use the default field name in the Py-ART configuration file.

Returns

noise_dict [field dictionary] Field dictionary containing the spectral noise power

References

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

pyart.retrieve.**compute_spectral_phase**(*spectra*, *signal_field=None*)

Computes the spectral phase from the complex spectra in ADU

Parameters

spectra [Radar spectra object] Object containing the required fields

signal_field [str, optional] Name of the field in radar which contains the signal. None will use the default field name in the Py-ART configuration file.

Returns

phase_dict [field dictionary] Field dictionary containing the spectral phase

pyart.retrieve.**compute_spectral_power**(*spectra*, *units='dBADU'*, *subtract_noise=False*, *smooth_window=None*, *signal_field=None*, *noise_field=None*)

Computes the spectral power from the complex spectra in ADU. Requires key dBADU_to_dBm_hh or dBADU_to_dBm_vv in radar_calibration if the units are to be dBm

Parameters

spectra [Radar spectra object] Object containing the required fields

units [str] The units of the returned signal. Can be 'ADU', 'dBADU' or 'dBm'

subtract_noise [Bool] If True noise will be subtracted from the signal

smooth_window [int or None] Size of the moving Gaussian smoothing window. If none no smoothing will be applied

signal_field, noise_field [str, optional] Name of the fields in radar which contains the signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

pwr_dict [field dictionary] Field dictionary containing the spectral power

pyart.retrieve.**compute_spectral_reflectivity**(*spectra*, *compute_power=True*, *subtract_noise=False*, *smooth_window=None*, *pwr_field=None*, *signal_field=None*, *noise_field=None*)

Computes the spectral reflectivity from the complex spectra in ADU or from the signal power in ADU. Requires keys dBADU_to_dBm_hh or dBADU_to_dBm_vv in radar_calibration if the to be computed

Parameters

spectra [Radar spectra object] Object containing the required fields

compute_power [Bool] If True the signal power will be computed. Otherwise the field given by the user will be used

subtract_noise [Bool] If True noise will be subtracted from the signal

smooth_window [int or None] Size of the moving Gaussian smoothing window. If none no smoothing will be applied

pwr_field, signal_field, noise_field [str, optional] Name of the fields in radar which contains the signal power, complex signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

sdBZ_dict [field dictionary] Field dictionary containing the spectral reflectivity

pyart.retrieve.**compute_spectral_rhohv**(*spectra*, *subtract_noise=False*, *signal_h_field=None*, *signal_v_field=None*, *noise_h_field=None*, *noise_v_field=None*)

Computes the spectral RhoHV from the complex spectras in ADU

Parameters

spectra [Radar spectra object] Object containing the required fields

subtract_noise [Bool] If True noise will be subtracted from the signals

signal_h_field, signal_v_field, noise_h_field, noise_v_field [str] Name of the fields in radar which contains the signal and noise. None will use the default field name in the Py-ART configuration file.

Returns

sRhoHV_dict [field dictionary] Field dictionary containing the spectral RhoHV

pyart.retrieve.**compute_st1_iq**(*radar*, *signal_field=None*)

Computes the statistical test one lag fluctuation from the horizontal or vertical channel IQ data

Parameters

radar [IQ radar object] Object containing the required fields

signal_field [str] Name of the field that contain the H or V IQ data. None will use the default field name in the Py-ART configuration file.

Returns

st1_dict [field dictionary] Field dictionary containing the st1

pyart.retrieve.**compute_st2_iq**(*radar*, *signal_field=None*)

Computes the statistical test two lag fluctuation from the horizontal or vertical channel IQ data

Parameters

radar [IQ radar object] Object containing the required fields

signal_field [str] Name of the field that contain the H or V IQ data. None will use the default field name in the Py-ART configuration file.

Returns

st2_dict [field dictionary] Field dictionary containing the st2

pyart.retrieve.**compute_svp**(*radar*, *field_names*, *lon*, *lat*, *angle*, *ref_time=None*, *ang_tol=1.0*, *latlon_tol=0.0005*, *delta_rng=15000.0*, *delta_azi=10*, *hmax=10000.0*, *hres=250.0*, *avg_type='mean'*, *nvalid_min=1*, *interp_kind='none'*, *qvp=None*)

Computes slanted vertical profiles.

Parameters

radar [Radar] Radar object used.

field_names [list of str] list of field names to add to the QVP

lat, lon [float] latitude and longitude of the point of interest [deg]

angle [int or float] If the radar object contains a PPI volume, the sweep number to use, if it contains an RHI volume the elevation angle.

ref_time [datetime object] reference time for current radar volume

ang_tol [float] If the radar object contains an RHI volume, the tolerance in the elevation angle for the conversion into PPI

latlon_tol [float] tolerance in latitude and longitude in deg.

delta_rng, delta_azi [float] maximum range distance [m] and azimuth distance [degree] from the central point of the evp containing data to average.

hmax [float] The maximum height to plot [m].

hres [float] The height resolution [m].

avg_type [str] The type of averaging to perform. Can be either “mean” or “median”

nvalid_min [int] Minimum number of valid points to accept average.

interp_kind [str] type of interpolation when projecting to vertical grid: ‘none’, or ‘nearest’, etc. ‘none’ will select from all data points within the regular grid height bin the closest to the center of the bin. ‘nearest’ will select the closest data point to the center of the height bin regardless if it is within the height bin or not. Data points can be masked values If another type of interpolation is selected masked values will be eliminated from the data points before the interpolation

qvp [QVP object or None] If it is not None this is the QVP object where to store the data from the current time step. Otherwise a new QVP object will be created

Returns

qvp [qvp object] The computed slanted vertical profile

pyart.retrieve.**compute_ts_along_coord**(*radar*, *field_name*, *mode='ALONG_AZI'*, *fixed_range=None*, *fixed_azimuth=None*, *fixed_elevation=None*, *ang_tol=1.0*, *rng_tol=50.0*, *value_start=None*, *value_stop=None*, *ref_time=None*, *acoord=None*)

Computes time series along a particular antenna coordinate, i.e. along azimuth, elevation or range

Parameters

radar [Radar] Radar object used.

field_name [str] Name of the field

mode [str] coordinate to extract data along. Can be ALONG_AZI, ALONG_ELE or ALONG_RNG

fixed_range, fixed_azimuth, fixed_elevation [float] The fixed range [m], azimuth [deg] or elevation [deg] to extract. In each mode two of these parameters have to be defined. If they are not defined they default to 0.

ang_tol, rng_tol [float] The angle tolerance [deg] and range tolerance [m] around the fixed range or azimuth/elevation

value_start, value_stop [float] The minimum and maximum value at which the data along a coordinate start and stop

ref_time [datetime object] reference time for current radar volume

acoord [acoord object or None] If it is not None this is the object where to store the data from the current time step. Otherwise a new acoord object will be created

Returns

acoord [acoord object] The computed data along a coordinate

pyart.retrieve.**compute_vol_refl**(*radar*, *kw=0.93*, *freq=None*, *refl_field=None*, *vol_refl_field=None*)

Computes the volumetric reflectivity from the effective reflectivity factor

Parameters

radar [Radar] radar object

kw [float] water constant

freq [None or float] radar frequency

refl_field [str] name of the reflectivity used for the calculations

vol_refl_field [str] name of the volumetric reflectivity

Returns

vol_refl_dict [dict] volumetric reflectivity and metadata in 10log10(cm^2 km^-3)

pyart.retrieve.**compute_vp**(*radar*, *field_names*, *lon*, *lat*, *ref_time=None*, *latlon_tol=0.0005*, *hmax=10000.0*, *hres=50.0*, *interp_kind='none'*, *qvp=None*)

Computes vertical profiles.

Parameters

radar [Radar] Radar object used.

field_names [list of str] list of field names to add to the QVP

lat, lon [float] latitude and longitude of the point of interest [deg]

ref_time [datetime object] reference time for current radar volume

latlon_tol [float] tolerance in latitude and longitude in deg.

hmax [float] The maximum height to plot [m].

hres [float] The height resolution [m].

interp_kind [str] type of interpolation when projecting to vertical grid: 'none', or 'nearest', etc. 'none' will select from all data points within the regular grid height bin the closest to the center of the bin. 'nearest' will select the closest data point to the center of the height bin regardless if it is within the height bin or not. Data points can be masked values If another type of interpolation is selected masked values will be eliminated from the data points before the interpolation

qvp [QVP object or None] If it is not None this is the QVP object where to store the data from the current time step. Otherwise a new QVP object will be created

Returns

qvp [qvp object] The computed vertical profile

pyart.retrieve.**compute_wbn_iq**(*radar*, *signal_field=None*)

Computes the wide band noise from the horizontal or vertical channel IQ data

Parameters

radar [IQ radar object] Object containing the required fields

signal_field [str] Name of the field that contain the H or V IQ data. None will use the default field name in the Py-ART configuration file.

Returns

wbn_dict [field dictionary] Field dictionary containing the wide band noise

pyart.retrieve.**detect_ml**(*radar*, *gatefilter=None*, *fill_value=None*, *refl_field=None*, *rhohv_field=None*, *ml_field=None*, *ml_pos_field=None*, *iso0_field=None*, *max_range=20000*, *detect_threshold=0.02*, *interp_holes=False*, *max_length_holes=250*, *check_min_length=True*, *get_iso0=False*)

Detects the melting layer (ML) using the reflectivity and copolar correlation coefficient. Internally it uses RHIs

Returns

ml_obj [radar-like object] A radar-like object containing the field melting layer height with the bottom (at range position 0) and top (at range position one) of the melting layer at each ray

ml_dict [dict] A dictionary containg the position of the range gate respect to the melting layer and metadata

iso0_dict [dict or None] A dictionary containing the distance respect to the melting layer and metadata

all_ml [dict] Dictionary containing internal parameters in polar and cartesian coordinates

pyart.retrieve.**est_rain_rate_a**(*radar*, *alpha=None*, *beta=None*, *a_field=None*, *rr_field=None*)

Estimates rainfall rate from specific attenuation using alpha power law.

Parameters

radar [Radar] Radar object.

alpha, beta [floats, optional] Factor (alpha) and exponent (beta) of the power law. If not set the factors are going to be determined according to the radar frequency.

a_field [str, optional] Name of the specific attenuation field to use.

rr_field [str, optional] Name of the rainfall rate field.

Returns

rain [dict] Field dictionary containing the rainfall rate.

References

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

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.

pyart.retrieve.**est_rain_rate_hydro**(*radar*, *alphazr=0.0376*, *betazr=0.6112*, *alphazs=0.1*, *betazs=0.5*, *alphaa=None*, *betaa=None*, *mp_factor=0.6*, *refl_field=None*, *a_field=None*, *hydro_field=None*, *rr_field=None*, *master_field=None*, *thresh=None*, *thresh_max=False*)

Estimates rainfall rate using different relations between R and the polarimetric variables depending on the hydrometeor type.

Parameters

radar [Radar] Radar object.

alphazr, betazr [floats, optional] Factor (alpha) and exponent (beta) of the z-r power law for rain.

alphazs, betazs [floats, optional] Factor (alpha) and exponent (beta) of the z-s power law for snow.

alphaa, betaa [floats, optional] Factor (alpha) and exponent (beta) of the a-r power law. If not set the factors are going to be determined according to the radar frequency.

mp_factor [float, optional] Factor applied to z-r relation in the melting layer.

refl_field [str, optional] Name of the reflectivity field to use.

a_field [str, optional] Name of the specific attenuation field to use.

hydro_field [str, optional] Name of the hydrometeor classification field to use.

rr_field [str, optional] Name of the rainfall rate field.

master_field [str, optional] Name of the field that is going to act as master. Has to be either refl_field or kdp_field. Default is refl_field.

thresh [float, optional] Value of the threshold that determines when to use the slave field.

thresh_max [Bool, optional] If true the master field is used up to the thresh value maximum. Otherwise the master field is not used below thresh value.

Returns

rain [dict] Field dictionary containing the rainfall rate.

pyart.retrieve.**est_rain_rate_kdp**(*radar*, *alpha=None*, *beta=None*, *kdp_field=None*, *rr_field=None*)

Estimates rainfall rate from kdp using alpha power law.

Parameters

radar [Radar] Radar object.

alpha, beta [floats, optional] Factor (alpha) and exponent (beta) of the power law. If not set the factors are going to be determined according to the radar frequency.

kdp_field [str, optional] Name of the specific differential phase field to use.

rr_field [str, optional] Name of the rainfall rate field.

Returns

rain [dict] Field dictionary containing the rainfall rate.

pyart.retrieve.**est_rain_rate_z**(*radar*, *alpha=0.0376*, *beta=0.6112*, *refl_field=None*, *rr_field=None*)

Estimates rainfall rate from reflectivity using a power law.

Parameters

radar [Radar] Radar object.

alpha, beta [floats, optional] Factor (alpha) and exponent (beta) of the power law.

refl_field [str, optional] Name of the reflectivity field to use.

rr_field [str, optional] Name of the rainfall rate field.

Returns

rain [dict] Field dictionary containing the rainfall rate.

pyart.retrieve.**est_rain_rate_za**(*radar*, *alphaz=0.0376*, *betaz=0.6112*, *alphaa=None*, *betaa=None*, *refl_field=None*, *a_field=None*, *rr_field=None*, *master_field=None*, *thresh=None*, *thresh_max=True*)

Estimates rainfall rate from a blending of power law r-alpha and r-z relations.

Parameters

radar [Radar] Radar object

alphaz, betaz [floats, optional] Factor (alpha) and exponent (beta) of the z-r power law.

alphaa,betaa [floats, optional] Factor (alpha) and exponent (beta) of the a-r power law. If not set the factors are going to be determined according to the radar frequency.

refl_field [str, optional] Name of the reflectivity field to use.

a_field [str, optional] Name of the specific attenuation field to use.

rr_field [str, optional] Name of the rainfall rate field.

master_field [str, optional] Name of the field that is going to act as master. Has to be either refl_field or kdp_field. Default is refl_field.

thresh [float, optional] Value of the threshold that determines when to use the slave field.

thresh_max [Bool, optional] If true the master field is used up to the thresh value maximum. Otherwise the master field is not used below thresh value.

Returns

rain_master [dict] Field dictionary containing the rainfall rate.

pyart.retrieve.**est_rain_rate_zkdp**(*radar*, *alphaz=0.0376*, *betaz=0.6112*, *alphakdp=None*, *betakdp=None*, *refl_field=None*, *kdp_field=None*, *rr_field=None*, *master_field=None*, *thresh=None*, *thresh_max=True*)

Estimates rainfall rate from a blending of power law r-kdp and r-z relations.

Parameters

radar [Radar] Radar object.

alphaz, betaz [floats, optional] Factor (alpha) and exponent (beta) of the z-r power law.

alphakdp, betakdp [floats, optional] Factor (alpha) and exponent (beta) of the kdp-r power law. If not set the factors are going to be determined according to the radar frequency.

refl_field [str, optional] Name of the reflectivity field to use.

kdp_field [str, optional] Name of the specific differential phase field to use.

rr_field [str, optional] Name of the rainfall rate field.

master_field [str, optional] Name of the field that is going to act as master. Has to be either refl_field or kdp_field. Default is refl_field.

thresh [float, optional] Value of the threshold that determines when to use the slave field [mm/h].

thresh_max [Bool, optional] If true the master field is used up to the thresh value maximum. Otherwise the master field is not used below thresh value.

Returns

rain_master [dict] Field dictionary containing the rainfall rate.

pyart.retrieve.**est_rain_rate_zpoly**(*radar*, *refl_field=None*, *rr_field=None*)

Estimates rainfall rate from reflectivity using a polynomial Z-R relation developed at McGill University.

Parameters

radar [Radar] Radar object.

refl_field [str, optional] Name of the reflectivity field to use.

rr_field [str, optional] Name of the rainfall rate field.

Returns

rain [dict] Field dictionary containing the rainfall rate.

pyart.retrieve.**est_vertical_windshear**(*radar*, *az_tol=0.5*, *wind_field=None*, *windshear_field=None*)

Estimates wind shear.

Parameters

radar [Radar] Radar object

az_tol [float] azimuth tolerance to consider gate on top of selected one

wind_field [str] name of the horizontal wind velocity field

windshear_field [str] name of the vertical wind shear field

Returns

windshear [dict] Field dictionary containing the wind shear field

pyart.retrieve.**est_wind_profile**(*radar*, *npoints_min=6*, *azi_spacing_max=45.0*, *vel_diff_max=10.0*, *sign=1*, *rad_vel_field=None*, *u_vel_field=None*, *v_vel_field=None*, *w_vel_field=None*, *vel_est_field=None*, *vel_std_field=None*, *vel_diff_field=None*)

Estimates the vertical wind profile using VAD techniques

Parameters

radar [Radar] Radar object

npoints_min [int] Minimum number of points in the VAD to retrieve wind components. 0 will retrieve them regardless

azi_spacing_max [float] Maximum spacing between valid gates in the VAD to retrieve wind components. 0 will retrieve them regardless.

vel_diff_max [float] Maximum velocity difference allowed between retrieved and measured radial velocity at each range gate. Gates exceeding this threshold will be removed and VAD will be recomputed. If -1 there will not be a second pass.

sign [int, optional] Sign convention which the radial velocities in the volume created from the sounding data will will. This should match the convention used in the radar data. A value of 1 represents when positive values velocities are towards the radar, -1 represents when negative velocities are towards the radar.

rad_vel_field [str] name of the measured radial velocity field

u_vel_field, v_vel_field, w_vel_field [str] names of the 3 wind components fields

vel_est_field [str] name of the retrieved radial Doppler velocity field

vel_std_field [str] name of the standard deviation of the velocity retrieval field

vel_diff_field [str] name of the diference between retrieved and measured radial velocity field

Returns

wind [dict] Field dictionary containing the estimated wind velocity

pyart.retrieve.**est_wind_vel**(*radar*, *vert_proj=False*, *vel_field=None*, *wind_field=None*)

Estimates wind velocity. Projects the radial wind component to the horizontal or vertical of the azimuth plane. It assumes that the orthogonal component is negligible.

The horizontal wind component is given by: v = v_r*cos(el)-v_el*sin(el)+v_az

where: v_r is the radial wind component (measured by the radar) v_el is the perpendicular wind component in the azimuth plane. v_az is the horizontal component perpendicular to the radial direction and the azimuth plane el is the elevation

The horizontal wind component in the azimuth plane is given by: v_h = v_r*cos(el) - v_el*sin(el)

which since we do not know v_el we assume: v_h ~ v_r*cos(el)

This assumption holds for small elevation angles

The vertical wind component in the azimuth plane is given by: v_h = v_r*sin(el) - v_el*cos(el)

which since we do not know v_el we assume: v_h ~ v_r*sin(el)

This assumption holds for angles close to 90 deg

Parameters

radar [Radar] Radar object

vert_proj [Boolean] If true estimates the vertical projection, otherwise the horizontal

vel_field [str] name of the velocity field

wind_field [str] name of the velocity field

Returns

wind [dict] Field dictionary containing the estimated wind velocity

pyart.retrieve.**fetch_radar_time_profile**(*sonde_dset*, *radar*, *time_key='time'*, *height_key='height'*, *nvars=None*)

Extract the correct profile from a interpolated sonde.

This is an ARM specific method which extract the correct profile out of netCDF Variables from a Interpolated Sonde VAP for the volume start time of a radar object.

Parameters

sonde_dset [Dataset] Interpolate sonde Dataset.

radar [Radar] Radar object from which the nearest profile will be found.

time_key [string, optional] Key to find a CF startard time variable.

height_key [string, optional] Key to find profile height data.

nvars [list, optional] NetCDF variable to generated profiles for. If None (the default) all variables with dimension of time, height will be found in ncvars.

Returns

return_dic [dict] Profiles at the start time of the radar.

pyart.retrieve.**get_coeff_attg**(*freq*)

get the 1-way gas attenuation for a particular frequency

Parameters

freq [float] radar frequency [Hz]

Returns

attg [float] 1-way gas attenuation

pyart.retrieve.**get_freq_band**(*freq*)

Returns the frequency band name (S, C, X, ...).

Parameters

freq [float] Radar frequency [Hz].

Returns

freq_band [str] Frequency band name.

pyart.retrieve.**grid_displacement_pc**(*grid1*, *grid2*, *field*, *level*, *return_value='pixels'*)

Calculate the grid displacement using phase correlation.

See: http://en.wikipedia.org/wiki/Phase_correlation

Implementation inspired by Christoph Gohlke: http://www.lfd.uci.edu/~gohlke/code/imreg.py.html

Note that the grid must have the same dimensions in x and y and assumed to have constant spacing in these dimensions.

Parameters

grid1, grid2 [Grid] Py-ART Grid objects separated in time and square in x/y.

field [string] Field to calculate advection from. Field must be in both grid1 and grid2.

level [integer] The vertical (z) level of the grid to use in the calculation.

return_value [str, optional] 'pixels', 'distance' or 'velocity'. Distance in pixels (default) or meters or velocity vector in m/s.

Returns

displacement [two-tuple] Calculated displacement in units of y and x. Value returned in integers if pixels, otherwise floats.

pyart.retrieve.**grid_shift**(*grid*, *advection*, *trim_edges=0*, *field_list=None*)
Shift a grid by a certain number of pixels.

Parameters

grid: Grid Py-ART Grid object.

advection [two-tuple of floats] Number of Pixels to shift the image by.

trim_edges: integer, optional Edges to cut off the grid and axes, both x and y. Defaults to zero.

field_list [list, optional] List of fields to include in new grid. None, the default, includes all fields from the input grid.

Returns

shifted_grid [Grid] Grid with fields shifted and, if requested, subset.

pyart.retrieve.**hydroclass_semisupervised**(*radar*, *mass_centers=None*, *weights=array([1., 1., 1., 0.75, 0.5])*, *value=50.0*, *refl_field=None*, *zdr_field=None*, *rhv_field=None*, *kdp_field=None*, *temp_field=None*, *iso0_field=None*, *hydro_field=None*, *entropy_field=None*, *temp_ref='temperature'*, *compute_entropy=False*, *output_distances=False*, *vectorize=False*)
Classifies precipitation echoes following the approach by Besic et al (2016).

Parameters

radar [radar] Radar object.

mass_centers [ndarray 2D, optional] The centroids for each variable and hydrometeor class in (nclasses, nvariables).

weights [ndarray 1D, optional] The weight given to each variable.

value [float] The value controlling the rate of decay in the distance transformation

refl_field, zdr_field, rhv_field, kdp_field, temp_field, iso0_field [str] Inputs. Field names within the radar object which represent the horizonal reflectivity, the differential reflectivity, the copolar correlation coefficient, the specific differential phase, the temperature and the height respect to the iso0 fields. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

hydro_field [str] Output. Field name which represents the hydrometeor class field. A value of None will use the default field name as defined in the Py-ART configuration file.

temp_ref [str] the field use as reference for temperature. Can be either temperature or height_over_iso0

compute_entropy [bool] If true, the entropy is computed

output_distances [bool] If true, the normalized distances to the centroids for each hydrometeor are provided as output

vectorize [bool] If true, a vectorized version of the class assignation is going to be used

Returns

fields_dict [dict] Dictionary containing the retrieved fields

References

Besic, N., Figueras i Ventura, J., Grazioli, J., Gabella, M., Germann, U., and Berne, A.: Hydrometeor classification through statistical clustering of polarimetric radar measurements: a semi-supervised approach, Atmos. Meas. Tech., 9, 4425-4445, doi:10.5194/amt-9-4425-2016, 2016

pyart.retrieve.**kdp_leastsquare_double_window**(*radar*, *swind_len=11*, *smin_valid=6*, *lwind_len=31*, *lmin_valid=16*, *zthr=40.0*, *phidp_field=None*, *refl_field=None*, *kdp_field=None*, *vectorize=False*)

Compute the specific differential phase (KDP) from differential phase data using a piecewise least square method. For optimal results PhiDP should be already smoothed and clutter filtered out.

Parameters

radar [Radar] Radar object.

swind_len [int] The lenght of the short moving window.

smin_valid [int] Minimum number of valid bins to consider the retrieval valid when using the short moving window

lwind_len [int] The lenght of the long moving window.

lmin_valid [int] Minimum number of valid bins to consider the retrieval valid when using the long moving window

zthr [float] reflectivity value above which the short window is used

phidp_field [str] Field name within the radar object which represent the differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

refl_field [str] Field name within the radar object which represent the reflectivity. A value of None will use the default field name as defined in the Py-ART configuration file.

kdp_field [str] Field name within the radar object which represent the specific differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

vectorize [bool] whether to use a vectorized version of the least square method

Returns

kdp_dict [dict] Retrieved specific differential phase data and metadata.

pyart.retrieve.**kdp_leastsquare_single_window**(*radar*, *wind_len=11*, *min_valid=6*, *phidp_field=None*, *kdp_field=None*, *vectorize=False*)

Compute the specific differential phase (KDP) from differential phase data using a piecewise least square method. For optimal results PhiDP should be already smoothed and clutter filtered out.

Parameters

radar [Radar] Radar object.

wind_len [int] The lenght of the moving window.

min_valid [int] Minimum number of valid bins to consider the retrieval valid

phidp_field [str] Field name within the radar object which represent the differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

kdp_field [str] Field name within the radar object which represent the specific differential phase shift. A value of None will use the default field name as defined in the Py-ART configuration file.

vectorize [bool] whether to use a vectorized version of the least square method

Returns

kdp_dict [dict] Retrieved specific differential phase data and metadata.

pyart.retrieve.**kdp_maesaka**(*radar*, *gatefilter=None*, *method='cg'*, *backscatter=None*, *Clpf=1.0*, *length_scale=None*, *first_guess=0.01*, *finite_order='low'*, *fill_value=None*, *proc=1*, *psidp_field=None*, *kdp_field=None*, *phidp_field=None*, *debug=False*, *verbose=False*, ***kwargs*)

Compute the specific differential phase (KDP) from corrected (e.g., unfolded) total differential phase data based on the variational method outlined in Maesaka et al. (2012). This method assumes a monotonically increasing propagation differential phase (PHIDP) with increasing range from the radar, and therefore is limited to rainfall below the melting layer and/or warm clouds at weather radar frequencies (e.g., S-, C-, and X-band). This method currently only supports radar data with constant range resolution.

Following the notation of Maesaka et al. (2012), the primary control variable k is proportional to KDP,

k**2 = 2 * KDP * dr

which, because of the square, assumes that KDP always takes a positive value.

Parameters

radar [Radar] Radar containing differential phase field.

gatefilter [GateFilter] A GateFilter indicating radar gates that should be excluded when analysing differential phase measurements.

method [str, optional] Type of scipy.optimize method to use when minimizing the cost functional. The default method uses a nonlinear conjugate gradient algorithm. In Maesaka et al. (2012) they use the Broyden-Fletcher- Goldfarb-Shanno (BFGS) algorithm, however for large functional size (e.g., 100K+ variables) this algorithm is considerably slower than a conjugate gradient algorithm.

backscatter [optional] Define the backscatter differential phase. If None, the backscatter differential phase is set to zero for all range gates. Note that backscatter differential phase can be parameterized using attentuation corrected differential reflectivity.

Clpf [float, optional] The low-pass filter (radial smoothness) constraint weight as in equation (15) of Maesaka et al. (2012).

length_scale [float, optional] Length scale in meters used to bring the dimension and magnitude of the low-pass filter cost functional in line with the observation cost functional. If None, the length scale is set to the range resolution.

first_guess [float, optional] First guess for control variable k. Since k is proportional to the square root of KDP, the first guess should be close to zero to signify a KDP field close to 0 deg/km everywhere. However, the first guess should not be exactly zero in order to avoid convergence criteria after the first iteration. In fact it is recommended to use a value closer to one than zero.

finite_order ['low' or 'high', optional] The finite difference accuracy to use when computing derivatives.

maxiter [int, optional] Maximum number of iterations to perform during cost functional minimization. The maximum number of iterations are only performed if convergence criteria are not met. For variational schemes such as this one, it is generally not recommended to try and achieve convergence criteria since the values of the cost functional and/or its gradient norm are somewhat arbitrary.

fill_value [float, optional] Value indicating missing or bad data in differential phase field.

proc [int, optional] The number of parallel threads (CPUs) to use. Currently no multiprocessing capability exists.

psidp_field [str, optional] Total differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

kdp_field [str, optional] Specific differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

phidp_field [str, optional] Propagation differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

debug [bool, optional] True to print debugging information, False to suppress.

verbose [bool, optional] True to print relevant information, False to suppress.

Returns

kdp_dict [dict] Retrieved specific differential phase data and metadata.

phidpf_dict, phidpr_dict [dict] Retrieved forward and reverse direction propagation differential phase data and metadata.

References

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.

pyart.retrieve.**kdp_schneebeli**(*radar*, *gatefilter=None*, *fill_value=None*, *psidp_field=None*, *kdp_field=None*, *phidp_field=None*, *band='C'*, *rcov=0*, *pcov=0*, *prefilter_psidp=False*, *filter_opt=None*, *parallel=True*)

Estimates Kdp with the Kalman filter method by Schneebeli and al. (2014) for a set of psidp measurements.

Parameters

radar [Radar] Radar containing differential phase field.

gatefilter [GateFilter, optional] A GateFilter indicating radar gates that should be excluded when analysing differential phase measurements.

fill_value [float, optional] Value indicating missing or bad data in differential phase field, if not specified, the default in the Py-ART configuration file will be used.

psidp_field [str, optional] Total differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

kdp_field [str, optional] Specific differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

phidp_field [str, optional] Propagation differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

band [char, optional] Radar frequency band string. Accepted "X", "C", "S" (capital or not). The band is used to compute intercepts -c and slope b of the delta = b*Kdp+c relation.

rcov [3x3 float array, optional] Measurement error covariance matrix.

pcov [4x4 float array, optional] Scaled state transition error covariance matrix.

prefilter_psidp [bool, optional] If set, the psidp measurements will first be filtered with the filter_psidp method, which can improve the quality of the final Kdp.

filter_opt [dict, optional] The arguments for the prefilter_psidp method, if empty, the defaults arguments of this method will be used.

parallel [bool, optional] Flag to enable parallel computation (one core for every psidp profile).

Returns

kdp_dict [dict] Retrieved specific differential phase data and metadata.

kdp_std_dict [dict] Estimated specific differential phase standard dev. data and metadata.

phidpr_dict,: dict Retrieved differential phase data and metadata.

References

Schneebeli, M., Grazioli, J., and Berne, A.: Improved Estimation of the Specific Differential Phase SHIFT Using a Compilation of Kalman Filter Ensembles, IEEE T. Geosci. Remote Sens., 52, 5137-5149, doi:10.1109/TGRS.2013.2287017, 2014.

pyart.retrieve.**kdp_vulpiani**(*radar*, *gatefilter=None*, *fill_value=None*, *psidp_field=None*, *kdp_field=None*, *phidp_field=None*, *band='C'*, *windsize=10*, *n_iter=10*, *interp=False*, *prefilter_psidp=False*, *filter_opt=None*, *parallel=False*)

Estimates Kdp with the Vulpiani method for a 2D array of psidp measurements with the first dimension being the distance from radar and the second dimension being the angles (azimuths for PPI, elev for RHI).The input psidp is assumed to be pre-filtered (for ex. with the filter_psidp function)

Parameters

radar [Radar] Radar containing differential phase field.

gatefilter [GateFilter, optional] A GateFilter indicating radar gates that should be excluded when analysing differential phase measurements.

fill_value [float, optional] Value indicating missing or bad data in differential phase field, if not specified, the default in the Py-ART configuration file will be used

psidp_field [str, optional] Total differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

kdp_field [str, optional] Specific differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

phidp_field [str, optional] Propagation differential phase field. If None, the default field name must be specified in the Py-ART configuration file.

band [char, optional] Radar frequency band string. Accepted "X", "C", "S" (capital or not). It is used to set default boundaries for expected values of Kdp.

windsize [int, optional] Size in # of gates of the range derivative window. Should be even.

n_iter [int, optional] Number of iterations of the method. Default is 10.

interp [bool, optional] If True, all the nans are interpolated.The advantage is that less data are lost (the iterations in fact are "eating the edges") but some non-linear errors may be introduced.

prefilter_psidp [bool, optional] If set, the psidp measurements will first be filtered with the filter_psidp method, which can improve the quality of the final Kdp.

filter_opt [dict, optional] The arguments for the prefilter_psidp method, if empty, the defaults arguments of this method will be used.

parallel [bool, optional] Flag to enable parallel computation (one core for every psidp profile).

Returns

kdp_dict [dict] Retrieved specific differential phase data and metadata.

phidpr_dict,: dict Retrieved differential phase data and metadata.

References

Gianfranco Vulpiani, Mario Montopoli, Luca Delli Passeri, Antonio G. Gioia, Pietro Giordano, and Frank S. Marzano, 2012: On the Use of Dual-Polarized C-Band Radar for Operational Rainfall Retrieval in Mountainous Areas. J. Appl. Meteor. Climatol., 51, 405-425, doi: 10.1175/JAMC-D-10-05024.1.

pyart.retrieve.**map_profile_to_gates**(*profile*, *heights*, *radar*, *toa=None*, *profile_field=None*, *height_field=None*)

Given a profile of a variable map it to the gates of radar assuming 4/3Re.

Parameters

profile [array] Profile array to map.

heights [array] Monotonically increasing heights in meters with same shape as profile.

radar [Radar] Radar to map to.

toa [float, optional] Top of atmosphere, where to use profile up to. If None check for mask and use lowest element, if no mask uses whole profile.

height_field [str, optional] Name to use for height field metadata. None will use the default field name from the Py-ART configuration file.

profile_field [str, optional] Name to use for interpolate profile field metadata. None will use the default field name from the Py-ART configuration file.

Returns

height_dict, profile_dict [dict] Field dictionaries containing the height of the gates and the profile interpolated onto the radar gates.

pyart.retrieve.**melting_layer_giangrande**(*radar*, *nVol=3*, *maxh=6000.0*, *hres=50.0*, *rmin=1000.0*, *elmin=4.0*, *elmax=10.0*, *rhomin=0.75*, *rhomax=0.94*, *zhmin=20.0*, *hwindow=500.0*, *mlzhmin=30.0*, *mlzhmax=50.0*, *mlzdrmin=1.0*, *mlzdrmax=5.0*, *htol=500.0*, *ml_bottom_diff_max=1000.0*, *time_accu_max=1800.0*, *nml_points_min=None*, *wlength=20.0*, *percentile_bottom=0.3*, *percentile_top=0.9*, *interpol=True*, *time_nodata_allowed=3600.0*, *refl_field=None*, *zdr_field=None*, *rhv_field=None*, *temp_field=None*, *iso0_field=None*, *ml_field=None*, *ml_pos_field=None*, *temp_ref=None*, *get_iso0=False*, *ml_global=None*)

Detects the melting layer following the approach by Giangrande et al (2008)

Parameters

radar [radar] radar object

Returns

ml_obj [radar-like object] A radar-like object containing the field melting layer height with the bottom (at range position 0) and top (at range position one) of the melting layer at each ray

ml_dict [dict] A dictionary containg the position of the range gate respect to the melting layer and metadata

iso0_dict [dict or None] A dictionary containing the distance respect to the melting layer and metadata

ml_global [dict or None] stack of previous volume data to introduce some time dependency. Its max size is controlled by the nVol parameter. It is always in (pseudo-)RHI mode.

Other Parameters

nVol [int] Number of volume scans to aggregate

maxh [float] Maximum possible height of the melting layer [m MSL]

hres [float] Step of the height of the melting layer [m]

rmin [float] Minimum range from radar where to look for melting layer contaminated range gates [m]

elmin, elmax [float] Minimum and maximum elevation angles where to look for melting layer contaminated range gates [degree]

rhomin, rhomax [float] min and max rhohv to consider pixel potential melting layer pixel

zhmin [float] Minimum reflectivity level of a range gate to consider it a potential melting layer gate [dBZ]

hwindow [float] Maximum distance (in range) from potential melting layer gate where to look for a maximum [m]

mlzhmin, mlzhmax [float] Minimum and maximum values that a peak in reflectivity within the melting layer may have to consider the range gate melting layer contaminated [dBZ]

mlzdrmin, mlzdrmax [float] Minimum and maximum values that a peak in differential reflectivity within the melting layer may have to consider the range gate melting layer contaminated [dB]

htol [float] maximum distance from the iso0 coming from model allowed to consider the range gate melting layer contaminated [m]

ml_bottom_dif_max [float] Maximum distance from the bottom of the melting layer computed in the previous time step to consider a range gate melting layer contaminated [m]

time_accu_max [float] Maximum time allowed to accumulate data from consecutive scans [s]

nml_points_min [int] minimum number of melting layer points to consider valid melting layer detection

wlength [float] length of the window to select the azimuth angles used to compute the melting layer limits at a particular azimuth [degree]

percentile_bottom, percentile_top [float [0,1]] percentile of ml points above which is considered that the bottom of the melting layer starts and the top ends

interpol [bool] Whether to interpolate the obtained results in order to get a value for each azimuth

time_nodata_allowed [float] The maximum time allowed for no data before considering the melting layer not valid [s]

refl_field, zdr_field, rhv_field, temp_field, iso0_field [str] Inputs. Field names within the radar object which represent the horizonal reflectivity, the differential reflectivity, the copolar correlation coefficient, the temperature and the height respect to the iso0 fields. A value of None for any of these parameters will use the default field name as defined in the Py-ART configuration file.

ml_field [str] Output. Field name which represents the melting layer field. A value of None will use the default field name as defined in the Py-ART configuration file.

ml_pos_field [str] Output. Field name which represents the melting layer top and bottom height field. A value of None will use the default field name as defined in the Py-ART configuration file.

temp_ref [str] the field use as reference for temperature. Can be temperature or height_over_iso0. If None, it excludes model data from the algorithm.

get_iso0 [bool] returns height w.r.t. freezing level top for each gate in the radar volume.

ml_global : stack of previous volume data to introduce some time dependency. Its max size is controlled by the nVol parameter. It is always in (pseudo-)RHI mode.

References

Giangrande, S.E., Krause, J.M., Ryzhkov, A.V.: Automatic Designation of the Melting Layer with a Polarimetric Prototype of the WSR-88D Radar, J. of Applied Meteo. and Clim., 47, 1354-1364, doi:10.1175/2007JAMC1634.1, 2008

pyart.retrieve.**melting_layer_hydroclass**(*radar*, *hydro_field=None*, *ml_field=None*, *ml_pos_field=None*, *iso0_field=None*, *force_continuity=True*, *dist_max=350.0*, *get_iso0=False*)

Using the results of the hydrometeor classification by Besic et al. estimates the position of the range gates respect to the melting layer, the melting layer top and bottom height and the distance of the range gate with respect to the freezing level.

Parameters

radar [Radar] Radar object. Must have and hydrometeor classification field

hydro_field [str] Name of the hydrometeor classification field. A value of None will use the default field name as defined in the Py-ART configuration file.

ml_field, ml_pos_field, iso0_field [str] Name of the melting layer, melting layer heightand iso0 field. A value of None for any of these parameters will use the default field names as defined in the Py-ART configuration file.

force_continuity [Bool] If True, the melting layer is forced to be continuous in range

dist_max [float] The maximum distance between range gates flagged as inside the melting layer to consider them as gates in the melting layer.

Returns

ml_obj [radar-like object] A radar-like object containing the field melting layer height with the bottom (at range position 0) and top (at range position one) of the melting layer at each ray

ml_dict [dict] A dictionary containg the position of the range gate respect to the melting layer and metadata

iso0_dict [dict or None] A dictionary containing the distance respect to the melting layer and metadata

pyart.retrieve.**quasi_vertical_profile**(*radar*, *desired_angle=None*, *fields=None*, *gatefilter=None*)

Quasi Vertical Profile.

Creates a QVP object containing fields from a radar object that can be used to plot and produce the quasi vertical profile

Parameters

radar [Radar] Radar object used.

field [string] Radar field to use for QVP calculation.

desired_angle [float] Radar tilt angle to use for indexing radar field data. None will result in wanted_angle = 20.0

Returns

qvp [Dictonary] A quasi vertical profile object containing fields from a radar object

Other Parameters

gatefilter [GateFilter] A GateFilter indicating radar gates that should be excluded from the import qvp calculation

References

Troemel, S., M. Kumjian, A. Ryzhkov, and C. Simmer, 2013: Backscatter differential phase - estimation and variability. J Appl. Meteor. Clim.. 52, 2529 - 2548.

Troemel, S., A. Ryzhkov, P. Zhang, and C. Simmer, 2014: Investigations of backscatter differential phase in the melting layer. J. Appl. Meteorol. Clim. 54, 2344 - 2359.

Ryzhkov, A., P. Zhang, H. Reeves, M. Kumjian, T. Tschallener, S. Tromel, C. Simmer, 2015: Quasi-vertical profiles - a new way to look at polarimetric radar data. Submitted to J. Atmos. Oceanic Technol.

pyart.retrieve.**steiner_conv_strat**(*grid*, *dx=None*, *dy=None*, *intense=42.0*, *work_level=3000.0*, *peak_relation='default'*, *area_relation='medium'*, *bkg_rad=11000.0*, *use_intense=True*, *fill_value=None*, *refl_field=None*)

Partition reflectivity into convective-stratiform using the Steiner et al. (1995) algorithm.

Parameters

grid [Grid] Grid containing reflectivity field to partition.

dx, dy [float, optional] The x- and y-dimension resolutions in meters, respectively. If None the resolution is determined from the first two axes values.

intense [float, optional] The intensity value in dBZ. Grid points with a reflectivity value greater or equal to the intensity are automatically flagged as convective. See reference for more information.

work_level [float, optional] The working level (separation altitude) in meters. This is the height at which the partitioning will be done, and should minimize bright band contamination. See reference for more information.

peak_relation ['default' or 'sgp', optional] The peakedness relation. See reference for more information.

area_relation ['small', 'medium', 'large', or 'sgp', optional] The convective area relation. See reference for more information.

bkg_rad [float, optional] The background radius in meters. See reference for more information.

use_intense [bool, optional] True to use the intensity criteria.

fill_value [float, optional] Missing value used to signify bad data points. A value of None will use the default fill value as defined in the Py-ART configuration file.

refl_field [str, optional] Field in grid to use as the reflectivity during partitioning. None will use the default reflectivity field name from the Py-ART configuration file.

Returns

eclass [dict] Steiner convective-stratiform classification dictionary.

References

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.

pyart.retrieve.**velocity_azimuth_display**(*radar*, *vel_field=None*, *z_want=None*, *gatefilter=None*)

Velocity azimuth display.

Creates a VAD object containing U Wind, V Wind and height that can then be used to plot and produce the velocity azimuth display.

Parameters

radar [Radar] Radar object used.

vel_field [string, optional] Velocity field to use for VAD calculation.

z_want [array, optional] Heights for where to sample vads from. None will result in np.linespace(0, 10000, 100).

gatefilter [GateFilter, optional] A GateFilter indicating radar gates that should be excluded from the import vad calculation.

Returns

vad: HorizontalWindProfile A velocity azimuth display object containing height, speed, direction, u_wind, v_wind from a radar object.

CHAPTER

EIGHT

MAPPING (PYART.MAP)

Py-ART has a robust function for mapping radar data from the collected radar coordinates to Cartesian coordinates.

grid_from_radars(radars, grid_shape, grid_limits)	Map one or more radars to a Cartesian grid returning a Grid object.
map_to_grid(radars, grid_shape, grid_limits)	Map one or more radars to a Cartesian grid.
map_gates_to_grid(radars, grid_shape, ... [, ...])	Map gates from one or more radars to a Cartesian grid.
example_roi_func_constant(zg, yg, xg)	Example RoI function which returns a constant radius.
example_roi_func_dist(zg, yg, xg)	Example RoI function which returns a radius which grows with distance.
example_roi_func_dist_beam(zg, yg, xg)	Example RoI function which returns a radius which grows with distance and whose parameters are based on virtual beam size.

pyart.map.**example_roi_func_constant**(*zg*, *yg*, *xg*)

Example RoI function which returns a constant radius.

Parameters

zg, yg, xg [float] Distance from the grid center in meters for the x, y and z axes.

Returns

roi [float] Radius of influence in meters

pyart.map.**example_roi_func_dist**(*zg*, *yg*, *xg*)

Example RoI function which returns a radius which grows with distance.

Parameters

zg, yg, xg [float] Distance from the grid center in meters for the x, y and z axes.

Returns

roi [float]

pyart.map.**example_roi_func_dist_beam**(*zg*, *yg*, *xg*)

Example RoI function which returns a radius which grows with distance and whose parameters are based on virtual beam size.

Parameters

zg, yg, xg [float] Distance from the grid center in meters for the x, y and z axes.

Returns

roi [float]

pyart.map.**get_earth_radius**(*latitude*)

Computes the earth radius for a given latitude

Parameters

latitude: latitude in degrees (WGS84)

Returns

earth_radius [the radius of the earth at the given latitude]

pyart.map.**grid_from_radars**(*radars*, *grid_shape*, *grid_limits*, *gridding_algo='map_gates_to_grid'*, ***kwargs*)

Map one or more radars to a Cartesian grid returning a Grid object.

Additional arguments are passed to *map_to_grid()* or *map_gates_to_grid()*.

Parameters

radars [Radar or tuple of Radar objects.] Radar objects which will be mapped to the Cartesian grid.

grid_shape [3-tuple of floats] Number of points in the grid (z, y, x).

grid_limits [3-tuple of 2-tuples] Minimum and maximum grid location (inclusive) in meters for the z, y, x coordinates.

gridding_algo ['map_to_grid' or 'map_gates_to_grid'] Algorithm to use for gridding. 'map_to_grid' finds all gates within a radius of influence for each grid point, 'map_gates_to_grid' maps each radar gate onto the grid using a radius of influence and is typically significantly faster.

Returns

grid [Grid] A `pyart.io.Grid` object containing the gridded radar data.

See also:

map_to_grid Map to grid and return a dictionary of radar fields.

map_gates_to_grid Map each gate onto a grid returning a dictionary of radar fields.

References

Barnes S., 1964: A Technique for Maximizing Details in Numerical Weather Map Analysis. Journal of Applied Meteorology and Climatology, 3(4), 396-409.

Cressman G., 1959: An operational objective analysis system. Monthly Weather Review, 87(10), 367-374.

Pauley, P. M. and X. Wu, 1990: The theoretical, discrete, and actual response of the Barnes objective analysis scheme for one- and two-dimensional fields. Monthly Weather Review, 118, 1145-1164

pyart.map.**map_gates_to_grid**(*radars*, *grid_shape*, *grid_limits*, *grid_origin=None*, *grid_origin_alt=None*, *grid_projection=None*, *fields=None*, *gatefilters=False*, *map_roi=True*, *weighting_function='Barnes'*, *toa=17000.0*, *roi_func='dist_beam'*, *constant_roi=None*, *z_factor=0.05*, *xy_factor=0.02*, *min_radius=500.0*, *h_factor=1.0*, *nb=1.5*, *bsp=1.0*, ***kwargs*)

Map gates from one or more radars to a Cartesian grid.

Generate a Cartesian grid of points for the requested fields from the collected points from one or more radars. For each radar gate that is not filtered a radius of influence is calculated. The weighted field values for that gate

are added to all grid points within that radius. This routine scaled linearly with the number of radar gates and the effective grid size.

Parameters not defined below are identical to those in *map_to_grid()*.

Parameters

roi_func [str or RoIFunction] Radius of influence function. A function which takes an z, y, x grid location, in meters, and returns a radius (in meters) within which all collected points will be included in the weighting for that grid points. Examples can be found in the Typically following strings can use to specify a built in radius of influence function:

- constant: constant radius of influence.
- dist: radius grows with the distance from each radar.
- dist_beam: radius grows with the distance from each radar and parameter are based of virtual beam sizes.

A custom RoIFunction can be defined using the RoIFunction class and defining a get_roi method which returns the radius. For efficient mapping this class should be implemented in Cython.

Returns

grids [dict] Dictionary of mapped fields. The keys of the dictionary are given by parameter fields. Each elements is a *grid_size* float64 array containing the interpolated grid for that field.

See also:

grid_from_radars Map to a grid and return a Grid object

map_to_grid Create grid by finding the radius of influence around each grid point.

pyart.map.**map_to_grid**(*radars*, *grid_shape*, *grid_limits*, *grid_origin=None*, *grid_origin_alt=None*, *grid_projection=None*, *fields=None*, *gatefilters=False*, *map_roi=True*, *weighting_function='Barnes'*, *toa=17000.0*, *copy_field_data=True*, *algorithm='kd_tree'*, *leafsize=10.0*, *roi_func='dist_beam'*, *constant_roi=None*, *z_factor=0.05*, *xy_factor=0.02*, *min_radius=500.0*, *h_factor=1.0*, *nb=1.5*, *bsp=1.0*, ***kwargs*)

Map one or more radars to a Cartesian grid.

Generate a Cartesian grid of points for the requested fields from the collected points from one or more radars. The field value for a grid point is found by interpolating from the collected points within a given radius of influence and weighting these nearby points according to their distance from the grid points. Collected points are filtered according to a number of criteria so that undesired points are not included in the interpolation.

Parameters

radars [Radar or tuple of Radar objects.] Radar objects which will be mapped to the Cartesian grid.

grid_shape [3-tuple of floats] Number of points in the grid (z, y, x).

grid_limits [3-tuple of 2-tuples] Minimum and maximum grid location (inclusive) in meters for the z, y, x coordinates.

grid_origin [(float, float) or None] Latitude and longitude of grid origin. None sets the origin to the location of the first radar.

grid_origin_alt: float or None Altitude of grid origin, in meters. None sets the origin to the location of the first radar.

grid_projection [dic or str] Projection parameters defining the map projection used to transform the locations of the radar gates in geographic coordinate to Cartesian coodinates. None will use the default dictionary which uses a native azimutal equidistance projection. See *pyart.core.Grid()* for additional details on this parameter. The geographic coordinates of the radar gates are calculated using the projection defined for each radar. No transformation is used if a grid_origin and grid_origin_alt are None and a single radar is specified.

fields [list or None] List of fields within the radar objects which will be mapped to the cartesian grid. None, the default, will map the fields which are present in all the radar objects.

gatefilters [GateFilter, tuple of GateFilter objects, optional] Specify what gates from each radar will be included in the interpolation onto the grid. Only gates specified in each gatefilters will be included in the mapping to the grid. A single GateFilter can be used if a single Radar is being mapped. A value of False for a specific element or the entire parameter will apply no filtering of gates for a specific radar or all radars (the default). Similarily a value of None will create a GateFilter from the radar moments using any additional arguments by passing them to `moment_based_gate_filter()`.

roi_func [str or function] Radius of influence function. A functions which takes an z, y, x grid location, in meters, and returns a radius (in meters) within which all collected points will be included in the weighting for that grid points. Examples can be found in the *example_roi_func_constant()*, *example_roi_func_dist()*, and *example_roi_func_dist_beam()*. Alternatively the following strings can use to specify a built in radius of influence function:

- constant: constant radius of influence.
- dist: radius grows with the distance from each radar.
- dist_beam: radius grows with the distance from each radar and parameter are based of virtual beam sizes.

The parameters which control these functions are listed in the *Other Parameters* section below.

map_roi [bool] True to include a radius of influence field in the returned dictionary under the 'ROI' key. This is the value of roi_func at all grid points.

weighting_function ['Barnes' or 'Barnes2' or 'Cressman' or 'Nearest'] Functions used to weight nearby collected points when interpolating a grid point.

toa [float] Top of atmosphere in meters. Collected points above this height are not included in the interpolation.

Returns

grids [dict] Dictionary of mapped fields. The keys of the dictionary are given by parameter fields. Each elements is a *grid_size* float64 array containing the interpolated grid for that field.

Other Parameters

constant_roi [float] Radius of influence parameter for the built in 'constant' function. This parameter is the constant radius in meter for all grid points. This parameter is used when *roi_func* is *constant* or constant_roi is not None. If constant_roi is not None, the constant roi_func is used automatically.

z_factor, xy_factor, min_radius [float] Radius of influence parameters for the built in 'dist' function. The parameter correspond to the radius size increase, in meters, per meter increase in the z-dimension from the nearest radar, the same foreach meteter in the xy-distance from

the nearest radar, and the minimum radius of influence in meters. These parameters are only used when *roi_func* is 'dist'.

h_factor, nb, bsp, min_radius [float] Radius of influence parameters for the built in 'dist_beam' function. The parameter correspond to the height scaling, virtual beam width, virtual beam spacing, and minimum radius of influence. These parameters are only used when *roi_func* is 'dist_mean'.

copy_field_data [bool] True to copy the data within the radar fields for faster gridding, the dtype for all fields in the grid will be float64. False will not copy the data which preserves the dtype of the fields in the grid, may use less memory but results in significantly slower gridding times. When False gates which are masked in a particular field but are not masked in the *refl_field* field will still be included in the interpolation. This can be prevented by setting this parameter to True or by gridding each field individually setting the *refl_field* parameter and the *fields* parameter to the field in question. It is recommended to set this parameter to True.

algorithm ['kd_tree'.] Algorithms to use for finding the nearest neighbors. 'kd_tree' is the only valid option.

leafsize [int] Leaf size passed to the neighbor lookup tree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem. This value should only effect the speed of the gridding, not the results.

See also:

grid_from_radars Map to grid and return a Grid object.

pyart.map.**polar_to_cartesian**(*radar_sweep*, *field_name*, *cart_res=75*, *max_range=None*, *mapping=None*)

Interpolates a PPI or RHI scan in polar coordinates to a regular cartesian grid of South-North and West-East coordinates (for PPI) or distance at ground and altitude coordinates (for RHI)

Parameters

radar [Radar] Radar instance as generated py pyart

sweep [int] Sweep number to project to cartesian coordinates.

field_name [str] Name of the radar field to be interpolated

cart_res [int, optional] Resolution (in m.) of the cartesian grid to which polar data is interpolated

max_range [int, optional] Maximal allowed range (in m.) from radar for gates to be interpolated

mapping [dict, optional] Dictionnary of mapping indexes (from polar to cartesian), gets returned by the function (see below). Can be used as input when interpolating sequentially several variables for the same scan, to save significant time

Returns

coords [tuple of 2 arrays] 2D coordinates of the cartesian grid

cart_data [2D array] Interpolated radar measurements (on the cartesian grid)

mapping,: dict Dictionnary of mapping indexes (from polar to cartesian),which contains the indexes mapping the polar grid to the cartesian grid as well as some metadata.

CHAPTER

NINE

GRAPHING (PYART.GRAPH)

Creating plots of Radar and Grid fields.

9.1 Plotting radar data

RadarDisplay(radar[, shift])	A display object for creating plots from data in a radar object.
RadarMapDisplay(radar[, shift, grid_projection])	A display object for creating plots on a geographic map from data in a Radar object.
AirborneRadarDisplay(radar[, shift])	A display object for creating plots from data in a airborne radar object.
RadarMapDisplayBasemap(radar[, shift])	A display object for creating plots on a geographic map from data in a Radar object.

9.2 Plotting grid data

GridMapDisplay(grid[, debug])	A class for creating plots from a grid object using xarray with a cartopy projection.
GridMapDisplayBasemap(grid[, debug])	A class for creating plots from a grid object on top of a Basemap.

class pyart.graph.**AirborneRadarDisplay**(*radar*, *shift=(0.0, 0.0)*)

Bases: pyart.graph.radardisplay.RadarDisplay

A display object for creating plots from data in a airborne radar object.

Parameters

radar [Radar] Radar object to use for creating plots, should be an airborne radar.

shift [(float, float)] Shifts in km to offset the calculated x and y locations.

Attributes

plots [list] List of plots created.

plot_vars [list] List of fields plotted, order matches plot list.

cbs [list] List of colorbars created.

origin [str] 'Origin' or 'Radar'.

shift [(float, float)] Shift in meters.

loc [(float, float)] Latitude and Longitude of radar in degrees.

fields [dict] Radar fields.

scan_type [str] Scan type.

ranges [array] Gate ranges in meters.

azimuths [array] Azimuth angle in degrees.

elevations [array] Elevations in degrees.

fixed_angle [array] Scan angle in degrees.

rotation [array] Rotation angle in degrees.

roll [array] Roll angle in degrees.

drift [array] Drift angle in degrees.

tilt [array] Tilt angle in degrees.

heading [array] Heading angle in degrees.

pitch [array] Pitch angle in degrees.

altitude [array] Altitude angle in meters.

Methods

generate_az_rhi_title(self, field, azimuth)	Generate a title for a ray plot.
generate_filename(self, field, sweep[, ext, ...])	Generate a filename for a plot.
generate_ray_title(self, field, ray)	Generate a title for a ray plot.
generate_title(self, field, sweep[, ...])	Generate a title for a plot.
generate_vpt_title(self, field)	Generate a title for a VPT plot.
label_xaxis_r(self[, ax])	Label the xaxis with the default label for r units.
label_xaxis_rays([ax])	Label the yaxis with the default label for rays.
label_xaxis_time([ax])	Label the yaxis with the default label for rays.
label_xaxis_x(self[, ax])	Label the xaxis with the default label for x units.
label_yaxis_field(self, field[, ax])	Label the yaxis with the default label for a field units.
label_yaxis_y(self[, ax])	Label the yaxis with the default label for y units.
label_yaxis_z(self[, ax])	Label the yaxis with the default label for z units.
plot(self, field[, sweep])	Create a plot appropiate for the radar.
plot_azimuth_to_rhi(self, field, target_azimuth)	Plot pseudo-RHI scan by extracting the vertical field associated with the given azimuth.
plot_colorbar(self[, mappable, field, ...])	Plot a colorbar.
plot_cross_hair(size[, npts, ax])	Plot a cross-hair on a ppi plot.
plot_grid_lines([ax, col, ls])	Plot grid lines.
plot_label(self, label, location[, symbol, ...])	Plot a single symbol and label at a given location.
plot_labels(self, labels, locations[, ...])	Plot symbols and labels at given locations.
plot_ppi(self, field[, sweep, mask_tuple, ...])	Plot a PPI.
plot_range_ring(range_ring_location_km[, ...])	Plot a single range ring.
plot_range_rings(self, range_rings[, ax, ...])	Plot a series of range rings.
plot_ray(self, field, ray[, format_str, ...])	Plot a single ray.

Continued on next page

Table 3 – continued from previous page

plot_rhi(self, field[, sweep, mask_tuple, ...])	Plot a RHI.
plot_sweep_grid(self, field[, sweep, ...])	Plot a sweep as a grid.
plot_vpt(self, field[, mask_tuple, vmin, ...])	Plot a VPT scan.
set_aspect_ratio([aspect_ratio, ax])	Set the aspect ratio for plot area.
set_limits([xlim, ylim, ax])	Set the display limits.

__class__
alias of builtins.type

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.graph.radardisplay_airborne', '__doc__'

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *radar*, *shift=(0.0, 0.0)*)
Initialize the object.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, */*)
Return self<=value.

__lt__(*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.graph.radardisplay_airborne'

__ne__(*self*, *value*, */*)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, */*)
Helper for pickle.

__reduce_ex__(*self*, *protocol*, */*)
Helper for pickle.

__repr__ (*self*, */*)
Return repr(self).

__setattr__ (*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__sizeof__ (*self*, */*)
Size of object in memory, in bytes.

__str__ (*self*, */*)
Return str(self).

__subclasshook__ ()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_get_azimuth_rhi_data_x_y_z (*self*, *field*, *target_azimuth*, *edges*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return pseudo-RHI data from a plot function.

_get_colorbar_label (*self*, *field*)
Return a colorbar label for a given field.

_get_data (*self*, *field*, *sweep*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return data from a plot function.

_get_ray_data (*self*, *field*, *ray*, *mask_tuple*, *gatefilter*)
Retrieve and return ray data from a plot function.

_get_vpt_data (*self*, *field*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return vpt data from a plot function.

_get_x_y (*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x and y coordinate in km.

_get_x_y_z (*self*, *sweep*, *edges*, *filter_transitions*, *ignoreTilt=False*)
Retrieve and return x, y, and z coordinate in km.

_get_x_z (*self*, *sweep*, *edges*, *filter_transitions*, *ignoreTilt=False*)
Retrieve and return x and z coordinate in km.

_label_axes_ppi (*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a PPI plot.

_label_axes_ray (*self*, *axis_labels*, *field*, *ax*)
Set the x and y axis labels for a ray plot.

_label_axes_rhi (*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a RHI plot.

_label_axes_vpt (*self*, *axis_labels*, *time_axis_flag*, *ax*)
Set the x and y axis labels for a PPI plot.

_set_az_rhi_title (*self*, *field*, *azimuth*, *title*, *ax*)
Set the figure title for a ray plot using a default title.

_set_ray_title (*self*, *field*, *ray*, *title*, *ax*)
Set the figure title for a ray plot using a default title.

_set_title(*self*, *field*, *sweep*, *title*, *ax*, *datetime_format=None*, *use_sweep_time=True*)
Set the figure title using a default title.

static _set_vpt_time_axis(*ax*, *date_time_form=None*, *tz=None*)
Set the x axis as a time formatted axis.

Parameters

ax [Matplotlib axis instance] Axis to plot. None will use the current axis.

date_time_form [str] Format of the time string for x-axis labels.

tz [str] Time zone info to use when creating axis labels (see datetime).

_set_vpt_title(*self*, *field*, *title*, *ax*)
Set the figure title using a default title.

generate_az_rhi_title(*self*, *field*, *azimuth*)
Generate a title for a ray plot.

Parameters

field [str] Field plotted.

azimuth [float] Azimuth plotted.

Returns

title [str] Plot title.

generate_filename(*self*, *field*, *sweep*, *ext='png'*, *datetime_format='%Y%m%d%H%M%S'*, *use_sweep_time=False*)
Generate a filename for a plot.

Generated filename has form: radar_name_field_sweep_time.ext

Parameters

field [str] Field plotted.

sweep [int] Sweep plotted.

ext [str] Filename extension.

datetime_format [str] Format of datetime (using strftime format).

use_sweep_time [bool] If true, the current sweep's beginning time is used.

Returns

filename [str] Filename suitable for saving a plot.

generate_ray_title(*self*, *field*, *ray*)
Generate a title for a ray plot.

Parameters

field [str] Field plotted.

ray [int] Ray plotted.

Returns

title [str] Plot title.

generate_title(*self*, *field*, *sweep*, *datetime_format=None*, *use_sweep_time=True*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

sweep [int] Sweep plotted.

datetime_format [str] Format of datetime (using strftime format).

use_sweep_time [bool] If true, the current sweep's beginning time is used.

Returns

title [str] Plot title.

generate_vpt_title(*self*, *field*)
Generate a title for a VPT plot.

Parameters

field [str] Field plotted.

Returns

title [str] Plot title.

label_xaxis_r(*self*, *ax=None*)
Label the xaxis with the default label for r units.

static label_xaxis_rays(*ax=None*)
Label the yaxis with the default label for rays.

static label_xaxis_time(*ax=None*)
Label the yaxis with the default label for rays.

label_xaxis_x(*self*, *ax=None*)
Label the xaxis with the default label for x units.

label_yaxis_field(*self*, *field*, *ax=None*)
Label the yaxis with the default label for a field units.

label_yaxis_y(*self*, *ax=None*)
Label the yaxis with the default label for y units.

label_yaxis_z(*self*, *ax=None*)
Label the yaxis with the default label for z units.

plot(*self*, *field*, *sweep=0*, ***kwargs*)
Create a plot appropiate for the radar.

This function calls the plotting function corresponding to the scan_type of the radar. Additional keywords can be passed to customize the plot, see the appropiate plot function for the allowed keywords.

Parameters

field [str] Field to plot.

sweep [int] Sweep number to plot, not used for VPT scans.

See also:

plot_ppi Plot a PPI scan

plot_sweep_grid Plot a RHI or VPT scan

plot_azimuth_to_rhi(*self*, *field*, *target_azimuth*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *reverse_xaxis=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, ***kwargs*)

Plot pseudo-RHI scan by extracting the vertical field associated with the given azimuth.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

target_azimuth [integer] Azimuthal angle in degrees where cross section will be taken.

Other Parameters

mask_tuple [(str, float)] 2-Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

reverse_xaxis [bool or None] True to reverse the x-axis so the plot reads east to west, False to have east to west. None (the default) will reverse the axis only when all the distances are negative.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to True to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_colorbar(*self*, *mappable=None*, *field=None*, *label=None*, *orient='vertical'*, *cax=None*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*)

Plot a colorbar.

Parameters

mappable [Image, ContourSet, etc.] Image, ContourSet, etc to which the colorbar applied. If None the last mappable object will be used.

field [str] Field to label colorbar with.

label [str] Colorbar label. None will use a default value from the last field plotted.

orient [str] Colorbar orientation, either 'vertical' [default] or 'horizontal'.

cax [Axis] Axis onto which the colorbar will be drawn. None is also valid.

ax [Axes] Axis onto which the colorbar will be drawn. None is also valid.

fig [Figure] Figure to place colorbar on. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

static plot_cross_hair(*size*, *npts=100*, *ax=None*)

Plot a cross-hair on a ppi plot.

Parameters

size [float] Size of cross-hair in km.

npts: int Number of points in the cross-hair, higher for better resolution.

ax [Axis] Axis to plot on. None will use the current axis.

static plot_grid_lines(*ax=None*, *col='k'*, *ls=':'*)

Plot grid lines.

Parameters

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for grid lines.

ls [str] Linestyle to use for grid lines.

plot_label(*self*, *label*, *location*, *symbol='r+'*, *text_color='k'*, *ax=None*)

Plot a single symbol and label at a given location.

Transforms of the symbol location in latitude and longitude units to x and y plot units is performed using an azimuthal equidistance map projection centered at the radar.

Parameters

label [str] Label text to place just above symbol.

location [2-tuples] Tuple of latitude, longitude (in degrees) at which the symbol will be place. The label is placed just above the symbol.

symbol [str] Matplotlib color+marker strings defining the symbol to place at the given location.

text_color [str] Matplotlib color defining the color of the label text.

ax [Axis] Axis to plot on. None will use the current axis.

plot_labels(*self*, *labels*, *locations*, *symbols='r+'*, *text_color='k'*, *ax=None*)

Plot symbols and labels at given locations.

Parameters

labels [list of str] List of labels to place just above symbols.

locations [list of 2-tuples] List of latitude, longitude (in degrees) tuples at which symbols will be place. Labels are placed just above the symbols.

symbols [list of str or str] List of matplotlib color+marker strings defining symbols to place at given locations. If a single string is provided, that symbol will be placed at all locations.

text_color [str] Matplotlib color defining the color of the label text.

ax [Axis] Axis to plot on. None will use the current axis.

plot_ppi(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, *title_datetime_format=None*, *title_use_sweep_time=True*, ***kwargs*)

Plot a PPI.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int, optional] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_datetime_format [str] Format of datetime in the title (using strftime format).

title_use_sweep_time [bool] True for the current sweep's beginning time to be used for the title. False for the radar's beginning time.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

static plot_range_ring(*range_ring_location_km*, *npts=100*, *ax=None*, *col='k'*, *ls='-'*, *lw=2*)

Plot a single range ring.

Parameters

range_ring_location_km [float] Location of range ring in km.

npts: int Number of points in the ring, higher for better resolution.

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for range rings.

ls [str] Linestyle to use for range rings.

plot_range_rings(*self*, *range_rings*, *ax=None*, *col='k'*, *ls='-'*, *lw=2*)
Plot a series of range rings.

Parameters

range_rings [list] List of locations in km to draw range rings.

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for range rings.

ls [str] Linestyle to use for range rings.

plot_ray(*self*, *field*, *ray*, *format_str='k-'*, *mask_tuple=None*, *ray_min=None*, *ray_max=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *gatefilter=None*, *axislabels_flag=True*, *ax=None*, *fig=None*)
Plot a single ray.

Parameters

field [str] Field to plot.

ray [int] Ray number to plot.

Other Parameters

format_str [str] Format string defining the line style and marker.

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

ray_min [float] Minimum ray value, None for default value, ignored if mask_outside is False.

ray_max [float] Maximum ray value, None for default value, ignored if mask_outside is False.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and ray parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

plot_rhi(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *reverse_xaxis=None*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, *title_datetime_format=None*, *title_use_sweep_time=True*, ***kwargs*)
Plot a RHI.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int,] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] 2-Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_datetime_format [str] Format of datetime in the title (using strftime format).

title_use_sweep_time [bool] True for the current sweep's beginning time to be used for the title. False for the radar's beginning time.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

reverse_xaxis [bool or None] True to reverse the x-axis so the plot reads west to east, False to have east to west. None (the default) will reverse the axis only when all the distances are negative. (i.e) axis will be absolute distance without taking into consideration the orientation

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_sweep_grid(*self*, *field*, *sweep=0*, *ignoreTilt=False*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *cmap=None*, *norm=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *filter_transitions=True*, *ax=None*, *fig=None*, *gatefilter=None*, *raster=False*, *ticks=None*, *ticklabs=None*, ***kwargs*)

Plot a sweep as a grid.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int, optional] Sweep number to plot.

Other Parameters

ignoreTilt [bool] True to ignore tilt angle when running the antenna_to_cartesian_track_relative coordinate transformation (by setting tilt angle to 0), effectively plotting data relative to slant range (the same plotting method utilized by the NCAR soloii/3 software). False (default) plots relative to the aircraft longitudinal axis.

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

plot_vpt(*self*, *field*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *time_axis_flag=False*, *date_time_form=None*, *tz=None*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, ***kwargs*)

Plot a VPT scan.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

time_axis_flag [bool] True to plot the x-axis as time. False uses the index number. Default is False - index-based.

date_time_form [str, optional] Format of the time string for x-axis labels. Parameter is ignored if time_axis_flag is set to False.

tz [str, optional] Time zone info to use when creating axis labels (see datetime). Parameter is ignored if time_axis_flag is set to False.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

static set_aspect_ratio(*aspect_ratio=0.75*, *ax=None*)
Set the aspect ratio for plot area.

static set_limits(*xlim=None*, *ylim=None*, *ax=None*)
Set the display limits.

Parameters

xlim [tuple, optional] 2-Tuple containing y-axis limits in km. None uses default limits.

ylim [tuple, optional] 2-Tuple containing x-axis limits in km. None uses default limits.

ax [Axis] Axis to adjust. None will adjust the current axis.

class pyart.graph.**GridMapDisplay**(*grid*, *debug=False*)

Bases: object

A class for creating plots from a grid object using xarray with a cartopy projection.

Parameters

grid [Grid] Grid with data which will be used to create plots.

debug [bool] True to print debugging messages, False to supress them.

Attributes

grid [Grid] Grid object.

debug [bool] True to print debugging messages, False to supress them.

Methods

cartopy_coastlines(self)	Get coastlines using cartopy.
cartopy_political_boundaries(self)	Get political boundaries using cartopy.
cartopy_states(self)	Get state boundaries using cartopy.
generate_filename(self, field, level[, ext])	Generate a filename for a grid plot.
generate_grid_title(self, field, level)	Generate a title for a plot.
generate_latitudinal_level_title(self, ...)	Generate a title for a plot.
generate_longitudinal_level_title(self, ...)	Generate a title for a plot.
get_dataset(self)	Creating an xarray dataset from a radar object.
plot_colorbar(self[, mappable, orientation, ...])	Plot a colorbar.
plot_crosshairs(self[, lon, lat, linestyle, ...])	Plot crosshairs at a given longitude and latitude.
plot_grid(self, field[, level, vmin, vmax, ...])	Plot the grid using xarray and cartopy.
plot_grid_contour(self, field[, level, ...])	Plot the grid contour using xarray and cartopy.
plot_latitude_slice(self, field[, lon, lat])	Plot a slice along a given latitude.
plot_latitudinal_level(self, field, y_index)	Plot a slice along a given latitude.
plot_latlon_level(self, field, ind_1, ind_2)	Plot a slice along two points given by its lat, lon Additional arguments are passed to Basemaps's pcolormesh function.
plot_latlon_slice(self, field[, coord1, coord2])	Plot a slice along a given longitude.
plot_longitude_slice(self, field[, lon, lat])	Plot a slice along a given longitude.
plot_longitudinal_level(self, field, x_index)	Plot a slice along a given longitude.

__class__

alias of builtins.type

__delattr__(*self*, *name*, */*)

Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.graph.gridmapdisplay', '__doc__': '\n A

__dir__(*self*, /)
Default dir() implementation.

__eq__(*self*, *value*, /)
Return self==value.

__format__(*self*, *format_spec*, /)
Default object formatter.

__ge__(*self*, *value*, /)
Return self>=value.

__getattribute__(*self*, *name*, /)
Return getattr(self, name).

__gt__(*self*, *value*, /)
Return self>value.

__hash__(*self*, /)
Return hash(self).

__init__(*self*, *grid*, *debug=False*)
initalize the object.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, /)
Return self<=value.

__lt__(*self*, *value*, /)
Return self<value.

__module__ = 'pyart.graph.gridmapdisplay'

__ne__(*self*, *value*, /)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, /)
Helper for pickle.

__reduce_ex__(*self*, *protocol*, /)
Helper for pickle.

__repr__(*self*, /)
Return repr(self).

__setattr__(*self*, *name*, *value*, /)
Implement setattr(self, name, value).

__sizeof__(*self*, /)
Size of object in memory, in bytes.

__str__(*self*, /)
Return str(self).

__subclasshook__()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_find_nearest_grid_indices(*self*, *lon*, *lat*)
Find the nearest x, y grid indices for a given latitude and longitude.

_get_label_x(*self*)
Get default label for x units.

_get_label_y(*self*)
Get default label for y units.

_get_label_z(*self*)
Get default label for z units.

_label_axes_grid(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a grid plot.

_label_axes_latitude(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a latitude slice.

_label_axes_latlon(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a lat-lon slice.

_label_axes_longitude(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a longitude slice.

cartopy_coastlines(*self*)
Get coastlines using cartopy.

cartopy_political_boundaries(*self*)
Get political boundaries using cartopy.

cartopy_states(*self*)
Get state boundaries using cartopy.

generate_filename(*self*, *field*, *level*, *ext='png'*)
Generate a filename for a grid plot.

Generated filename has form: grid_name_field_level_time.ext

Parameters

field [str] Field plotted.

level [int] Level plotted.

ext [str] Filename extension.

Returns

filename [str] Filename suitable for saving a plot.

generate_grid_title(*self*, *field*, *level*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

level [int] Vertical level plotted.

Returns

title [str] Plot title.

generate_latitudinal_level_title(*self*, *field*, *level*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

level [int] Latitudinal level plotted.

Returns

title [str] Plot title.

generate_longitudinal_level_title(*self*, *field*, *level*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

level [int] Longitudinal level plotted.

Returns

title [str] Plot title.

get_dataset(*self*)
Creating an xarray dataset from a radar object. This function has been removed from Py-ART ARM-DOE

plot_colorbar(*self*, *mappable=None*, *orientation='horizontal'*, *label=None*, *cax=None*, *ax=None*, *fig=None*, *field=None*, *ticks=None*, *ticklabs=None*)
Plot a colorbar.

Parameters

mappable [Image, ContourSet, etc.] Image, ContourSet, etc to which the colorbar applied. If None the last mappable object will be used.

field [str] Field to label colorbar with.

label [str] Colorbar label. None will use a default value from the last field plotted.

orient [str] Colorbar orientation, either 'vertical' [default] or 'horizontal'.

cax [Axis] Axis onto which the colorbar will be drawn. None is also valid.

ax [Axes] Axis onto which the colorbar will be drawn. None is also valid.

fig [Figure] Figure to place colorbar on. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

plot_crosshairs(*self*, *lon=None*, *lat=None*, *linestyle='–'*, *color='r'*, *linewidth=2*, *ax=None*)
Plot crosshairs at a given longitude and latitude.

Parameters

lon, lat [float] Longitude and latitude (in degrees) where the crosshairs should be placed. If None the center of the grid is used.

linestyle [str] Matplotlib string describing the line style.

color [str] Matplotlib string for color of the line.

linewidth [float] Width of markers in points.

ax [axes or None] Axis to add the crosshairs to, if None the current axis is used.

plot_grid(*self, field, level=0, vmin=None, vmax=None, norm=None, cmap=None, mask_outside=False, title=None, title_flag=True, axislabels=(None, None), axislabels_flag=False, colorbar_flag=True, colorbar_label=None, colorbar_orient='vertical', ax=None, fig=None, lat_lines=None, lon_lines=None, projection=None, embellish=True, maps_list=['countries', 'coastlines'], resolution='110m', alpha=None, background_zoom=8, ticks=None, ticklabs=None, imshow=False, **kwargs*)

Plot the grid using xarray and cartopy.

Additional arguments are passed to Xarray's pcolormesh function.

Parameters

field [str] Field to be plotted.

level [int] Index corresponding to the height level to be plotted.

Other Parameters

vmin, vmax [float] Lower and upper range for the colormesh. If either parameter is None, a value will be determined from the field attributes (if available) or the default values of -8, 64 will be used. Parameters are used for luminance scaling.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None will use the default generated from the field and level parameters. Parameter is ignored if the title_flag is False.

title_flag [bool] True to add title to plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

lat_lines, lon_lines [array or None] Location at which to draw latitude and longitude lines. None will use default values which are reasonable for maps of North America.

projection [cartopy.crs class] Map projection supported by cartopy. Used for all subsequent calls to the GeoAxes object generated. Defaults to PlateCarree.

embelish [bool] True by default. Set to False to supress drawinf of coastlines etc... Use for speedup when specifying shapefiles. Note that lat lon labels only work with certain projections.

maps_list: list of strings if embelish is true the list of maps to use. default countries, coastlines

resolution ['10m', '50m', '110m'.] Resolution of NaturalEarthFeatures to use. See Cartopy documentation for details.

alpha [float or None] Set the alpha tranparency of the grid plot. Useful for overplotting radar over other datasets.

background_zoom [int] Zoom of the background image. A highest number provides more detail at the cost of processing speed

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

imshow [bool] If used, plot uses ax.imshow instead of ax.pcolormesh. Default is False.

plot_grid_contour(*self, field, level=0, vmin=None, vmax=None, mask_outside=False, title=None, title_flag=True, ax=None, fig=None, lat_lines=None, lon_lines=None, projection=None, contour_values=None, linewidths=1.5, embelish=True, maps_list=['countries', 'coastlines'], resolution='110m', background_zoom=8, **kwargs*)

Plot the grid contour using xarray and cartopy.

Additional arguments are passed to Xarray's pcolormesh function.

Parameters

field [str] Field to be plotted.

level [int] Index corresponding to the height level to be plotted.

Other Parameters

vmin, vmax [float] Lower and upper range for the colormesh. If either parameter is None, a value will be determined from the field attributes (if available) or the default values of -8, 64 will be used. Parameters are used for luminance scaling.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None will use the default generated from the field and level parameters. Parameter is ignored if the title_flag is False.

title_flag [bool] True to add title to plot, False does not add a title.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

lat_lines, lon_lines [array or None] Location at which to draw latitude and longitude lines. None will use default values which are resonable for maps of North America.

projection [cartopy.crs class] Map projection supported by cartopy. Used for all subsequent calls to the GeoAxes object generated. Defaults to PlateCarree.

contour_values [float array]

list of contours to plot

linewidths [float] width of the contour lines

embelish [bool] True by default. Set to False to supress drawinf of coastlines etc... Use for speedup when specifying shapefiles. Note that lat lon labels only work with certain projections.

maps_list: list of strings if embelish is true the list of maps to use. default countries, coastlines

resolution ['10m', '50m', '110m'.] Resolution of NaturalEarthFeatures to use. See Cartopy documentation for details.

background_zoom [int] Zoom of the background image. A highest number provides more detail at the cost of processing speed

plot_latitude_slice(*self*, *field*, *lon=None*, *lat=None*, ***kwargs*)

Plot a slice along a given latitude.

For documentation of additional arguments see *plot_latitudinal_level()*.

Parameters

field [str] Field to be plotted.

lon, lat [float] Longitude and latitude (in degrees) specifying the slice. If None the center of the grid is used.

plot_latitudinal_level(*self*, *field*, *y_index*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None, None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, ***kwargs*)

Plot a slice along a given latitude.

Additional arguments are passed to Basemaps's pcolormesh function.

Parameters

field [str] Field to be plotted.

y_index [float] Index of the latitudinal level to plot.

vmin, vmax [float] Lower and upper range for the colormesh. If either parameter is None, a value will be determined from the field attributes (if available) or the default values of -8, 64 will be used. Parameters are ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and lat,lon parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

plot_latlon_level(*self*, *field*, *ind_1*, *ind_2*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, ***kwargs*)

Plot a slice along two points given by its lat, lon Additional arguments are passed to Basemaps's pcolormesh function. Parameters ———- field : str

Field to be plotted.

ind_1, ind_2 [float] x,y indices of the two points crossed by the slice.

vmin, vmax [float] Lower and upper range for the colormesh. If either parameter is None, a value will be determined from the field attributes (if available) or the default values of -8, 64 will be used. Parameters are ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and lat,lon parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

plot_latlon_slice(*self*, *field*, *coord1=None*, *coord2=None*, ***kwargs*)
Plot a slice along a given longitude. For documentation of additional arguments see *plot_longitudinal_level()*. Parameters ————- field : str

Field to be plotted.

coord1, coord2 [tupple of floats] tupple of floats containing the longitude and latitude (in degrees) specifying the two points crossed by the slice. If none two extremes of the grid is used

plot_longitude_slice(*self*, *field*, *lon=None*, *lat=None*, ***kwargs*)
Plot a slice along a given longitude.

For documentation of additional arguments see *plot_longitudinal_level()*.

Parameters

field [str] Field to be plotted.

lon, lat [float] Longitude and latitude (in degrees) specifying the slice. If None the center of the grid is used.

plot_longitudinal_level(*self*, *field*, *x_index*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, ***kwargs*)
Plot a slice along a given longitude.

Additional arguments are passed to Basemaps's pcolormesh function.

Parameters

field [str] Field to be plotted.

x_index [float] Index of the longitudinal level to plot.

vmin, vmax [float] Lower and upper range for the colormesh. If either parameter is None, a value will be determined from the field attributes (if available) or the default values of -8, 64 will be used. Parameters are ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and lat,lon parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

class pyart.graph.**GridMapDisplayBasemap**(*grid*, *debug=False*)

Bases: object

A class for creating plots from a grid object on top of a Basemap.

Parameters

grid [Grid] Grid with data which will be used to create plots.

debug [bool] True to print debugging messages, False to supress them.

Attributes

grid [Grid] Grid object.

debug [bool] True to print debugging messages, False to supressed them.

basemap [Basemap] Last plotted basemap, None when no basemap has been plotted.

mappables [list] List of ContourSet, etc. which have been plotted, useful when adding colorbars.

fields [list] List of fields which have been plotted.

Methods

generate_filename(self, field, level[, ext])	Generate a filename for a grid plot.
generate_grid_title(self, field, level)	Generate a title for a plot.
generate_latitudinal_level_title(self, ...)	Generate a title for a plot.
generate_longitudinal_level_title(self, ...)	Generate a title for a plot.
get_basemap(self)	get basemap of the plot
plot_basemap(self[, lat_lines, lon_lines, ...])	Plot a basemap.

Continued on next page

Table 5 – continued from previous page

plot_colorbar(self[, mappable, orientation, ...])	Plot a colorbar.
plot_crosshairs(self[, lon, lat, ...])	Plot crosshairs at a given longitude and latitude.
plot_grid(self, field[, level, vmin, vmax, ...])	Plot the grid onto the current basemap.
plot_latitude_slice(self, field[, lon, lat])	Plot a slice along a given latitude.
plot_latitudinal_level(self, field, y_index)	Plot a slice along a given latitude.
plot_longitude_slice(self, field[, lon, lat])	Plot a slice along a given longitude.
plot_longitudinal_level(self, field, x_index)	Plot a slice along a given longitude.

__class__
alias of builtins.type

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.graph.gridmapdisplay_basemap', '__doc__

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *grid*, *debug=False*)
initalize the object.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, */*)
Return self<=value.

__lt__(*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.graph.gridmapdisplay_basemap'

__ne__(*self*, *value*, */*)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, /)
Helper for pickle.

__reduce_ex__(*self*, *protocol*, /)
Helper for pickle.

__repr__(*self*, /)
Return repr(self).

__setattr__(*self*, *name*, *value*, /)
Implement setattr(self, name, value).

__sizeof__(*self*, /)
Size of object in memory, in bytes.

__str__(*self*, /)
Return str(self).

__subclasshook__()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_find_nearest_grid_indices(*self*, *lon*, *lat*)
Find the nearest x, y grid indices for a given latitude and longitude.

_get_label_x(*self*)
Get default label for x units.

_get_label_y(*self*)
Get default label for y units.

_get_label_z(*self*)
Get default label for z units.

_label_axes_grid(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a grid plot.

_label_axes_latitude(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a latitude slice.

_label_axes_longitude(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a longitude slice.

_make_basemap(*self*, *resolution='l'*, *area_thresh=10000*, *auto_range=True*, *min_lon=-92*, *max_lon=-86*, *min_lat=40*, *max_lat=44*, *ax=None*, ***kwargs*)
Make a basemap.

Parameters

auto_range [bool] True to determine map ranges from the latitude and longitude limits of the grid. False will use the min_lon, max_lon, min_lat, and max_lat parameters for the map range.

min_lat, max_lat, min_lon, max_lon [float] Latitude and longitude ranges for the map projection region in degrees. These parameter are not used if auto_range is True.

resolution ['c', 'l', 'i', 'h', or 'f'.] Resolution of boundary database to use. See Basemap documentation for details.

area_thresh [int] Basemap area_thresh parameter. See Basemap documentation.

ax [axes or None.] Axis to add the basemap to, if None the current axis is used.

kwargs [Basemap options] Options to be passed to Basemap. If projection is not specified here it uses proj='merc' (mercator).

generate_filename(*self*, *field*, *level*, *ext='png'*)
Generate a filename for a grid plot.

Generated filename has form: grid_name_field_level_time.ext

Parameters

field [str] Field plotted.

level [int] Level plotted.

ext [str] Filename extension.

Returns

filename [str] Filename suitable for saving a plot.

generate_grid_title(*self*, *field*, *level*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

level [int] Verical level plotted.

Returns

title [str] Plot title.

generate_latitudinal_level_title(*self*, *field*, *level*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

level [int] Longitudinal level plotted.

Returns

title [str] Plot title.

generate_longitudinal_level_title(*self*, *field*, *level*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

level [int] Longitudinal level plotted.

Returns

title [str] Plot title.

get_basemap(*self*)
get basemap of the plot

plot_basemap(*self*, *lat_lines=None*, *lon_lines=None*, *resolution='l'*, *area_thresh=10000*, *auto_range=True*, *min_lon=-92*, *max_lon=-86*, *min_lat=40*, *max_lat=44*, *ax=None*, ***kwargs*)

Plot a basemap.

Parameters

lat_lines, lon_lines [array or None] Locations at which to draw latitude and longitude lines. None will use default values which are reasonable for maps of North America.

auto_range [bool] True to determine map ranges from the latitude and longitude limits of the grid. False will use the min_lon, max_lon, min_lat, and max_lat parameters for the map range.

min_lat, max_lat, min_lon, max_lon [float] Latitude and longitude ranges for the map projection region in degrees. These parameter are not used if auto_range is True.

resolution ['c', 'l', 'i', 'h', or 'f'.] Resolution of boundary database to use. See Basemap documentation for details.

area_thresh [int] Basemap area_thresh parameter. See Basemap documentation.

ax [axes or None.] Axis to add the basemap to, if None the current axis is used.

kwargs: Basemap options Options to be passed to Basemap. If projection is not specified here it uses proj='merc' (mercator).

plot_colorbar(*self*, *mappable=None*, *orientation='horizontal'*, *label=None*, *cax=None*, *ax=None*, *fig=None*, *field=None*, *ticks=None*, *ticklabs=None*)

Plot a colorbar.

Parameters

mappable [Image, ContourSet, etc.] Image, ContourSet, etc to which the colorbar applied. If None the last mappable object will be used.

field [str] Field to label colorbar with.

label [str] Colorbar label. None will use a default value from the last field plotted.

orient [str] Colorbar orientation, either 'vertical' [default] or 'horizontal'.

cax [Axis] Axis onto which the colorbar will be drawn. None is also valid.

ax [Axes] Axis onto which the colorbar will be drawn. None is also valid.

fig [Figure] Figure to place colorbar on. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

plot_crosshairs(*self*, *lon=None*, *lat=None*, *line_style='r–'*, *linewidth=2*, *ax=None*)

Plot crosshairs at a given longitude and latitude.

Parameters

lon, lat [float] Longitude and latitude (in degrees) where the crosshairs should be placed. If None the center of the grid is used.

line_style [str] Matplotlib string describing the line style.

linewidth [float] Width of markers in points.

ax [axes or None.] Axis to add the crosshairs to, if None the current axis is used.

plot_grid(*self*, *field*, *level=0*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=False*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, ***kwargs*)

Plot the grid onto the current basemap.

Additional arguments are passed to Basemaps's pcolormesh function.

Parameters

field [str] Field to be plotted.

level [int] Index corresponding to the height level to be plotted.

vmin, vmax [float] Lower and upper range for the colormesh. If either parameter is None, a value will be determined from the field attributes (if available) or the default values of -8, 64 will be used. Parameters are ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and level parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

plot_latitude_slice(*self*, *field*, *lon=None*, *lat=None*, ***kwargs*)

Plot a slice along a given latitude.

For documentation of additional arguments see *plot_latitudinal_level()*.

Parameters

field [str] Field to be plotted.

lon, lat [float] Longitude and latitude (in degrees) specifying the slice. If None the center of the grid is used.

plot_latitudinal_level(*self*, *field*, *y_index*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, ***kwargs*)

Plot a slice along a given latitude.

Additional arguments are passed to Basemaps's pcolormesh function.

Parameters

field [str] Field to be plotted.

y_index [float] Index of the latitudinal level to plot.

vmin, vmax [float] Lower and upper range for the colormesh. If either parameter is None, a value will be determined from the field attributes (if available) or the default values of -8, 64 will be used. Parameters are ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and lat,lon parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

plot_longitude_slice(*self*, *field*, *lon=None*, *lat=None*, ***kwargs*)
Plot a slice along a given longitude.

For documentation of additional arguments see *plot_longitudinal_level()*.

Parameters

field [str] Field to be plotted.

lon, lat [float] Longitude and latitude (in degrees) specifying the slice. If None the center of the grid is used.

plot_longitudinal_level(*self*, *field*, *x_index*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, ***kwargs*)
Plot a slice along a given longitude.

Additional arguments are passed to Basemaps's pcolormesh function.

Parameters

field [str] Field to be plotted.

x_index [float] Index of the longitudinal level to plot.

vmin, vmax [float] Lower and upper range for the colormesh. If either parameter is None, a value will be determined from the field attributes (if available) or the default values of -8, 64 will be used. Parameters are ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and lat,lon parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

class pyart.graph.**RadarDisplay**(*radar*, *shift=(0.0, 0.0)*)

Bases: object

A display object for creating plots from data in a radar object.

Parameters

radar [Radar] Radar object to use for creating plots.

shift [(float, float)] Shifts in km to offset the calculated x and y locations.

Attributes

plots [list] List of plots created.

plot_vars [list] List of fields plotted, order matches plot list.

cbs [list] List of colorbars created.

origin [str] 'Origin' or 'Radar'.

shift [(float, float)] Shift in meters.

loc [(float, float)] Latitude and Longitude of radar in degrees.

fields [dict] Radar fields.

scan_type [str] Scan type.

ranges [array] Gate ranges in meters.

azimuths [array] Azimuth angle in degrees.

elevations [array] Elevations in degrees.

fixed_angle [array] Scan angle in degrees.

antenna_transition [array or None] Antenna transition flag (1 in transition, 0 in transition) or None if no antenna transition.

Methods

generate_az_rhi_title(self, field, azimuth)	Generate a title for a ray plot.
generate_filename(self, field, sweep[, ext, ...])	Generate a filename for a plot.
generate_ray_title(self, field, ray)	Generate a title for a ray plot.
generate_title(self, field, sweep[, ...])	Generate a title for a plot.
generate_vpt_title(self, field)	Generate a title for a VPT plot.
label_xaxis_r(self[, ax])	Label the xaxis with the default label for r units.
label_xaxis_rays([ax])	Label the yaxis with the default label for rays.
label_xaxis_time([ax])	Label the yaxis with the default label for rays.
label_xaxis_x(self[, ax])	Label the xaxis with the default label for x units.
label_yaxis_field(self, field[, ax])	Label the yaxis with the default label for a field units.
label_yaxis_y(self[, ax])	Label the yaxis with the default label for y units.
label_yaxis_z(self[, ax])	Label the yaxis with the default label for z units.
plot(self, field[, sweep])	Create a plot appropiate for the radar.

Continued on next page

Table 6 – continued from previous page

plot_azimuth_to_rhi(self, field, target_azimuth)	Plot pseudo-RHI scan by extracting the vertical field associated with the given azimuth.
plot_colorbar(self[, mappable, field, ...])	Plot a colorbar.
plot_cross_hair(size[, npts, ax])	Plot a cross-hair on a ppi plot.
plot_grid_lines([ax, col, ls])	Plot grid lines.
plot_label(self, label, location[, symbol, ...])	Plot a single symbol and label at a given location.
plot_labels(self, labels, locations[, ...])	Plot symbols and labels at given locations.
plot_ppi(self, field[, sweep, mask_tuple, ...])	Plot a PPI.
plot_range_ring(range_ring_location_km[, ...])	Plot a single range ring.
plot_range_rings(self, range_rings[, ax, ...])	Plot a series of range rings.
plot_ray(self, field, ray[, format_str, ...])	Plot a single ray.
plot_rhi(self, field[, sweep, mask_tuple, ...])	Plot a RHI.
plot_vpt(self, field[, mask_tuple, vmin, ...])	Plot a VPT scan.
set_aspect_ratio([aspect_ratio, ax])	Set the aspect ratio for plot area.
set_limits([xlim, ylim, ax])	Set the display limits.

__class__
alias of builtins.type

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.graph.radardisplay', '__doc__': "\n A

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *radar*, *shift=(0.0, 0.0)*)
Initialize the object.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, */*)
Return self<=value.

__lt__(*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.graph.radardisplay'

__ne__(*self*, *value*, */*)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, */*)
Helper for pickle.

__reduce_ex__(*self*, *protocol*, */*)
Helper for pickle.

__repr__(*self*, */*)
Return repr(self).

__setattr__(*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__sizeof__(*self*, */*)
Size of object in memory, in bytes.

__str__(*self*, */*)
Return str(self).

__subclasshook__()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_get_azimuth_rhi_data_x_y_z(*self*, *field*, *target_azimuth*, *edges*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return pseudo-RHI data from a plot function.

_get_colorbar_label(*self*, *field*)
Return a colorbar label for a given field.

_get_data(*self*, *field*, *sweep*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return data from a plot function.

_get_ray_data(*self*, *field*, *ray*, *mask_tuple*, *gatefilter*)
Retrieve and return ray data from a plot function.

_get_vpt_data(*self*, *field*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return vpt data from a plot function.

_get_x_y(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x and y coordinate in km.

_get_x_y_z(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x, y, and z coordinate in km.

_get_x_z(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x and z coordinate in km.

_label_axes_ppi(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a PPI plot.

_label_axes_ray(*self*, *axis_labels*, *field*, *ax*)
Set the x and y axis labels for a ray plot.

_label_axes_rhi(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a RHI plot.

_label_axes_vpt(*self*, *axis_labels*, *time_axis_flag*, *ax*)
Set the x and y axis labels for a PPI plot.

_set_az_rhi_title(*self*, *field*, *azimuth*, *title*, *ax*)
Set the figure title for a ray plot using a default title.

_set_ray_title(*self*, *field*, *ray*, *title*, *ax*)
Set the figure title for a ray plot using a default title.

_set_title(*self*, *field*, *sweep*, *title*, *ax*, *datetime_format=None*, *use_sweep_time=True*)
Set the figure title using a default title.

static _set_vpt_time_axis(*ax*, *date_time_form=None*, *tz=None*)
Set the x axis as a time formatted axis.

Parameters

ax [Matplotlib axis instance] Axis to plot. None will use the current axis.

date_time_form [str] Format of the time string for x-axis labels.

tz [str] Time zone info to use when creating axis labels (see datetime).

_set_vpt_title(*self*, *field*, *title*, *ax*)
Set the figure title using a default title.

generate_az_rhi_title(*self*, *field*, *azimuth*)
Generate a title for a ray plot.

Parameters

field [str] Field plotted.

azimuth [float] Azimuth plotted.

Returns

title [str] Plot title.

generate_filename(*self*, *field*, *sweep*, *ext='png'*, *datetime_format='%Y%m%d%H%M%S'*, *use_sweep_time=False*)
Generate a filename for a plot.

Generated filename has form: radar_name_field_sweep_time.ext

Parameters

field [str] Field plotted.

sweep [int] Sweep plotted.

ext [str] Filename extension.

datetime_format [str] Format of datetime (using strftime format).

use_sweep_time [bool] If true, the current sweep's beginning time is used.

Returns

filename [str] Filename suitable for saving a plot.

generate_ray_title(*self*, *field*, *ray*)
Generate a title for a ray plot.

Parameters

field [str] Field plotted.

ray [int] Ray plotted.

Returns

title [str] Plot title.

generate_title(*self*, *field*, *sweep*, *datetime_format=None*, *use_sweep_time=True*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

sweep [int] Sweep plotted.

datetime_format [str] Format of datetime (using strftime format).

use_sweep_time [bool] If true, the current sweep's beginning time is used.

Returns

title [str] Plot title.

generate_vpt_title(*self*, *field*)
Generate a title for a VPT plot.

Parameters

field [str] Field plotted.

Returns

title [str] Plot title.

label_xaxis_r(*self*, *ax=None*)
Label the xaxis with the default label for r units.

static label_xaxis_rays(*ax=None*)
Label the yaxis with the default label for rays.

static label_xaxis_time(*ax=None*)
Label the yaxis with the default label for rays.

label_xaxis_x(*self*, *ax=None*)
Label the xaxis with the default label for x units.

label_yaxis_field(*self*, *field*, *ax=None*)
Label the yaxis with the default label for a field units.

label_yaxis_y(*self*, *ax=None*)
Label the yaxis with the default label for y units.

label_yaxis_z(*self*, *ax=None*)
Label the yaxis with the default label for z units.

plot(*self*, *field*, *sweep=0*, ***kwargs*)
Create a plot appropiate for the radar.

This function calls the plotting function corresponding to the scan_type of the radar. Additional keywords can be passed to customize the plot, see the appropiate plot function for the allowed keywords.

Parameters

field [str] Field to plot.

sweep [int] Sweep number to plot, not used for VPT scans.

See also:

`plot_ppi` Plot a PPI scan

`plot_rhi` Plot a RHI scan

`plot_vpt` Plot a VPT scan

plot_azimuth_to_rhi(*self*, *field*, *target_azimuth*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *reverse_xaxis=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, ***kwargs*)

Plot pseudo-RHI scan by extracting the vertical field associated with the given azimuth.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

target_azimuth [integer] Azimuthal angle in degrees where cross section will be taken.

Other Parameters

mask_tuple [(str, float)] 2-Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

reverse_xaxis [bool or None] True to reverse the x-axis so the plot reads east to west, False to have east to west. None (the default) will reverse the axis only when all the distances are negative.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to True to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_colorbar(*self*, *mappable=None*, *field=None*, *label=None*, *orient='vertical'*, *cax=None*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*)

Plot a colorbar.

Parameters

mappable [Image, ContourSet, etc.] Image, ContourSet, etc to which the colorbar applied. If None the last mappable object will be used.

field [str] Field to label colorbar with.

label [str] Colorbar label. None will use a default value from the last field plotted.

orient [str] Colorbar orientation, either 'vertical' [default] or 'horizontal'.

cax [Axis] Axis onto which the colorbar will be drawn. None is also valid.

ax [Axes] Axis onto which the colorbar will be drawn. None is also valid.

fig [Figure] Figure to place colorbar on. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

static plot_cross_hair(*size*, *npts=100*, *ax=None*)

Plot a cross-hair on a ppi plot.

Parameters

size [float] Size of cross-hair in km.

npts: int Number of points in the cross-hair, higher for better resolution.

ax [Axis] Axis to plot on. None will use the current axis.

static plot_grid_lines(*ax=None*, *col='k'*, *ls=':'*)
Plot grid lines.

Parameters

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for grid lines.

ls [str] Linestyle to use for grid lines.

plot_label(*self*, *label*, *location*, *symbol='r+'*, *text_color='k'*, *ax=None*)
Plot a single symbol and label at a given location.

Transforms of the symbol location in latitude and longitude units to x and y plot units is performed using an azimuthal equidistance map projection centered at the radar.

Parameters

label [str] Label text to place just above symbol.

location [2-tuples] Tuple of latitude, longitude (in degrees) at which the symbol will be place. The label is placed just above the symbol.

symbol [str] Matplotlib color+marker strings defining the symbol to place at the given location.

text_color [str] Matplotlib color defining the color of the label text.

ax [Axis] Axis to plot on. None will use the current axis.

plot_labels(*self*, *labels*, *locations*, *symbols='r+'*, *text_color='k'*, *ax=None*)
Plot symbols and labels at given locations.

Parameters

labels [list of str] List of labels to place just above symbols.

locations [list of 2-tuples] List of latitude, longitude (in degrees) tuples at which symbols will be place. Labels are placed just above the symbols.

symbols [list of str or str] List of matplotlib color+marker strings defining symbols to place at given locations. If a single string is provided, that symbol will be placed at all locations.

text_color [str] Matplotlib color defining the color of the label text.

ax [Axis] Axis to plot on. None will use the current axis.

plot_ppi(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, *title_datetime_format=None*, *title_use_sweep_time=True*, ***kwargs*)
Plot a PPI.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int, optional] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_datetime_format [str] Format of datetime in the title (using strftime format).

title_use_sweep_time [bool] True for the current sweep's beginning time to be used for the title. False for the radar's beginning time.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure

to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

static plot_range_ring(*range_ring_location_km*, *npts=100*, *ax=None*, *col='k'*, *ls='-'*, *lw=2*)
Plot a single range ring.

Parameters

range_ring_location_km [float] Location of range ring in km.

npts: int Number of points in the ring, higher for better resolution.

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for range rings.

ls [str] Linestyle to use for range rings.

plot_range_rings(*self*, *range_rings*, *ax=None*, *col='k'*, *ls='-'*, *lw=2*)
Plot a series of range rings.

Parameters

range_rings [list] List of locations in km to draw range rings.

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for range rings.

ls [str] Linestyle to use for range rings.

plot_ray(*self*, *field*, *ray*, *format_str='k-'*, *mask_tuple=None*, *ray_min=None*, *ray_max=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *gatefilter=None*, *axislabels_flag=True*, *ax=None*, *fig=None*)
Plot a single ray.

Parameters

field [str] Field to plot.

ray [int] Ray number to plot.

Other Parameters

format_str [str] Format string defining the line style and marker.

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

ray_min [float] Minimum ray value, None for default value, ignored if mask_outside is False.

ray_max [float] Maximum ray value, None for default value, ignored if mask_outside is False.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and ray parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

plot_rhi(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *reverse_xaxis=None*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, *title_datetime_format=None*, *title_use_sweep_time=True*, ***kwargs*)

Plot a RHI.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int,] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] 2-Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_datetime_format [str] Format of datetime in the title (using strftime format).

title_use_sweep_time [bool] True for the current sweep's beginning time to be used for the title. False for the radar's beginning time.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

reverse_xaxis [bool or None] True to reverse the x-axis so the plot reads west to east, False to have east to west. None (the default) will reverse the axis only when all the distances are negative. (i.e) axis will be absolute distance without taking into consideration the orientation

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_vpt(*self*, *field*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *time_axis_flag=False*, *date_time_form=None*, *tz=None*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, ***kwargs*)

Plot a VPT scan.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

time_axis_flag [bool] True to plot the x-axis as time. False uses the index number. Default is False - index-based.

date_time_form [str, optional] Format of the time string for x-axis labels. Parameter is ignored if time_axis_flag is set to False.

tz [str, optional] Time zone info to use when creating axis labels (see datetime). Parameter is ignored if time_axis_flag is set to False.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

static set_aspect_ratio(*aspect_ratio=0.75*, *ax=None*)
Set the aspect ratio for plot area.

static set_limits(*xlim=None*, *ylim=None*, *ax=None*)
Set the display limits.

Parameters

xlim [tuple, optional] 2-Tuple containing y-axis limits in km. None uses default limits.

ylim [tuple, optional] 2-Tuple containing x-axis limits in km. None uses default limits.

ax [Axis] Axis to adjust. None will adjust the current axis.

class pyart.graph.**RadarMapDisplay**(*radar*, *shift=(0.0, 0.0)*, *grid_projection=None*)

Bases: pyart.graph.radardisplay.RadarDisplay

A display object for creating plots on a geographic map from data in a Radar object.

This class is still a work in progress. Some functionality may not work correctly. Please report any problems to the Py-ART GitHub Issue Tracker.

Parameters

radar [Radar] Radar object to use for creating plots.

shift [(float, float)] Shifts in km to offset the calculated x and y locations.

Attributes

plots [list] List of plots created.

plot_vars [list] List of fields plotted, order matches plot list.

cbs [list] List of colorbars created.

origin [str] 'Origin' or 'Radar'.

shift [(float, float)] Shift in meters.

loc [(float, float)] Latitude and Longitude of radar in degrees.

fields [dict] Radar fields.

scan_type [str] Scan type.

ranges [array] Gate ranges in meters.

azimuths [array] Azimuth angle in degrees.

elevations [array] Elevations in degrees.

fixed_angle [array] Scan angle in degrees.

grid_projection [cartopy.crs] AzimuthalEquidistant cartopy projection centered on radar. Used to transform points into map projection.

Methods

generate_az_rhi_title(self, field, azimuth)	Generate a title for a ray plot.
generate_filename(self, field, sweep[, ext, ...])	Generate a filename for a plot.
generate_ray_title(self, field, ray)	Generate a title for a ray plot.
generate_title(self, field, sweep[, ...])	Generate a title for a plot.
generate_vpt_title(self, field)	Generate a title for a VPT plot.
label_xaxis_r(self[, ax])	Label the xaxis with the default label for r units.
label_xaxis_rays([ax])	Label the yaxis with the default label for rays.
label_xaxis_time([ax])	Label the yaxis with the default label for rays.
label_xaxis_x(self[, ax])	Label the xaxis with the default label for x units.
label_yaxis_field(self, field[, ax])	Label the yaxis with the default label for a field units.
label_yaxis_y(self[, ax])	Label the yaxis with the default label for y units.
label_yaxis_z(self[, ax])	Label the yaxis with the default label for z units.
plot(self, field[, sweep])	Create a plot appropiate for the radar.

Continued on next page

Table 7 – continued from previous page

plot_azimuth_to_rhi(self, field, target_azimuth)	Plot pseudo-RHI scan by extracting the vertical field associated with the given azimuth.
plot_colorbar(self[, mappable, field, ...])	Plot a colorbar.
plot_cross_hair(size[, npts, ax])	Plot a cross-hair on a ppi plot.
plot_grid_lines([ax, col, ls])	Plot grid lines.
plot_label(self, label, location[, symbol, ...])	Plot a single symbol and label at a given location.
plot_labels(self, labels, locations[, ...])	Plot symbols and labels at given locations.
plot_line_geo(self, line_lons, line_lats[, ...])	Plot a line segments on the current map given values in lat and lon.
plot_line_xy(self, line_x, line_y[, line_style])	Plot a line segments on the current map given radar x, y values.
plot_point(self, lon, lat[, symbol, ...])	Plot a point on the current map.
plot_ppi(self, field[, sweep, mask_tuple, ...])	Plot a PPI.
plot_ppi_map(self, field[, sweep, ...])	Plot a PPI volume sweep onto a geographic map.
plot_range_ring(self, range_ring_location_km)	Plot a single range ring on the map.
plot_range_rings(self, range_rings[, ax, ...])	Plot a series of range rings.
plot_ray(self, field, ray[, format_str, ...])	Plot a single ray.
plot_rhi(self, field[, sweep, mask_tuple, ...])	Plot a RHI.
plot_vpt(self, field[, mask_tuple, vmin, ...])	Plot a VPT scan.
set_aspect_ratio([aspect_ratio, ax])	Set the aspect ratio for plot area.
set_limits([xlim, ylim, ax])	Set the display limits.

__class__
alias of builtins.type

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.graph.radarmapdisplay', '__doc__': "\n

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *radar*, *shift=(0.0, 0.0)*, *grid_projection=None*)
Initialize the object.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, /)
Return self<=value.

__lt__(*self*, *value*, /)
Return self<value.

__module__ = 'pyart.graph.radarmapdisplay'

__ne__(*self*, *value*, /)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, /)
Helper for pickle.

__reduce_ex__(*self*, *protocol*, /)
Helper for pickle.

__repr__(*self*, /)
Return repr(self).

__setattr__(*self*, *name*, *value*, /)
Implement setattr(self, name, value).

__sizeof__(*self*, /)
Size of object in memory, in bytes.

__str__(*self*, /)
Return str(self).

__subclasshook__()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_check_ax(*self*)
Check that a GeoAxes object exists, raise ValueError if not

_get_azimuth_rhi_data_x_y_z(*self*, *field*, *target_azimuth*, *edges*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return pseudo-RHI data from a plot function.

_get_colorbar_label(*self*, *field*)
Return a colorbar label for a given field.

_get_data(*self*, *field*, *sweep*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return data from a plot function.

_get_ray_data(*self*, *field*, *ray*, *mask_tuple*, *gatefilter*)
Retrieve and return ray data from a plot function.

_get_vpt_data(*self*, *field*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return vpt data from a plot function.

_get_x_y(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x and y coordinate in km.

_get_x_y_z(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x, y, and z coordinate in km.

_get_x_z(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x and z coordinate in km.

_label_axes_ppi(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a PPI plot.

_label_axes_ray(*self*, *axis_labels*, *field*, *ax*)
Set the x and y axis labels for a ray plot.

_label_axes_rhi(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a RHI plot.

_label_axes_vpt(*self*, *axis_labels*, *time_axis_flag*, *ax*)
Set the x and y axis labels for a PPI plot.

_set_az_rhi_title(*self*, *field*, *azimuth*, *title*, *ax*)
Set the figure title for a ray plot using a default title.

_set_ray_title(*self*, *field*, *ray*, *title*, *ax*)
Set the figure title for a ray plot using a default title.

_set_title(*self*, *field*, *sweep*, *title*, *ax*, *datetime_format=None*, *use_sweep_time=True*)
Set the figure title using a default title.

static _set_vpt_time_axis(*ax*, *date_time_form=None*, *tz=None*)
Set the x axis as a time formatted axis.

Parameters

ax [Matplotlib axis instance] Axis to plot. None will use the current axis.

date_time_form [str] Format of the time string for x-axis labels.

tz [str] Time zone info to use when creating axis labels (see datetime).

_set_vpt_title(*self*, *field*, *title*, *ax*)
Set the figure title using a default title.

generate_az_rhi_title(*self*, *field*, *azimuth*)
Generate a title for a ray plot.

Parameters

field [str] Field plotted.

azimuth [float] Azimuth plotted.

Returns

title [str] Plot title.

generate_filename(*self*, *field*, *sweep*, *ext='png'*, *datetime_format='%Y%m%d%H%M%S'*, *use_sweep_time=False*)
Generate a filename for a plot.

Generated filename has form: radar_name_field_sweep_time.ext

Parameters

field [str] Field plotted.

sweep [int] Sweep plotted.

ext [str] Filename extension.

datetime_format [str] Format of datetime (using strftime format).

use_sweep_time [bool] If true, the current sweep's beginning time is used.

Returns

filename [str] Filename suitable for saving a plot.

generate_ray_title(*self*, *field*, *ray*)
Generate a title for a ray plot.

Parameters

field [str] Field plotted.

ray [int] Ray plotted.

Returns

title [str] Plot title.

generate_title(*self*, *field*, *sweep*, *datetime_format=None*, *use_sweep_time=True*)
Generate a title for a plot.

Parameters

field [str] Field plotted.

sweep [int] Sweep plotted.

datetime_format [str] Format of datetime (using strftime format).

use_sweep_time [bool] If true, the current sweep's beginning time is used.

Returns

title [str] Plot title.

generate_vpt_title(*self*, *field*)
Generate a title for a VPT plot.

Parameters

field [str] Field plotted.

Returns

title [str] Plot title.

label_xaxis_r(*self*, *ax=None*)
Label the xaxis with the default label for r units.

static label_xaxis_rays(*ax=None*)
Label the yaxis with the default label for rays.

static label_xaxis_time(*ax=None*)
Label the yaxis with the default label for rays.

label_xaxis_x(*self*, *ax=None*)
Label the xaxis with the default label for x units.

label_yaxis_field(*self*, *field*, *ax=None*)
Label the yaxis with the default label for a field units.

label_yaxis_y(*self*, *ax=None*)
Label the yaxis with the default label for y units.

label_yaxis_z(*self*, *ax=None*)
Label the yaxis with the default label for z units.

plot(*self*, *field*, *sweep=0*, ***kwargs*)
Create a plot appropiate for the radar.

This function calls the plotting function corresponding to the scan_type of the radar. Additional keywords can be passed to customize the plot, see the appropiate plot function for the allowed keywords.

Parameters

field [str] Field to plot.

sweep [int] Sweep number to plot, not used for VPT scans.

See also:

plot_ppi Plot a PPI scan

plot_rhi Plot a RHI scan

plot_vpt Plot a VPT scan

plot_azimuth_to_rhi(*self*, *field*, *target_azimuth*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *reverse_xaxis=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, ***kwargs*)
Plot pseudo-RHI scan by extracting the vertical field associated with the given azimuth.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

target_azimuth [integer] Azimuthal angle in degrees where cross section will be taken.

Other Parameters

mask_tuple [(str, float)] 2-Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

reverse_xaxis [bool or None] True to reverse the x-axis so the plot reads east to west, False to have east to west. None (the default) will reverse the axis only when all the distances are negative.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to True to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_colorbar(*self*, *mappable=None*, *field=None*, *label=None*, *orient='vertical'*, *cax=None*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*)

Plot a colorbar.

Parameters

mappable [Image, ContourSet, etc.] Image, ContourSet, etc to which the colorbar applied. If None the last mappable object will be used.

field [str] Field to label colorbar with.

label [str] Colorbar label. None will use a default value from the last field plotted.

orient [str] Colorbar orientation, either 'vertical' [default] or 'horizontal'.

cax [Axis] Axis onto which the colorbar will be drawn. None is also valid.

ax [Axes] Axis onto which the colorbar will be drawn. None is also valid.

fig [Figure] Figure to place colorbar on. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

static plot_cross_hair(*size*, *npts=100*, *ax=None*)

Plot a cross-hair on a ppi plot.

Parameters

size [float] Size of cross-hair in km.

npts: int Number of points in the cross-hair, higher for better resolution.

ax [Axis] Axis to plot on. None will use the current axis.

static plot_grid_lines(*ax=None*, *col='k'*, *ls=':'*)
Plot grid lines.

Parameters

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for grid lines.

ls [str] Linestyle to use for grid lines.

plot_label(*self*, *label*, *location*, *symbol='r+'*, *text_color='k'*, *ax=None*)
Plot a single symbol and label at a given location.

Transforms of the symbol location in latitude and longitude units to x and y plot units is performed using an azimuthal equidistance map projection centered at the radar.

Parameters

label [str] Label text to place just above symbol.

location [2-tuples] Tuple of latitude, longitude (in degrees) at which the symbol will be place. The label is placed just above the symbol.

symbol [str] Matplotlib color+marker strings defining the symbol to place at the given location.

text_color [str] Matplotlib color defining the color of the label text.

ax [Axis] Axis to plot on. None will use the current axis.

plot_labels(*self*, *labels*, *locations*, *symbols='r+'*, *text_color='k'*, *ax=None*)
Plot symbols and labels at given locations.

Parameters

labels [list of str] List of labels to place just above symbols.

locations [list of 2-tuples] List of latitude, longitude (in degrees) tuples at which symbols will be place. Labels are placed just above the symbols.

symbols [list of str or str] List of matplotlib color+marker strings defining symbols to place at given locations. If a single string is provided, that symbol will be placed at all locations.

text_color [str] Matplotlib color defining the color of the label text.

ax [Axis] Axis to plot on. None will use the current axis.

plot_line_geo(*self*, *line_lons*, *line_lats*, *line_style='r-'*, ***kwargs*)
Plot a line segments on the current map given values in lat and lon.

Additional arguments are passed to ax.plot.

Parameters

line_lons [array] Longitude of line segment to plot.

line_lats [array] Latitude of line segment to plot.

line_style [str] Matplotlib compatible string which specifies the line style.

plot_line_xy(*self*, *line_x*, *line_y*, *line_style='r-'*, ***kwargs*)
Plot a line segments on the current map given radar x, y values.

Additional arguments are passed to ax.plot.

Parameters

line_x [array] X location of points to plot in meters from the radar.

line_y [array] Y location of points to plot in meters from the radar.

line_style [str, optional] Matplotlib compatible string which specifies the line style.

plot_point(*self*, *lon*, *lat*, *symbol='ro'*, *label_text=None*, *label_offset=(None*, *None)*, ***kwargs*)
Plot a point on the current map.

Additional arguments are passed to ax.plot.

Parameters

lon [float] Longitude of point to plot.

lat [float] Latitude of point to plot.

symbol [str] Matplotlib compatible string which specified the symbol of the point.

label_text [str, optional.] Text to label symbol with. If None no label will be added.

label_offset [[float, float]] Offset in lon, lat degrees for the bottom left corner of the label text relative to the point. A value of None will use 0.01.

plot_ppi(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, *title_datetime_format=None*, *title_use_sweep_time=True*, ***kwargs*)
Plot a PPI.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int, optional] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_datetime_format [str] Format of datetime in the title (using strftime format).

title_use_sweep_time [bool] True for the current sweep's beginning time to be used for the title. False for the radar's beginning time.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_ppi_map(*self, field, sweep=0, mask_tuple=None, vmin=None, vmax=None, cmap=None, norm=None, mask_outside=False, title=None, title_flag=True, colorbar_flag=True, colorbar_label=None, ax=None, fig=None, lat_lines=None, lon_lines=None, projection=None, min_lon=None, max_lon=None, min_lat=None, max_lat=None, width=None, height=None, lon_0=None, lat_0=None, resolution='110m', shapefile=None, shapefile_kwargs=None, edges=True, gatefilter=None, filter_transitions=True, embelish=True, maps_list=['countries', 'coastlines'], raster=False, ticks=None, ticklabs=None, alpha=None, background_zoom=8*)

Plot a PPI volume sweep onto a geographic map.

Parameters

field [str] Field to plot.

sweep [int, optional] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and tilt parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ax [Cartopy GeoAxes instance] If None, create GeoAxes instance using other keyword info. If provided, ax must have a Cartopy crs projection and projection kwarg below is ignored.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

lat_lines, lon_lines [array or None] Locations at which to draw latitude and longitude lines. None will use default values which are resonable for maps of North America.

projection [cartopy.crs class] Map projection supported by cartopy. Used for all subsequent calls to the GeoAxes object generated. Defaults to LambertConformal centered on radar.

min_lat, max_lat, min_lon, max_lon [float] Latitude and longitude ranges for the map projection region in degrees.

width, height [float] Width and height of map domain in meters. Only this set of parameters or the previous set of parameters (min_lat, max_lat, min_lon, max_lon) should be specified. If neither set is specified then the map domain will be determined from the extend of the radar gate locations.

shapefile [str] Filename for a shapefile to add to map.

shapefile_kwargs [dict] Key word arguments used to format shapefile. Projection defaults to lat lon (cartopy.crs.PlateCarree())

resolution ['10m', '50m', '110m'.] Resolution of NaturalEarthFeatures to use. See Cartopy documentation for details.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

embelish: bool True by default. Set to False to supress drawing of coastlines etc.. Use for speedup when specifying shapefiles. Note that lat lon labels only work with certain projections.

maps_list: list of strings if embelish is true the list of maps to use. default countries, coastlines

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

alpha [float or None] Set the alpha transparency of the radar plot. Useful for overplotting radar over other datasets.

background_zoom [int] Zoom of the background image. A highest number provides more detail at the cost of processing speed

plot_range_ring(*self*, *range_ring_location_km*, *npts=360*, *line_style='k-'*, ***kwargs*)
Plot a single range ring on the map.

Additional arguments are passed to ax.plot.

Parameters

range_ring_location_km [float] Location of range ring in km.

npts [int] Number of points in the ring, higher for better resolution.

line_style [str] Matplotlib compatible string which specified the line style of the ring.

plot_range_rings(*self*, *range_rings*, *ax=None*, *col='k'*, *ls='-'*, *lw=2*)
Plot a series of range rings.

Parameters

range_rings [list] List of locations in km to draw range rings.

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for range rings.

ls [str] Linestyle to use for range rings.

plot_ray(*self*, *field*, *ray*, *format_str='k-'*, *mask_tuple=None*, *ray_min=None*, *ray_max=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *gatefilter=None*, *axislabels_flag=True*, *ax=None*, *fig=None*)
Plot a single ray.

Parameters

field [str] Field to plot.

ray [int] Ray number to plot.

Other Parameters

format_str [str] Format string defining the line style and marker.

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

ray_min [float] Minimum ray value, None for default value, ignored if mask_outside is False.

ray_max [float] Maximum ray value, None for default value, ignored if mask_outside is False.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and ray parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

plot_rhi(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *reverse_xaxis=None*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, *title_datetime_format=None*, *title_use_sweep_time=True*, ***kwargs*)

Plot a RHI.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int,] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] 2-Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_datetime_format [str] Format of datetime in the title (using strftime format).

title_use_sweep_time [bool] True for the current sweep's beginning time to be used for the title. False for the radar's beginning time.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

reverse_xaxis [bool or None] True to reverse the x-axis so the plot reads west to east, False to have east to west. None (the default) will reverse the axis only when all the distances are negative. (i.e) axis will be absolute distance without taking into consideration the orientation

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_vpt(*self*, *field*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *time_axis_flag=False*, *date_time_form=None*, *tz=None*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, ***kwargs*)

Plot a VPT scan.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

time_axis_flag [bool] True to plot the x-axis as time. False uses the index number. Default is False - index-based.

date_time_form [str, optional] Format of the time string for x-axis labels. Parameter is ignored if time_axis_flag is set to False.

tz [str, optional] Time zone info to use when creating axis labels (see datetime). Parameter is ignored if time_axis_flag is set to False.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

static set_aspect_ratio(*aspect_ratio=0.75*, *ax=None*)
Set the aspect ratio for plot area.

static set_limits(*xlim=None*, *ylim=None*, *ax=None*)
Set the display limits.

Parameters

xlim [tuple, optional] 2-Tuple containing y-axis limits in km. None uses default limits.

ylim [tuple, optional] 2-Tuple containing x-axis limits in km. None uses default limits.

ax [Axis] Axis to adjust. None will adjust the current axis.

class pyart.graph.**RadarMapDisplayBasemap**(*radar*, *shift=(0.0, 0.0)*)
Bases: pyart.graph.radardisplay.RadarDisplay

A display object for creating plots on a geographic map from data in a Radar object.

This class is still a work in progress. Some functionality may not work correctly. Please report any problems to the Py-ART GitHub Issue Tracker.

Parameters

radar [Radar] Radar object to use for creating plots.

shift [(float, float)] Shifts in km to offset the calculated x and y locations.

Attributes

plots [list] List of plots created.

plot_vars [list] List of fields plotted, order matches plot list.

cbs [list] List of colorbars created.

origin [str] 'Origin' or 'Radar'.

shift [(float, float)] Shift in meters.

loc [(float, float)] Latitude and Longitude of radar in degrees.

fields [dict] Radar fields.

scan_type [str] Scan type.

ranges [array] Gate ranges in meters.

azimuths [array] Azimuth angle in degrees.

elevations [array] Elevations in degrees.

fixed_angle [array] Scan angle in degrees.

proj [Proj] Object for performing cartographic transformations specific to the geographic map plotted.

basemap [Basemap] Last plotted basemap, None when no basemap has been plotted.

Methods

generate_az_rhi_title(self, field, azimuth)	Generate a title for a ray plot.
generate_filename(self, field, sweep[, ext, ...])	Generate a filename for a plot.
generate_ray_title(self, field, ray)	Generate a title for a ray plot.
generate_title(self, field, sweep[, ...])	Generate a title for a plot.
generate_vpt_title(self, field)	Generate a title for a VPT plot.
label_xaxis_r(self[, ax])	Label the xaxis with the default label for r units.
label_xaxis_rays([ax])	Label the yaxis with the default label for rays.
label_xaxis_time([ax])	Label the yaxis with the default label for rays.
label_xaxis_x(self[, ax])	Label the xaxis with the default label for x units.
label_yaxis_field(self, field[, ax])	Label the yaxis with the default label for a field units.
label_yaxis_y(self[, ax])	Label the yaxis with the default label for y units.
label_yaxis_z(self[, ax])	Label the yaxis with the default label for z units.
plot(self, field[, sweep])	Create a plot appropiate for the radar.
plot_azimuth_to_rhi(self, field, target_azimuth)	Plot pseudo-RHI scan by extracting the vertical field associated with the given azimuth.
plot_colorbar(self[, mappable, field, ...])	Plot a colorbar.
plot_cross_hair(size[, npts, ax])	Plot a cross-hair on a ppi plot.
plot_grid_lines([ax, col, ls])	Plot grid lines.
plot_label(self, label, location[, symbol, ...])	Plot a single symbol and label at a given location.
plot_labels(self, labels, locations[, ...])	Plot symbols and labels at given locations.
plot_line_geo(self, line_lons, line_lats[, ...])	Plot a line segments on the current map given values in lat and lon.
plot_line_xy(self, line_x, line_y[, line_style])	Plot a line segments on the current map given radar x, y values.
plot_point(self, lon, lat[, symbol, ...])	Plot a point on the current map.
plot_ppi(self, field[, sweep, mask_tuple, ...])	Plot a PPI.
plot_ppi_map(self, field[, sweep, ...])	Plot a PPI volume sweep onto a geographic map.
plot_range_ring(self, range_ring_location_km)	Plot a single range ring on the map.
plot_range_rings(self, range_rings[, ax, ...])	Plot a series of range rings.
plot_ray(self, field, ray[, format_str, ...])	Plot a single ray.
plot_rhi(self, field[, sweep, mask_tuple, ...])	Plot a RHI.
plot_vpt(self, field[, mask_tuple, vmin, ...])	Plot a VPT scan.
set_aspect_ratio([aspect_ratio, ax])	Set the aspect ratio for plot area.
set_limits([xlim, ylim, ax])	Set the display limits.

__class__
alias of `builtins.type`

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.graph.radarmapdisplay_basemap', '__doc_

__dir__(*self*, */*)
Default dir() implementation.

__eq__(*self*, *value*, */*)
Return self==value.

__format__(*self*, *format_spec*, */*)

Default object formatter.

__ge__ (*self*, *value*, */*)
Return self>=value.

__getattribute__ (*self*, *name*, */*)
Return getattr(self, name).

__gt__ (*self*, *value*, */*)
Return self>value.

__hash__ (*self*, */*)
Return hash(self).

__init__ (*self*, *radar*, *shift=(0.0, 0.0)*)
Initialize the object.

__init_subclass__ ()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__ (*self*, *value*, */*)
Return self<=value.

__lt__ (*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.graph.radarmapdisplay_basemap'

__ne__ (*self*, *value*, */*)
Return self!=value.

__new__ (**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__ (*self*, */*)
Helper for pickle.

__reduce_ex__ (*self*, *protocol*, */*)
Helper for pickle.

__repr__ (*self*, */*)
Return repr(self).

__setattr__ (*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__sizeof__ (*self*, */*)
Size of object in memory, in bytes.

__str__ (*self*, */*)
Return str(self).

__subclasshook__ ()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

_check_basemap(*self*)
Check that basemap is not None, raise ValueError if it is.

_get_azimuth_rhi_data_x_y_z(*self*, *field*, *target_azimuth*, *edges*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return pseudo-RHI data from a plot function.

_get_colorbar_label(*self*, *field*)
Return a colorbar label for a given field.

_get_data(*self*, *field*, *sweep*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return data from a plot function.

_get_ray_data(*self*, *field*, *ray*, *mask_tuple*, *gatefilter*)
Retrieve and return ray data from a plot function.

_get_vpt_data(*self*, *field*, *mask_tuple*, *filter_transitions*, *gatefilter*)
Retrieve and return vpt data from a plot function.

_get_x_y(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x and y coordinate in km.

_get_x_y_z(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x, y, and z coordinate in km.

_get_x_z(*self*, *sweep*, *edges*, *filter_transitions*)
Retrieve and return x and z coordinate in km.

_label_axes_ppi(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a PPI plot.

_label_axes_ray(*self*, *axis_labels*, *field*, *ax*)
Set the x and y axis labels for a ray plot.

_label_axes_rhi(*self*, *axis_labels*, *ax*)
Set the x and y axis labels for a RHI plot.

_label_axes_vpt(*self*, *axis_labels*, *time_axis_flag*, *ax*)
Set the x and y axis labels for a PPI plot.

_set_az_rhi_title(*self*, *field*, *azimuth*, *title*, *ax*)
Set the figure title for a ray plot using a default title.

_set_ray_title(*self*, *field*, *ray*, *title*, *ax*)
Set the figure title for a ray plot using a default title.

_set_title(*self*, *field*, *sweep*, *title*, *ax*, *datetime_format=None*, *use_sweep_time=True*)
Set the figure title using a default title.

static _set_vpt_time_axis(*ax*, *date_time_form=None*, *tz=None*)
Set the x axis as a time formatted axis.

Parameters

ax [Matplotlib axis instance] Axis to plot. None will use the current axis.

date_time_form [str] Format of the time string for x-axis labels.

tz [str] Time zone info to use when creating axis labels (see datetime).

_set_vpt_title(*self*, *field*, *title*, *ax*)
Set the figure title using a default title.

generate_az_rhi_title(*self*, *field*, *azimuth*)
Generate a title for a ray plot.

Parameters

field [str] Field plotted.

azimuth [float] Azimuth plotted.

Returns

title [str] Plot title.

generate_filename(*self*, *field*, *sweep*, *ext='png'*, *datetime_format='%Y%m%d%H%M%S'*, *use_sweep_time=False*)

Generate a filename for a plot.

Generated filename has form: radar_name_field_sweep_time.ext

Parameters

field [str] Field plotted.

sweep [int] Sweep plotted.

ext [str] Filename extension.

datetime_format [str] Format of datetime (using strftime format).

use_sweep_time [bool] If true, the current sweep's beginning time is used.

Returns

filename [str] Filename suitable for saving a plot.

generate_ray_title(*self*, *field*, *ray*)

Generate a title for a ray plot.

Parameters

field [str] Field plotted.

ray [int] Ray plotted.

Returns

title [str] Plot title.

generate_title(*self*, *field*, *sweep*, *datetime_format=None*, *use_sweep_time=True*)

Generate a title for a plot.

Parameters

field [str] Field plotted.

sweep [int] Sweep plotted.

datetime_format [str] Format of datetime (using strftime format).

use_sweep_time [bool] If true, the current sweep's beginning time is used.

Returns

title [str] Plot title.

generate_vpt_title(*self*, *field*)

Generate a title for a VPT plot.

Parameters

field [str] Field plotted.

Returns

title [str] Plot title.

label_xaxis_r(*self*, *ax=None*)
Label the xaxis with the default label for r units.

static label_xaxis_rays(*ax=None*)
Label the yaxis with the default label for rays.

static label_xaxis_time(*ax=None*)
Label the yaxis with the default label for rays.

label_xaxis_x(*self*, *ax=None*)
Label the xaxis with the default label for x units.

label_yaxis_field(*self*, *field*, *ax=None*)
Label the yaxis with the default label for a field units.

label_yaxis_y(*self*, *ax=None*)
Label the yaxis with the default label for y units.

label_yaxis_z(*self*, *ax=None*)
Label the yaxis with the default label for z units.

plot(*self*, *field*, *sweep=0*, ***kwargs*)
Create a plot appropiate for the radar.

This function calls the plotting function corresponding to the scan_type of the radar. Additional keywords can be passed to customize the plot, see the appropiate plot function for the allowed keywords.

Parameters

field [str] Field to plot.

sweep [int] Sweep number to plot, not used for VPT scans.

See also:

plot_ppi Plot a PPI scan

plot_rhi Plot a RHI scan

plot_vpt Plot a VPT scan

plot_azimuth_to_rhi(*self*, *field*, *target_azimuth*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *reverse_xaxis=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, ***kwargs*)
Plot pseudo-RHI scan by extracting the vertical field associated with the given azimuth.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

target_azimuth [integer] Azimuthal angle in degrees where cross section will be taken.

Other Parameters

mask_tuple [(str, float)] 2-Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

reverse_xaxis [bool or None] True to reverse the x-axis so the plot reads east to west, False to have east to west. None (the default) will reverse the axis only when all the distances are negative.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to True to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_colorbar(*self*, *mappable=None*, *field=None*, *label=None*, *orient='vertical'*, *cax=None*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*)

Plot a colorbar.

Parameters

mappable [Image, ContourSet, etc.] Image, ContourSet, etc to which the colorbar applied. If None the last mappable object will be used.

field [str] Field to label colorbar with.

label [str] Colorbar label. None will use a default value from the last field plotted.

orient [str] Colorbar orientation, either 'vertical' [default] or 'horizontal'.

cax [Axis] Axis onto which the colorbar will be drawn. None is also valid.

ax [Axes] Axis onto which the colorbar will be drawn. None is also valid.

fig [Figure] Figure to place colorbar on. None will use the current figure.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

static plot_cross_hair(*size*, *npts=100*, *ax=None*)

Plot a cross-hair on a ppi plot.

Parameters

size [float] Size of cross-hair in km.

npts: int Number of points in the cross-hair, higher for better resolution.

ax [Axis] Axis to plot on. None will use the current axis.

static plot_grid_lines(*ax=None*, *col='k'*, *ls=':'*)

Plot grid lines.

Parameters

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for grid lines.

ls [str] Linestyle to use for grid lines.

plot_label(*self*, *label*, *location*, *symbol='r+'*, *text_color='k'*, *ax=None*)

Plot a single symbol and label at a given location.

Transforms of the symbol location in latitude and longitude units to x and y plot units is performed using an azimuthal equidistance map projection centered at the radar.

Parameters

label [str] Label text to place just above symbol.

location [2-tuples] Tuple of latitude, longitude (in degrees) at which the symbol will be place. The label is placed just above the symbol.

symbol [str] Matplotlib color+marker strings defining the symbol to place at the given location.

text_color [str] Matplotlib color defining the color of the label text.

ax [Axis] Axis to plot on. None will use the current axis.

plot_labels(*self*, *labels*, *locations*, *symbols='r+'*, *text_color='k'*, *ax=None*)

Plot symbols and labels at given locations.

Parameters

labels [list of str] List of labels to place just above symbols.

locations [list of 2-tuples] List of latitude, longitude (in degrees) tuples at which symbols will be place. Labels are placed just above the symbols.

symbols [list of str or str] List of matplotlib color+marker strings defining symbols to place at given locations. If a single string is provided, that symbol will be placed at all locations.

text_color [str] Matplotlib color defining the color of the label text.

ax [Axis] Axis to plot on. None will use the current axis.

plot_line_geo(*self*, *line_lons*, *line_lats*, *line_style='r-'*, ***kwargs*)
Plot a line segments on the current map given values in lat and lon.

Additional arguments are passed to basemap.plot.

Parameters

line_lons [array] Longitude of line segment to plot.

line_lats [array] Latitude of line segment to plot.

line_style [str] Matplotlib compatible string which specifies the line style.

plot_line_xy(*self*, *line_x*, *line_y*, *line_style='r-'*, ***kwargs*)
Plot a line segments on the current map given radar x, y values.

Additional arguments are passed to basemap.plot.

Parameters

line_x [array] X location of points to plot in meters from the radar.

line_y [array] Y location of points to plot in meters from the radar.

line_style [str, optional] Matplotlib compatible string which specifies the line style.

plot_point(*self*, *lon*, *lat*, *symbol='ro'*, *label_text=None*, *label_offset=(None*, *None)*, ***kwargs*)
Plot a point on the current map.

Additional arguments are passed to basemap.plot.

Parameters

lon [float] Longitude of point to plot.

lat [float] Latitude of point to plot.

symbol [str] Matplotlib compatible string which specified the symbol of the point.

label_text [str, optional.] Text to label symbol with. If None no label will be added.

label_offset [[float, float]] Offset in lon, lat degrees for the bottom left corner of the label text relative to the point. A value of None will use 0.01 de

plot_ppi(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, *title_datetime_format=None*, *title_use_sweep_time=True*, ***kwargs*)
Plot a PPI.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int, optional] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_datetime_format [str] Format of datetime in the title (using strftime format).

title_use_sweep_time [bool] True for the current sweep's beginning time to be used for the title. False for the radar's beginning time.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_ppi_map(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *cmap=None*, *norm=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *ax=None*, *fig=None*, *lat_lines=None*, *lon_lines=None*, *projection='lcc'*, *area_thresh=10000*, *min_lon=None*, *max_lon=None*, *min_lat=None*, *max_lat=None*, *width=None*, *height=None*, *lon_0=None*, *lat_0=None*, *resolution='h'*, *shapefile=None*, *edges=True*, *gatefilter=None*, *basemap=None*, *filter_transitions=True*, *embelish=True*, *ticks=None*, *ticklabs=None*, *raster=False*, *alpha=None*, ***kwargs*)

Plot a PPI volume sweep onto a geographic map.

Additional arguments are passed to Basemap.

Parameters

field [str] Field to plot.

sweep [int, optional] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and tilt parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

lat_lines, lon_lines [array or None] Locations at which to draw latitude and longitude lines. None will use default values which are resonable for maps of North America.

projection [str] Map projection supported by basemap. The use of cylindrical projections (mill, merc, etc) is not recommended as they exhibit large distortions at high latitudes. Equal area (aea, laea), conformal (lcc, tmerc, stere) or equidistant projection (aeqd, cass) work well even at high latitudes. The cylindrical equidistant projection (cyl) is not supported as coordinate transformations cannot be performed.

area_thresh [float] Coastline or lake with an area smaller than area_thresh in km^2 will not be plotted.

min_lat, max_lat, min_lon, max_lon [float] Latitude and longitude ranges for the map projection region in degrees.

width, height [float] Width and height of map domain in meters. Only this set of parameters or the previous set of parameters (min_lat, max_lat, min_lon, max_lon) should be specified. If neither set is specified then the map domain will be determined from the extend of the radar gate locations.

lon_0, lat_0 [float] Center of the map domain in degrees. If the default, None is used the latitude and longitude of the radar will be used.

shapefile [str] Filename for a ESRI shapefile as background (untested).

resolution ['c', 'l', 'i', 'h', or 'f'.] Resolution of boundary database to use. See Basemap documentation for details.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselved as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

embelish: bool True by default. Set to false to supress drawing of coastlines etc.. Use for speedup when specifying shapefiles.

basemap: Basemap instance If None, create basemap instance using other keyword info. If not None, use the user-specifed basemap instance.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

alpha [float or None] Set the alpha tranparency of the radar plot. Useful for overplotting radar over other datasets.

plot_range_ring(*self*, *range_ring_location_km*, *npts=360*, *line_style='k-'*, ***kwargs*)

Plot a single range ring on the map.

Additional arguments are passed to basemap.plot.

Parameters

range_ring_location_km [float] Location of range ring in km.

npts: int Number of points in the ring, higher for better resolution.

line_style [str] Matplotlib compatible string which specified the line style of the ring.

plot_range_rings(*self*, *range_rings*, *ax=None*, *col='k'*, *ls='-'*, *lw=2*)
Plot a series of range rings.

Parameters

range_rings [list] List of locations in km to draw range rings.

ax [Axis] Axis to plot on. None will use the current axis.

col [str or value] Color to use for range rings.

ls [str] Linestyle to use for range rings.

plot_ray(*self*, *field*, *ray*, *format_str='k-'*, *mask_tuple=None*, *ray_min=None*, *ray_max=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *gatefilter=None*, *axislabels_flag=True*, *ax=None*, *fig=None*)
Plot a single ray.

Parameters

field [str] Field to plot.

ray [int] Ray number to plot.

Other Parameters

format_str [str] Format string defining the line style and marker.

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

ray_min [float] Minimum ray value, None for default value, ignored if mask_outside is False.

ray_max [float] Maximum ray value, None for default value, ignored if mask_outside is False.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and ray parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

plot_rhi(*self*, *field*, *sweep=0*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *reverse_xaxis=None*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, *title_datetime_format=None*, *title_use_sweep_time=True*, ***kwargs*)

Plot a RHI.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

sweep [int,] Sweep number to plot.

Other Parameters

mask_tuple [(str, float)] 2-Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_datetime_format [str] Format of datetime in the title (using strftime format).

title_use_sweep_time [bool] True for the current sweep's beginning time to be used for the title. False for the radar's beginning time.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

reverse_xaxis [bool or None] True to reverse the x-axis so the plot reads west to east, False to have east to west. None (the default) will reverse the axis only when all the distances are negative. (i.e) axis will be absolute distance without taking into consideration the orientation

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

plot_vpt(*self*, *field*, *mask_tuple=None*, *vmin=None*, *vmax=None*, *norm=None*, *cmap=None*, *mask_outside=False*, *title=None*, *title_flag=True*, *axislabels=(None*, *None)*, *axislabels_flag=True*, *colorbar_flag=True*, *colorbar_label=None*, *colorbar_orient='vertical'*, *edges=True*, *gatefilter=None*, *filter_transitions=True*, *time_axis_flag=False*, *date_time_form=None*, *tz=None*, *ax=None*, *fig=None*, *ticks=None*, *ticklabs=None*, *raster=False*, ***kwargs*)

Plot a VPT scan.

Additional arguments are passed to Matplotlib's pcolormesh function.

Parameters

field [str] Field to plot.

Other Parameters

mask_tuple [(str, float)] Tuple containing the field name and value below which to mask field prior to plotting, for example to mask all data where NCP < 0.5 set mask_tuple to ['NCP', 0.5]. None performs no masking.

vmin [float] Luminance minimum value, None for default value. Parameter is ignored is norm is not None.

vmax [float] Luminance maximum value, None for default value. Parameter is ignored is norm is not None.

norm [Normalize or None, optional] matplotlib Normalize instance used to scale luminance data. If not None the vmax and vmin parameters are ignored. If None, vmin and vmax are used for luminance scaling.

cmap [str or None] Matplotlib colormap name. None will use the default colormap for the field being plotted as specified by the Py-ART configuration.

mask_outside [bool] True to mask data outside of vmin, vmax. False performs no masking.

title [str] Title to label plot with, None to use default title generated from the field and sweep parameters. Parameter is ignored if title_flag is False.

title_flag [bool] True to add a title to the plot, False does not add a title.

axislabels [(str, str)] 2-tuple of x-axis, y-axis labels. None for either label will use the default axis label. Parameter is ignored if axislabels_flag is False.

axislabels_flag [bool] True to add label the axes, False does not label the axes.

colorbar_flag [bool] True to add a colorbar with label to the axis. False leaves off the colorbar.

colorbar_label [str] Colorbar label, None will use a default label generated from the field information.

ticks [array] Colorbar custom tick label locations.

ticklabs [array] Colorbar custom tick labels.

colorbar_orient ['vertical' or 'horizontal'] Colorbar orientation.

edges [bool] True will interpolate and extrapolate the gate edges from the range, azimuth and elevations in the radar, treating these as specifying the center of each gate. False treats these coordinates themselves as the gate edges, resulting in a plot in which the last gate in each ray and the entire last ray are not not plotted.

gatefilter [GateFilter] GateFilter instance. None will result in no gatefilter mask being applied to data.

filter_transitions [bool] True to remove rays where the antenna was in transition between sweeps from the plot. False will include these rays in the plot. No rays are filtered when the antenna_transition attribute of the underlying radar is not present.

time_axis_flag [bool] True to plot the x-axis as time. False uses the index number. Default is False - index-based.

date_time_form [str, optional] Format of the time string for x-axis labels. Parameter is ignored if time_axis_flag is set to False.

tz [str, optional] Time zone info to use when creating axis labels (see datetime). Parameter is ignored if time_axis_flag is set to False.

ax [Axis] Axis to plot on. None will use the current axis.

fig [Figure] Figure to add the colorbar to. None will use the current figure.

raster [bool] False by default. Set to true to render the display as a raster rather than a vector in call to pcolormesh. Saves time in plotting high resolution data over large areas. Be sure to set the dpi of the plot for your application if you save it as a vector format (i.e., pdf, eps, svg).

static set_aspect_ratio(*aspect_ratio=0.75*, *ax=None*)
Set the aspect ratio for plot area.

static set_limits(*xlim=None*, *ylim=None*, *ax=None*)
Set the display limits.

Parameters

xlim [tuple, optional] 2-Tuple containing y-axis limits in km. None uses default limits.

ylim [tuple, optional] 2-Tuple containing x-axis limits in km. None uses default limits.

ax [Axis] Axis to adjust. None will adjust the current axis.

CHAPTER

TEN

UTILITIES (PYART.UTIL)

Miscellaneous utility functions.

The location and names of these functions within Py-ART may change between versions without depreciation, use with caution.

10.1 Direction statistics

angular_mean(angles)	Compute the mean of a distribution of angles in radians.
angular_std(angles)	Compute the standard deviation of a distribution of angles in radians.
angular_mean_deg(angles)	Compute the mean of a distribution of angles in degrees.
angular_std_deg(angles)	Compute the standard deviation of a distribution of angles in degrees.
interval_mean(dist, interval_min, interval_max)	Compute the mean of a distribution within an interval.
interval_std(dist, interval_min, interval_max)	Compute the standard deviation of a distribution within an interval.
mean_of_two_angles(angles1, angles2)	Compute the element by element mean of two sets of angles.
mean_of_two_angles_deg(angle1, angle2)	Compute the element by element mean of two sets of angles in degrees.

10.2 Miscellaneous functions

cross_section_ppi(radar, target_azimuths[, ...])	Extract cross sections from a PPI volume along one or more azimuth angles.
cross_section_rhi(radar, target_elevations)	Extract cross sections from an RHI volume along one or more elevation angles.
colocated_gates(radar1, radar2[, h_tol, ...])	Flags radar gates of radar1 colocated with radar2
intersection(radar1, radar2[, h_tol, ...])	Flags region of radar1 that is intersecting with radar2 and complies with criteria regarding visibility, altitude, range, elevation angle and azimuth angle
datetime_from_radar(radar[, epoch])	Return a datetime for the first ray in a Radar.
datetimes_from_radar(radar[, epoch])	Return an array of datetimes for the rays in a Radar.
datetime_from_dataset(dataset[, epoch])	Return a datetime for the first time in a netCDF Dataset.
datetimes_from_dataset(dataset[, epoch])	Return an array of datetimes for the times in a netCDF Dataset.

Continued on next page

Table 2 – continued from previous page

datetime_from_grid(grid[, epoch])	Return a datetime for the volume start in a Grid.
estimate_noise_hs74(spectrum[, navg, nnoise_min])	Estimate noise parameters of a Doppler spectrum.
is_vpt(radar[, offset])	Determine if a Radar appears to be a vertical pointing scan.
to_vpt(radar[, single_scan])	Convert an existing Radar object to represent a vertical pointing scan.
join_radar(radar1, radar2)	Combine two radar instances into one.
join_spectra(spectra1, spectra2)	Combine two spectra instances into one.
cut_radar(radar, field_names[, rng_min, ...])	Cuts the radar object into new dimensions
cut_radar_spectra(radar, field_names[, ...])	Cuts the radar spectra object into new dimensions
radar_from_spectra(psr)	obtain a Radar object from a RadarSpectra object
interpol_spectra(psr[, kind, fill_value])	Interpolates the spectra so that it has a uniform grid
simulated_vel_from_profile(radar, profile[, ...])	Create simulated radial velocities from a profile of horizontal winds.
texture_along_ray(radar, var[, wind_size])	Compute field texture along ray using a user specified window size.
texture(radar, var)	Determine a texture field using an 11pt stdev texarray=texture(pyradarobj, field).
rolling_window(a, window)	create a rolling window object for application of functions eg: result=np.ma.std(array, 11), 1)
angular_texture_2d(image, N, interval)	Compute the angular texture of an image.

pyart.util.**angular_mean**(*angles*)

Compute the mean of a distribution of angles in radians.

Parameters

angles [array like] Distribution of angles in radians.

Returns

mean [float] The mean angle of the distribution in radians.

pyart.util.**angular_mean_deg**(*angles*)

Compute the mean of a distribution of angles in degrees.

Parameters

angles [array like] Distribution of angles in degrees.

Returns

mean [float] The mean angle of the distribution in degrees.

pyart.util.**angular_std**(*angles*)

Compute the standard deviation of a distribution of angles in radians.

Parameters

angles [array like] Distribution of angles in radians.

Returns

std [float] Standard deviation of the distribution.

pyart.util.**angular_std_deg**(*angles*)

Compute the standard deviation of a distribution of angles in degrees.

Parameters

angles [array like] Distribution of angles in degrees.

Returns

std [float] Standard deviation of the distribution.

pyart.util.**angular_texture_2d**(*image*, *N*, *interval*)

Compute the angular texture of an image. Uses convolutions in order to speed up texture calculation by a factor of ~50 compared to using ndimage.generic_filter.

Parameters

image [2D array of floats] The array containing the velocities in which to calculate texture from.

N [int] This is the window size for calculating texture. The texture will be calculated from an N by N window centered around the gate.

interval [float] The absolute value of the maximum velocity. In conversion to radial coordinates, pi will be defined to be interval and -pi will be -interval. It is recommended that interval be set to the Nyquist velocity.

Returns

std_dev [float array] Texture of the radial velocity field.

pyart.util.**colocated_gates**(*radar1*, *radar2*, *h_tol=0.0*, *latlon_tol=0.0*, *coloc_gates_field=None*)

Flags radar gates of radar1 colocated with radar2

Parameters

radar1 [Radar] radar object that is going to be flagged

radar2 [Radar] radar object

h_tol [float] tolerance in altitude [m]

latlon_tol [float] tolerance in latitude/longitude [deg]

coloc_gates_field [string] Name of the field to retrieve the data

Returns

coloc_dict [dict] a dictionary containing the colocated positions of radar 1 (ele, azi, rng) and radar 2

coloc_rad1 : field with the colocated gates of radar1 flagged, i.e: 1: not colocated gates 2: colocated (0 is reserved)

pyart.util.**cross_section_ppi**(*radar*, *target_azimuths*, *az_tol=None*)

Extract cross sections from a PPI volume along one or more azimuth angles.

Parameters

radar [Radar] Radar volume containing PPI sweeps from which azimuthal cross sections will be extracted.

target_azimuth [list] Azimuthal angles in degrees where cross sections will be taken.

az_tol [float, optional] Azimuth angle tolerance in degrees. If none the nearest angle is used. If valid only angles within the tolerance distance are considered.

Returns

radar_rhi [Radar] Radar volume containing RHI sweeps which contain azimuthal cross sections from the original PPI volume.

pyart.util.**cross_section_rhi**(*radar*, *target_elevations*, *el_tol=None*)
Extract cross sections from an RHI volume along one or more elevation angles.

Parameters

radar [Radar] Radar volume containing RHI sweeps from which azimuthal cross sections will be extracted.

target_elevations [list] Elevation angles in degrees where cross sections will be taken.

el_tol [float, optional] Elevation angle tolerance in degrees. If none the nearest angle is used. If valid only angles within the tolerance distance are considered.

Returns

radar_ppi [Radar] Radar volume containing PPI sweeps which contain azimuthal cross sections from the original RHI volume.

pyart.util.**cut_radar**(*radar*, *field_names*, *rng_min=None*, *rng_max=None*, *ele_min=None*, *ele_max=None*, *azi_min=None*, *azi_max=None*)
Cuts the radar object into new dimensions

Parameters

radar [radar object] The radar object containing the data

field_names [str or None] The fields to keep in the new radar

rng_min, rng_max [float] The range limits [m]. If None the entire coverage of the radar is going to be used

ele_min, ele_max, azi_min, azi_max [float or None] The limits of the grid [deg]. If None the limits will be the limits of the radar volume

Returns

radar [radar object] The radar object containing only the desired data

pyart.util.**cut_radar_spectra**(*radar*, *field_names*, *rng_min=None*, *rng_max=None*, *ele_min=None*, *ele_max=None*, *azi_min=None*, *azi_max=None*)
Cuts the radar spectra object into new dimensions

Parameters

radar [radar object] The radar object containing the data

field_names [str or None] The fields to keep in the new radar

rng_min, rng_max [float] The range limits [m]. If None the entire coverage of the radar is going to be used

ele_min, ele_max, azi_min, azi_max [float or None] The limits of the grid [deg]. If None the limits will be the limits of the radar volume

Returns

radar [radar object] The radar object containing only the desired data

pyart.util.**datetime_from_dataset**(*dataset*, *epoch=False*, ***kwargs*)
Return a datetime for the first time in a netCDF Dataset.

pyart.util.**datetime_from_grid**(*grid*, *epoch=False*, ***kwargs*)
Return a datetime for the volume start in a Grid.

pyart.util.**datetime_from_radar**(*radar*, *epoch=False*, ***kwargs*)
Return a datetime for the first ray in a Radar.

pyart.util.**datetimes_from_dataset**(*dataset*, *epoch=False*, ***kwargs*)
Return an array of datetimes for the times in a netCDF Dataset.

pyart.util.**datetimes_from_radar**(*radar*, *epoch=False*, ***kwargs*)
Return an array of datetimes for the rays in a Radar.

pyart.util.**estimate_noise_hs74**(*spectrum*, *navg=1*, *nnoise_min=1*)
Estimate noise parameters of a Doppler spectrum.

Use the method of estimating the noise level in Doppler spectra outlined by Hildebrand and Sehkon, 1974.

Parameters

spectrum [array like] Doppler spectrum in linear units.

navg [int, optional] The number of spectral bins over which a moving average has been taken. Corresponds to the **p** variable from equation 9 of the article. The default value of 1 is appropiate when no moving average has been applied to the spectrum.

nnoise_min [int] Minimum number of noise samples to consider the estimation valid

Returns

mean [float-like] Mean of points in the spectrum identified as noise.

threshold [float-like] Threshold separating noise from signal. The point in the spectrum with this value or below should be considered as noise, above this value signal. It is possible that all points in the spectrum are identified as noise. If a peak is required for moment calculation then the point with this value should be considered as signal.

var [float-like] Variance of the points in the spectrum identified as noise.

nnoise [int] Number of noise points in the spectrum.

References

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

pyart.util.**interpol_spectra**(*psr*, *kind='linear'*, *fill_value=0.0*)
Interpolates the spectra so that it has a uniform grid

Parameters

psr [RadarSpectra object] The original spectra

Returns

psr_interp [RadarSpectra object] The interpolated spectra

pyart.util.**intersection**(*radar1*, *radar2*, *h_tol=0.0*, *latlon_tol=0.0*, *vol_d_tol=None*, *vismin=None*, *hmin=None*, *hmax=None*, *rmin=None*, *rmax=None*, *elmin=None*, *elmax=None*, *azmin=None*, *azmax=None*, *visib_field=None*, *intersec_field=None*)
Flags region of radar1 that is intersecting with radar2 and complies with criteria regarding visibility, altitude, range, elevation angle and azimuth angle

Parameters

radar1 [Radar] radar object that is going to be flagged

radar2 [Radar] radar object checked for intersecting region

h_tol [float] tolerance in altitude [m]

latlon_tol [float] latitude and longitude tolerance [decimal deg]

vol_d_tol [float] pulse volume diameter tolerance [m]

vismin [float] minimum visibility [percentage]

hmin, hmax [floats] min and max altitude [m MSL]

rmin, rmax [floats] min and max range from radar [m]

elmin, elmax [floats] min and max elevation angle [deg]

azmin, azmax [floats] min and max azimuth angle [deg]

Returns

intersec_rad1_dict [dict] the field with the gates of radar1 in the same region as radar2 flagged, i.e.: 1 not intersecting, 2 intersecting, 0 is reserved

pyart.util.**interval_mean**(*dist*, *interval_min*, *interval_max*)

Compute the mean of a distribution within an interval.

Return the average of the array elements which are interpreted as being taken from a circular interval with endpoints given by interval_min and interval_max.

Parameters

dist [array like] Distribution of values within an interval.

interval_min, interval_max [float] The endpoints of the interval.

Returns

mean [float] The mean value of the distribution.

pyart.util.**interval_std**(*dist*, *interval_min*, *interval_max*)

Compute the standard deviation of a distribution within an interval.

Return the standard deviation of the array elements which are interpreted as being taken from a circular interval with endpoints given by interval_min and interval_max.

Parameters

dist [array_like] Distribution of values within an interval.

interval_min, interval_max [float] The endpoints of the interval.

Returns

std [float] The standard deviation of the distribution.

pyart.util.**is_vpt**(*radar*, *offset=0.5*)

Determine if a Radar appears to be a vertical pointing scan.

This function only verifies that the object is a vertical pointing scan, use the *to_vpt()* function to convert the radar to a vpt scan if this function returns True.

Parameters

radar [Radar] Radar object to determine if.

offset [float, optional] Maximum offset of the elevation from 90 degrees to still consider to be vertically pointing.

Returns

flag [bool] True if the radar appear to be verticle pointing, False if not.

pyart.util.**join_radar**(*radar1*, *radar2*)
Combine two radar instances into one.

Parameters

radar1 [Radar] Radar object.

radar2 [Radar] Radar object.

pyart.util.**join_spectra**(*spectra1*, *spectra2*)
Combine two spectra instances into one.

Parameters

spectra1 [spectra] spectra object.

spectra2 [spectra] spectra object.

pyart.util.**mean_of_two_angles**(*angles1*, *angles2*)
Compute the element by element mean of two sets of angles.

Parameters

angles1 [array] First set of angles in radians.

angles2 [array] Second set of angles in radians.

Returns

mean [array] Elements by element angular mean of the two sets of angles in radians.

pyart.util.**mean_of_two_angles_deg**(*angle1*, *angle2*)
Compute the element by element mean of two sets of angles in degrees.

Parameters

angle1 [array] First set of angles in degrees.

angle2 [array] Second set of angles in degrees.

Returns

mean [array] Elements by element angular mean of the two sets of angles in degrees.

pyart.util.**radar_from_spectra**(*psr*)
obtain a Radar object from a RadarSpectra object

Parameters

psr [RadarSpectra object] The reference object

Returns

radar [radar object] The new radar object

pyart.util.**rolling_window**(*a*, *window*)
create a rolling window object for application of functions eg: result=np.ma.std(array, 11), 1)

pyart.util.**simulated_vel_from_profile**(*radar*, *profile*, *interp_kind='linear'*, *sim_vel_field=None*)
Create simulated radial velocities from a profile of horizontal winds.

Parameters

radar [Radar] Radar instance which provides the scanning parameters for the simulated radial velocities.

profile [HorizontalWindProfile] Profile of horizontal winds.

interp_kind [str, optional] Specifies the kind of interpolation used to determine the winds at a given height. Must be one of 'linear', 'nearest', 'zero', 'slinear', 'quadratic', or 'cubic'. The the documentation for the SciPy scipy.interpolate.interp1d function for descriptions.

sim_vel_field [str, optional] Name to use for the simulated velocity field metadata. None will use the default field name from the Py-ART configuration file.

Returns

sim_vel [dict] Dictionary containing a radar field of simulated radial velocities.

pyart.util.**texture**(*radar*, *var*)

Determine a texture field using an 11pt stdev texarray=texture(pyradarobj, field).

pyart.util.**texture_along_ray**(*radar*, *var*, *wind_size=7*)

Compute field texture along ray using a user specified window size.

Parameters

radar [radar object] The radar object where the field is.

var [str] Name of the field which texture has to be computed.

wind_size [int, optional] Optional. Size of the rolling window used.

Returns

tex [radar field] The texture of the specified field.

pyart.util.**to_vpt**(*radar*, *single_scan=True*)

Convert an existing Radar object to represent a vertical pointing scan.

This function does not verify that the Radar object contains a vertical pointing scan. To perform such a check use *is_vpt()*.

Parameters

radar [Radar] Mislabeled vertical pointing scan Radar object to convert to be properly labeled. This object is converted in place, no copy of the existing data is made.

single_scan [bool, optional] True to convert the volume to a single scan, any azimuth angle data is lost. False will convert the scan to contain the same number of scans as rays, azimuth angles are retained.

CHAPTER

ELEVEN

TESTING UTILITIES (PYART.TESTING)

Utilities helpful when writing and running unit tests.

11.1 Testing functions

make_empty_ppi_radar(ngates, rays_per_sweep, ...)	Return an Radar object, representing a PPI scan.
make_target_radar()	Return a PPI radar with a target like reflectivity field.
make_single_ray_radar()	Return a PPI radar with a single ray taken from a ARM C-SAPR Radar.
make_velocity_aliased_radar([alias])	Return a PPI radar with a target like reflectivity field.
make_empty_grid(grid_shape, grid_limits)	Make an empty grid object without any fields or metadata.
make_target_grid()	Make a sample Grid with a rectangular target.
make_storm_grid()	Make a sample Grid with a rectangular storm target.
make_normal_storm(sigma, mu)	Make a sample Grid with a gaussian storm target.

11.2 Testing classes

InTemporaryDirectory([suffix, prefix, dir])	Create, return, and change directory to a temporary directory.

class pyart.testing.**InTemporaryDirectory**(*suffix=''*, *prefix='tmp'*, *dir=None*)

Bases: pyart.testing.tmpdirs.TemporaryDirectory

Create, return, and change directory to a temporary directory.

Examples

```
>>> import os
>>> my_cwd = os.getcwd()
>>> with InTemporaryDirectory() as tmpdir:
...     _ = open('test.txt', 'wt').write('some text')
...     assert os.path.isfile('test.txt')
...     assert os.path.isfile(os.path.join(tmpdir, 'test.txt'))
>>> os.path.exists(tmpdir)
False
```

(continues on next page)

(continued from previous page)

```
>>> os.getcwd() == my_cwd
True
```

Methods

cleanup	

__class__
alias of builtins.type

__delattr__(*self*, *name*, */*)
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'pyart.testing.tmpdirs', '__doc__': " Create,

__dir__(*self*, */*)
Default dir() implementation.

__enter__(*self*)

__eq__(*self*, *value*, */*)
Return self==value.

__exit__(*self*, *exc*, *value*, *tb*)

__format__(*self*, *format_spec*, */*)
Default object formatter.

__ge__(*self*, *value*, */*)
Return self>=value.

__getattribute__(*self*, *name*, */*)
Return getattr(self, name).

__gt__(*self*, *value*, */*)
Return self>value.

__hash__(*self*, */*)
Return hash(self).

__init__(*self*, *suffix=''*, *prefix='tmp'*, *dir=None*)
Initialize self. See help(type(self)) for accurate signature.

__init_subclass__()
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__le__(*self*, *value*, */*)
Return self<=value.

__lt__(*self*, *value*, */*)
Return self<value.

__module__ = 'pyart.testing.tmpdirs'

__ne__(*self*, *value*, */*)
Return self!=value.

__new__(**args*, ***kwargs*)
Create and return a new object. See help(type) for accurate signature.

__reduce__(*self*, */*)
Helper for pickle.

__reduce_ex__(*self*, *protocol*, */*)
Helper for pickle.

__repr__(*self*, */*)
Return repr(self).

__setattr__(*self*, *name*, *value*, */*)
Implement setattr(self, name, value).

__sizeof__(*self*, */*)
Size of object in memory, in bytes.

__str__(*self*, */*)
Return str(self).

__subclasshook__()
Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

cleanup(*self*)

pyart.testing.**make_empty_grid**(*grid_shape*, *grid_limits*)
Make an empty grid object without any fields or metadata.

Parameters

grid_shape [3-tuple of floats] Number of points in the grid (z, y, x).

grid_limits [3-tuple of 2-tuples] Minimum and maximum grid location (inclusive) in meters for the z, y, x coordinates.

Returns

grid [Grid] Empty Grid object, centered near the ARM SGP site (Oklahoma).

pyart.testing.**make_empty_ppi_radar**(*ngates*, *rays_per_sweep*, *nsweeps*)
Return an Radar object, representing a PPI scan.

Parameters

ngates [int] Number of gates per ray.

rays_per_sweep [int] Number of rays in each PPI sweep.

nsweeps [int] Number of sweeps.

Returns

radar [Radar] Radar object with no fields, other parameters are set to default values.

pyart.testing.**make_empty_rhi_radar**(*ngates*, *rays_per_sweep*, *nsweeps*)
Return an Radar object, representing a RHI scan.

Parameters

ngates [int] Number of gates per ray.

rays_per_sweep [int] Number of rays in each PPI sweep.

nsweeps [int] Number of sweeps.

Returns

radar [Radar] Radar object with no fields, other parameters are set to default values.

pyart.testing.**make_normal_storm**(*sigma*, *mu*)
Make a sample Grid with a gaussian storm target.

pyart.testing.**make_single_ray_radar**()
Return a PPI radar with a single ray taken from a ARM C-SAPR Radar.

Radar object returned has 'reflectivity_horizontal', 'norm_coherent_power', 'copol_coeff', 'dp_phase_shift', 'diff_phase', and 'differential_reflectivity' fields with no metadata but a 'data' key. This radar is used for unit tests in correct modules.

pyart.testing.**make_storm_grid**()
Make a sample Grid with a rectangular storm target.

pyart.testing.**make_target_grid**()
Make a sample Grid with a rectangular target.

pyart.testing.**make_target_radar**()
Return a PPI radar with a target like reflectivity field.

pyart.testing.**make_velocity_aliased_radar**(*alias=True*)
Return a PPI radar with a target like reflectivity field.

Set alias to False to return a de-aliased radar.

pyart.testing.**make_velocity_aliased_rhi_radar**(*alias=True*)
Return a RHI radar with a target like reflectivity field.

Set alias to False to return a de-aliased radar.

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] http://www.ncdc.noaa.gov/

[2] http://thredds.ucar.edu/thredds/catalog.html

[1] http://www.unidata.ucar.edu/software/netcdf-java/documentation.htm

[2] http://thredds.ucar.edu/thredds/catalog.html

[1] http://www.ncdc.noaa.gov/

[2] http://www.roc.noaa.gov/wsr88d/Level_III/Level3Info.asp

[1] Doviak and Zrnic, Doppler Radar and Weather Observations, Second Edition, 1993, p. 21.

[1] Snyder, J. P. Map Projections–A Working Manual. U. S. Geological Survey Professional Paper 1395, 1987, pp. 191-202.

[1] Snyder, J. P. Map Projections–A Working Manual. U. S. Geological Survey Professional Paper 1395, 1987, pp. 191-202.

[1] Miguel Arevallilo Herraez, David R. Burton, Michael J. Lalor, and Munther A. Gdeisat, "Fast two-dimensional phase-unwrapping algorithm based on sorting by reliability following a noncontinuous path", Journal Applied Optics, Vol. 41, No. 35 (2002) 7437,

[2] Abdul-Rahman, H., Gdeisat, M., Burton, D., & Lalor, M., "Fast three-dimensional phase-unwrapping algorithm based on sorting by reliability following a non-continuous path. In W. Osten, C. Gorecki, & E. L. Novak (Eds.), Optical Metrology (2005) 32–40, International Society for Optics and Photonics.

PYTHON MODULE INDEX

p

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

P

Q

R

S

T

U

V

W