pyrad library reference for users

Release 0.0.1

meteoswiss-mdr

Oct 20, 2016

CONTENTS

Contents:

CHAPTER

ONE

PROCESSING FLOW CONTROL (PYRAD.FLOW)

Functions to control the Pyrad data processing flow

main(cfgfile, starttime, endtime)	main flow control. Processes data over a given period of time

pyrad.flow.**main**(*cfgfile*, *starttime*, *endtime*)

main flow control. Processes data over a given period of time

Parameters **cfgfile** : str

path of the main config file

starttime, endtime : datetime object

start and end time of the data to be processed

Returns None

CHAPTER

TWO

DATASET PROCESSING (PYRAD.PROC)

Initiate the dataset processing.

2.1 Auxiliary functions

get_process_type(dataset_type)	maps the dataset type into its processing function and data set format
process_raw(procstatus, dscfg[, radar])	dummy function that returns the initial input data set
process_save_radar(procstatus, dscfg[, radar])	dummy function that allows to save the entire radar object
process_point_measurement(procstatus, dscfg)	Obtains the radar data at a point measurement

2.2 Echo classification and filtering

process_echo_id(procstatus, dscfg[, radar])	identifies echoes as 0: No data, 1: Noise, 2: Clutter,
process_echo_filter(procstatus, dscfg[, radar])	Masks all echo types that are not of the class specified in
process_filter_snr(procstatus, dscfg[, radar])	filters out low SNR echoes
process_hydroclass(procstatus, dscfg[, radar])	Classifies precipitation echoes

2.3 Phase processing and attenuation correction

process_estimate_phidp0(procstatus, dscfg[, ...])	estimates the system differential phase offset at each ray
process_correct_phidp0(procstatus, dscfg[, ...])	corrects phidp of the system phase
process_smooth_phidp_single_window(...[, radar])	corrects phidp of the system phase and smoothes it using one window
process_smooth_phidp_double_window(...[, radar])	corrects phidp of the system phase and smoothes it using one window
process_kdp_leastsquare_single_window(...[, ...])	Computes specific differential phase using a piecewise least square method
process_kdp_leastsquare_double_window(...[, ...])	Computes specific differential phase using a piecewise least square method
process_phidp_kdp_Maesaka(procstatus, dscfg)	Estimates PhiDP and KDP using the method by Maesaka
process_phidp_kdp_lp(procstatus, dscfg[, radar])	Estimates PhiDP and KDP using a linear programming algorithm

Continued on next page

Table 2.3 – continued from previous page

process_attenuation(procstatus, dscfg[, radar])	Computes specific attenuation and specific differential attenuation using

2.4 Monitoring, calibration and noise correction

process_correct_bias(procstatus, dscfg[, radar])	Corrects a bias on the data
process_correct_noise_rhohv(procstatus, dscfg)	identifies echoes as 0: No data, 1: Noise, 2: Clutter,
process_rhohv_rain(procstatus, dscfg[, radar])	Computes quantiles of RhoHV in rain
process_sun_hits(procstatus, dscfg[, radar])	monitoring of the radar using sun hits
process_selfconsistency_kdp_phidp(...[, radar])	Computes specific differential phase and differential phase in rain using
process_selfconsistency_bias(procstatus, dscfg)	Estimates the reflectivity bias by means of the selfconsistency

2.5 Retrievals

process_signal_power(procstatus, dscfg[, radar])	Computes the signal power in dBm
process_snr(procstatus, dscfg[, radar])	Computes SNR
process_l(procstatus, dscfg[, radar])	Computes L parameter
process_cdr(procstatus, dscfg[, radar])	Computes Circular Depolarization Ratio
process_rainrate(procstatus, dscfg[, radar])	Estimates rainfall rate from polarimetric moments

pyrad.proc.**get_process_type**(*dataset_type*)

maps the dataset type into its processing function and data set format

Parameters **dataset_type** : str

data set type, i.e. 'RAW', 'SAN', etc.

Returns **func_name** : str

pyrad function used to process the data set type

dsformat : str

data set format, i.e.: 'VOL', etc.

pyrad.proc.**process_attenuation**(*procstatus*, *dscfg*, *radar=None*)

Computes specific attenuation and specific differential attenuation using the Z-Phi method and corrects reflectivity and differential reflectivity

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

ATT_METHOD [float. Dataset keyword] The attenuation estimation method used. One of the following: ZPhi, Philin

radar : Radar

Optional. Radar object

Returns **radar** : Radar

radar object

pyrad.proc.**process_cdr**(*procstatus*, *dscfg*, *radar=None*)

Computes Circular Depolarization Ratio

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [string. Dataset keyword] The input data type

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_correct_bias**(*procstatus*, *dscfg*, *radar=None*)

Corrects a bias on the data

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [string. Dataset keyword] The data type to correct for bias

bias [float. Dataset keyword] The bias to be corrected [dB]

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_correct_noise_rhohv**(*procstatus*, *dscfg*, *radar=None*)

identifies echoes as 0: No data, 1: Noise, 2: Clutter, 3: Precipitation

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The data types used in the correction

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_correct_phidp0**(*procstatus*, *dscfg*, *radar=None*)
corrects phidp of the system phase

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rmin [float. Dataset keyword] The minimum range where to look for valid data [m]

rmax [float. Dataset keyword] The maximum range where to look for valid data [m]

rcell [float. Dataset keyword] The length of a continuous cell to consider it valid precip [m]

Zmin [float. Dataset keyword] The minimum reflectivity [dBZ]

Zmax [float. Dataset keyword] The maximum reflectivity [dBZ]

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_echo_filter**(*procstatus*, *dscfg*, *radar=None*)
Masks all echo types that are not of the class specified in keyword echo_type

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

echo_type [int] The type of echo to keep: 1 noise, 2 clutter, 3 precipitation

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_echo_id**(*procstatus*, *dscfg*, *radar=None*)
identifies echoes as 0: No data, 1: Noise, 2: Clutter, 3: Precipitation

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_estimate_phidp0**(*procstatus*, *dscfg*, *radar=None*)
estimates the system differential phase offset at each ray

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rmin [float. Dataset keyword] The minimum range where to look for valid data [m]

rmax [float. Dataset keyword] The maximum range where to look for valid data [m]

rcell [float. Dataset keyword] The length of a continuous cell to consider it valid precip [m]

Zmin [float. Dataset keyword] The minimum reflectivity [dBZ]

Zmax [float. Dataset keyword] The maximum reflectivity [dBZ]

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_filter_snr**(*procstatus*, *dscfg*, *radar=None*)
filters out low SNR echoes

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

SNRmin [float. Dataset keyword] The minimum SNR to keep the data.

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_hydroclass**(*procstatus*, *dscfg*, *radar=None*)
Classifies precipitation echoes

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

HYDRO_METHOD [string. Dataset keyword] The hydrometeor classification method. One of the following: SEMISUPERVISED

RADARCENTROIDS [string. Datset keyword] Used with HYDRO_METHOD SEMISUPERVISED. The name of the radar of which the derived centroids will be used. One of the following: A Albis, L Lema, P Plaine Morte, DX50

radar : Radar

Optional. Radar object

Returns **radar** : Radar

radar object

pyrad.proc.**process_kdp_leastsquare_double_window**(*procstatus*, *dscfg*, *radar=None*)

Computes specific differential phase using a piecewise least square method

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rwinds [float. Dataset keyword] The length of the short segment for the least square method [m]

rwindl [float. Dataset keyword] The length of the long segment for the least square method [m]

Zthr [float. Dataset keyword] The threshold defining which estimated data to use [dBZ]

radar : Radar

Optional. Radar object

Returns **radar** : Radar

radar object

pyrad.proc.**process_kdp_leastsquare_single_window**(*procstatus*, *dscfg*, *radar=None*)

Computes specific differential phase using a piecewise least square method

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rwind [float. Dataset keyword] The length of the segment for the least square method [m]

radar : Radar

Optional. Radar object

Returns **radar** : Radar

radar object

pyrad.proc.**process_l**(*procstatus*, *dscfg*, *radar=None*)

Computes L parameter

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [string. Dataset keyword] The input data type

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_phidp_kdp_Maesaka**(*procstatus*, *dscfg*, *radar=None*)

Estimates PhiDP and KDP using the method by Maesaka

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rmin [float. Dataset keyword] The minimum range where to look for valid data [m]

rmax [float. Dataset keyword] The maximum range where to look for valid data [m]

rcell [float. Dataset keyword] The length of a continuous cell to consider it valid precip [m]

Zmin [float. Dataset keyword] The minimum reflectivity [dBZ]

Zmax [float. Dataset keyword] The maximum reflectivity [dBZ]

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_phidp_kdp_lp**(*procstatus*, *dscfg*, *radar=None*)

Estimates PhiDP and KDP using a linear programming algorithm

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_point_measurement**(*procstatus*, *dscfg*, *radar=None*)

Obtains the radar data at a point measurement

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [string. Dataset keyword] The data type where we want to extract the point measurement

latlon [boolean. Dataset keyword] if True position is obtained from latitude, longitude information, otherwise position is obtained from antenna coordinates (range, azimuth, elevation).

truealt [boolean. Dataset keyword] if True the user input altitude is used to determine the point of interest. if False use the altitude at a given radar elevation ele over the point of interest.

lon [float. Dataset keyword] the longitude [deg]. Use when latlon is True.

lat [float. Dataset keyword] the latitude [deg]. Use when latlon is True.

alt [float. Dataset keyword] altitude [m MSL]. Use when latlon is True.

ele [float. Dataset keyword] radar elevation [deg]. Use when latlon is False or when latlon is True and truealt is False

azi [float. Dataset keyword] radar azimuth [deg]. Use when latlon is False

rng [float. Dataset keyword] range from radar [m]. Use when latlon is False

AziTol [float. Dataset keyword] azimuthal tolerance to determine which radar azimuth to use [deg]

EleTol [float. Dataset keyword] elevation tolerance to determine which radar elevation to use [deg]

RngTol [float. Dataset keyword] range tolerance to determine which radar bin to use [m]

radar : Radar

Optional. Radar object

Returns **new_dataset** : dict

dictionary containing the data and metadata of the point of interest

pyrad.proc.**process_rainrate**(*procstatus*, *dscfg*, *radar=None*)

Estimates rainfall rate from polarimetric moments

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [string. Dataset keyword] The input data type

RR_METHOD [string. Dataset keyword] The rainfall rate estimation method. One of the following: Z, ZPoly, KDP, A, ZKDP, ZA, hydro

radar : Radar

Optional. Radar object

Returns **radar** : Radar

radar object

pyrad.proc.**process_raw**(*procstatus*, *dscfg*, *radar=None*)

dummy function that returns the initial input data set

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_rhohv_rain**(*procstatus*, *dscfg*, *radar=None*)

Computes quantiles of RhoHV in rain

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rmin [float. Dataset keyword] minimum range where to look for rain [m]

rmax [float. Dataset keyword] maximum range where to look for rain [m]

Zmin [float. Dataset keyword] minimum reflectivity to consider the bin as precipitation [dBZ]

Zmax [float. Dataset keyword] maximum reflectivity to consider the bin as precipitation [dBZ]

radar : Radar

Optional. Radar object

Returns **radar** : Radar

radar object

pyrad.proc.**process_save_radar**(*procstatus*, *dscfg*, *radar=None*)

dummy function that allows to save the entire radar object

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_selfconsistency_bias**(*procstatus*, *dscfg*, *radar=None*)

Estimates the reflectivity bias by means of the selfconsistency algorithm by Gourley

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rsmooth [float. Dataset keyword] length of the smoothing window [m]

radar : Radar

Optional. Radar object

Returns **radar** : Radar

radar object

pyrad.proc.**process_selfconsistency_kdp_phidp**(*procstatus*, *dscfg*, *radar=None*)

Computes specific differential phase and differential phase in rain using the selfconsistency between Zdr, Zh and KDP

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of strings. Dataset keyword] The input data types

rsmooth [float. Dataset keyword] length of the smoothing window [m]

radar : Radar

Optional. Radar object

Returns **radar** : Radar

radar object

pyrad.proc.**process_signal_power**(*procstatus*, *dscfg*, *radar=None*)

Computes the signal power in dBm

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

mflossv [float. Global keyword] The matching filter losses of the vertical channel. Used if input is vertical reflectivity

radconstv [float. Global keyword] The vertical channel radar constant. Used if input is vertical reflectivity

mflossh [float. Global keyword] The matching filter losses of the vertical channel. Used if input is horizontal reflectivity

radconsth [float. Global keyword] The horizontal channel radar constant. Used if input is horizontal reflectivity

attg [float. Dataset keyword] The gas attenuation

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_smooth_phidp_double_window**(*procstatus*, *dscfg*, *radar=None*)

corrects phidp of the system phase and smoothes it using one window

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rmin [float. Dataset keyword] The minimum range where to look for valid data [m]

rmax [float. Dataset keyword] The maximum range where to look for valid data [m]

rcell [float. Dataset keyword] The length of a continuous cell to consider it valid precip [m]

rwinds [float. Dataset keyword] The length of the short smoothing window [m]

rwindl [float. Dataset keyword] The length of the long smoothing window [m]

Zmin [float. Dataset keyword] The minimum reflectivity [dBZ]

Zmax [float. Dataset keyword] The maximum reflectivity [dBZ]

Zthr [float. Dataset keyword] The threshold defining wich smoothed data to used [dBZ]

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_smooth_phidp_single_window**(*procstatus*, *dscfg*, *radar=None*)

corrects phidp of the system phase and smoothes it using one window

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rmin [float. Dataset keyword] The minimum range where to look for valid data [m]

rmax [float. Dataset keyword] The maximum range where to look for valid data [m]

rcell [float. Dataset keyword] The length of a continuous cell to consider it valid precip [m]

rwind [float. Dataset keyword] The length of the smoothing window [m]

Zmin [float. Dataset keyword] The minimum reflectivity [dBZ]

Zmax [float. Dataset keyword] The maximum reflectivity [dBZ]

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_snr**(*procstatus*, *dscfg*, *radar=None*)

Computes SNR

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [string. Dataset keyword] The input data type

output_type [string. Dataset keyword] The output data type. Either SNRh or SNRv

radar : Radar

Optional. Radar object

Returns **new_dataset** : Radar

radar object

pyrad.proc.**process_sun_hits**(*procstatus*, *dscfg*, *radar=None*)

monitoring of the radar using sun hits

Parameters **procstatus** : int

Processing status: 0 initializing, 1 processing volume, 2 post-processing

dscfg : dictionary of dictionaries

data set configuration. Accepted Configuration Keywords:

datatype [list of string. Dataset keyword] The input data types

rmin [float. Dataset keyword] minimum range where to look for a sun hit signal [m]

delev_max [float. Dataset keyword] maximum elevation distance from nominal radar elevation where to look for a sun hit signal [deg]

dazim_max [float. Dataset keyword] maximum azimuth distance from nominal radar elevation where to look for a sun hit signal [deg]

elmin [float. Dataset keyword] minimum radar elevation where to look for sun hits [deg]

percent_bins [float. Dataset keyword] minimum percentage of range bins that have to contain signal to consider the ray a potential sun hit

attg [float. Dataset keyword] gaseous attenuation

max_std [float. Dataset keyword] maximum standard deviation to consider the data noise

az_width_co [float. Dataset keyword] co-polar antenna azimuth width (convoluted with sun width) [deg]

el_width_co [float. Dataset keyword] co-polar antenna elevation width (convoluted with sun width) [deg]

az_width_cross [float. Dataset keyword] cross-polar antenna azimuth width (convoluted with sun width) [deg]

el_width_cross [float. Dataset keyword] cross-polar antenna elevation width (convoluted with sun width) [deg]

ndays [int. Dataset keyword] number of days used in sun retrieval

radar : Radar

Optional. Radar object

Returns **sun_hits_dict** : dict

dictionary containing a radar object, a sun_hits dict and a sun_retrieval dictionary

---

CHAPTER

THREE

PRODUCTS GENERATION (PYRAD.PROD)

Initiate the products generation.

3.1 Auxiliary functions

get_product_type(product_type)	maps the product type into its processing function

3.2 Product generation

generate_vol_products(dataset, prdcfg)	generates radar volume products
generate_timeseries_products(dataset, prdcfg)	generates time series products
generate_sun_hits_products(dataset, prdcfg)	generates sun hits products

pyrad.prod.**generate_sun_hits_products**(*dataset*, *prdcfg*)
generates sun hits products

Parameters **dataset** : tuple

radar object and sun hits dictionary

prdcfg : dictionary of dictionaries

product configuration dictionary of dictionaries

Returns **filename** : str

the name of the file created. None otherwise

pyrad.prod.**generate_timeseries_products**(*dataset*, *prdcfg*)
generates time series products

Parameters **dataset** : dictionary

radar object

prdcfg : dictionary of dictionaries

product configuration dictionary of dictionaries

Returns no return

pyrad.prod.**generate_vol_products**(*dataset*, *prdcfg*)
generates radar volume products

Parameters **dataset** : Radar

radar object

prdcfg : dictionary of dictionaries

product configuration dictionary of dictionaries

Returns no return

`pyrad.prod.`**`get_product_type`**(*product_type*)

maps the product type into its processing function

Parameters **product_type** : str

product type, i.e. 'VOL', etc.

Returns **func_name** : str

pyrad function used to generate the product

CHAPTER

FOUR

INPUT AND OUTPUT (pyrad.io)

Functions to read and write data and configuration files.

4.1 Reading configuration files

read_config(fname[, cfg])	Read a pyrad config file.

4.2 Reading radar data

get_data(voltime, datatypesdescr, cfg)	Reads pyrad input data.

4.3 Reading other data

read_status(voltime, cfg)	Reads rad4alp xml status file.
read_rad4alp_cosmo(fname, datatype)	Reads rad4alp COSMO data binary file.
read_timeseries(fname)	Reads a time series contained in a csv file
get_sensor_data(date, datatype, cfg)	Gets data from a point measurement sensor (rain gauge or disdrometer)
read_smn(fname)	Reads SwissMetNet data contained in a csv file
read_disdro_scattering(fname)	Reads scattering parameters computed from disdrometer data contained in a
read_sun_hits(fname)	Reads sun hits data contained in a csv file
read_sun_hits_multiple_days(cfg[, nfiles])	Reads sun hits data from multiple file sources
read_sun_retrieval(fname)	Reads sun retrieval data contained in a csv file
read_selfconsistency(fname)	Reads a self-consistency table with Zdr, Kdp/Zh columns

4.4 Writing data

write_timeseries(dataset, fname)	writes time series of data
write_sun_hits(sun_hits, fname)	Writes sun hits data.
write_sun_retrieval(sun_retrieval, fname)	Writes sun retrieval data.

4.5 Auxiliary functions

get_save_dir(basepath, procname, dsname, prdname)	obtains the path to a product directory and eventually creates it
make_filename(prdtype, dstype, dsname, ext)	creates a product file name
get_datetime(fname, datadescriptor)	gets date and time from file name
get_datasetfields	
get_file_list(scan, datadescriptor, ...)	gets the list of files with a time period
get_datatypefields	
get_fieldname_rainbow	
generate_field_name_str(datatype)	Generates a field name in a nice to read format.

pyrad.io.**generate_field_name_str**(*datatype*)
Generates a field name in a nice to read format.

Parameters **datatype** : str

The data type

Returns **field_str** : str

The field name

pyrad.io.**get_data**(*voltime*, *datatypesdescr*, *cfg*)
Reads pyrad input data.

Parameters **voltime** : datetime object

volume scan time

datatypesdescr : list

list of radar field types to read. Format : [radar file type]:[datatype]

cfg: dictionary of dictionaries

configuration info to figure out where the data is

Returns **radar** : Radar

radar object

pyrad.io.**get_dataset_fields**(*datasetdescr*)
splits the dataset type descriptor and provides each individual member

Parameters **datasetdescr** : str

dataset type. Format : [processing level]:[dataset type]

Returns **proclevel** : str

dataset processing level

dataset : str

dataset type, i.e. dBZ, ZDR, ISO0, ...

pyrad.io.**get_datatype_fields**(*datadescriptor*)
splits the data type descriptor and provides each individual member

Parameters **datadescriptor** : str

radar field type. Format : [radar file type]:[datatype]

Returns **datagroup** : str

data type group, i.e. RAINBOW, RAD4ALP, SAVED, COSMO, ...

datatype : str

data type, i.e. dBZ, ZDR, ISO0, ...

dataset : str

dataset type (for saved data only)

product : str

product type (for saved data only)

pyrad.io.**get_datetime**(*fname*, *datadescriptor*)

gets date and time from file name

Parameters **fname** : file name

datadescriptor : str

radar field type. Format : [radar file type]:[datatype]

Returns **fdatetime** : datetime object

date and time in file name

pyrad.io.**get_fieldname_pyart**(*datatype*)

maps de config file radar data type name into the corresponding rainbow Py-ART field name

Parameters **datatype** : str

config file radar data type name

Returns **field_name** : str

Py-ART field name

pyrad.io.**get_file_list**(*scan*, *datadescriptor*, *starttime*, *endtime*, *cfg*)

gets the list of files with a time period

Parameters **scan** : str

scan name

datadescriptor : str

radar field type. Format : [radar file type]:[datatype]

startime : datetime object

start of time period

endtime : datetime object

end of time period

cfg: dictionary of dictionaries

configuration info to figure out where the data is

Returns **radar** : Radar

radar object

pyrad.io.**get_save_dir**(*basepath*, *procname*, *dsname*, *prdname*, *timeinfo=None*, *timeformat='%Y-%m-%d'*, *create_dir=True*)

obtains the path to a product directory and eventually creates it

Parameters **basepath** : str

product base path

procname : str

name of processing space

dsname : str

data set name

prdname : str

product name

timeinfo : datetime

time info to generate the date directory. If None there is no time format in the path

timeformat : str

Optional. The time format.

create_dir : boolean

If True creates the directory

Returns **savedir** : str

path to product

pyrad.io.**get_sensor_data**(*date*, *datatype*, *cfg*)

Gets data from a point measurement sensor (rain gauge or disdrometer)

Parameters **date** : datetime object

measurement date

datatype : str

name of the data type to read

cfg : dictionary

dictionary containing sensor information

Returns **sensordate , sensorvalue, label, period** : tupple

date, value, type of sensor and measurement period

pyrad.io.**make_filename**(*prdtype*, *dstype*, *dsname*, *ext*, *prdcfginfo=None*, *timeinfo=None*, *timeformat='%Y%m%d%H%M%S'*)

creates a product file name

Parameters **timeinfo** : datetime

time info to generate the date directory

prdtype : str

product type, i.e. 'ppi', etc.

dstype : str

data set type, i.e. 'raw', etc.

dsname : str

data set name

ext : str

file name extension, i.e. 'png'

prdcfginfo : str

Optional. string to add product configuration information, i.e. 'el0.4'

timeformat : str

Optional. The time format

Returns **fname** : str

file name

pyrad.io.**read_config**(*fname*, *cfg=None*)

Read a pyrad config file.

Parameters **fname** : str

Name of the configuration file to read.

cfg : dict of dicts, optional

dictionary of dictionaries containing configuration parameters where the new parameters will be placed

Returns **cfg** : dict of dicts

dictionary of dictionaries containing the configuration parameters

pyrad.io.**read_disdro_scattering**(*fname*)

Reads scattering parameters computed from disdrometer data contained in a text file

Parameters **fname** : str

path of time series file

Returns **id, date , pressure, temp, rh, precip, wspeed, wdir** : arrays

The read values

pyrad.io.**read_rad4alp_cosmo**(*fname*, *datatype*)

Reads rad4alp COSMO data binary file.

Parameters **fname** : str

name of the file to read

datatype : str

name of the data type

Returns **field** : dictionary

The data field

pyrad.io.**read_selfconsistency**(*fname*)

Reads a self-consistency table with Zdr, Kdp/Zh columns

Parameters **fname** : str

path of time series file

Returns **zdr, kdpzh** : arrays

The read values

pyrad.io.**read_smn**(*fname*)
Reads SwissMetNet data contained in a csv file

Parameters **fname** : str

path of time series file

Returns **id, date , pressure, temp, rh, precip, wspeed, wdir** : tupple

The read values

pyrad.io.**read_status**(*voltime*, *cfg*)
Reads rad4alp xml status file.

Parameters **voltime** : datetime object

volume scan time

cfg: dictionary of dictionaries

configuration info to figure out where the data is

Returns **root** : root element object

The information contained in the status file

pyrad.io.**read_sun_hits**(*fname*)
Reads sun hits data contained in a csv file

Parameters **fname** : str

path of time series file

Returns date, ray, nrng, rad_el, rad_az, sun_el, sun_az, ph, ph_std, nph, nvalh,

pv, pv_std, npv, nvalv, zdr, zdr_std, nzdr, nvalzdr : tupple

Each parameter is an array containing a time series of information on a variable

pyrad.io.**read_sun_hits_multiple_days**(*cfg*, *nfiles=1*)
Reads sun hits data from multiple file sources

Parameters **cfg** : dict

dictionary with configuration data to find out the right file

nfiles : int

number of files to read

Returns date, ray, nrng, rad_el, rad_az, sun_el, sun_az, ph, ph_std, nph, nvalh,

pv, pv_std, npv, nvalv, zdr, zdr_std, nzdr, nvalzdr : tupple

Each parameter is an array containing a time series of information on a variable

pyrad.io.**read_sun_retrieval**(*fname*)
Reads sun retrieval data contained in a csv file

Parameters **fname** : str

path of time series file

Returns nhits_h, el_width_h, az_width_h, el_bias_h, az_bias_h, dBm_sun_est,

std_dBm_sun_est, nhits_v, el_width_v, az_width_v, el_bias_v, az_bias_v,

dBmv_sun_est, std_dBmv_sun_est, nhits_zdr, zdr_sun_est,

std_zdr_sun_est : tupple

Each parameter is an array containing a time series of information on a variable

pyrad.io.**read_timeseries**(*fname*)
Reads a time series contained in a csv file

Parameters **fname** : str

path of time series file

Returns **date , value** : tupple

A datetime object array containing the time and a numpy masked array containing the value. None otherwise

pyrad.io.**write_sun_hits**(*sun_hits*, *fname*)
Writes sun hits data.

Parameters **sun_hits** : dict

dictionary containing the sun hits parameters

fname : str

file name where to store the data

Returns **fname** : str

the name of the file where data has written

pyrad.io.**write_sun_retrieval**(*sun_retrieval*, *fname*)
Writes sun retrieval data.

Parameters **sun_retrieval** : dict

dictionary containing the sun retrieval parameters

fname : str

file name where to store the data

Returns **fname** : str

the name of the file where data has written

pyrad.io.**write_timeseries**(*dataset*, *fname*)
writes time series of data

Parameters **dataset** : dict

dictionary containing the time series parameters

fname : str

file name where to store the data

Returns **fname** : str

the name of the file where data has written

CHAPTER

FIVE

PLOTTING (PYRAD.GRAPH)

Functions to plot graphics.

5.1 Plots

plot_ppi(radar, field_name, ind_el, prdcfg, ...)	plots a PPI
plot_rhi(radar, field_name, ind_az, prdcfg, ...)	plots an RHI
plot_cappi(radar, field_name, altitude, ...)	plots a Constant Altitude Plan Position Indicator CAPPI
plot_bscope(radar, field_name, ind_sweep, ...)	plots a B-Scope (angle-range representation)
plot_quantiles(quant, value, fname[, ...])	plots quantiles
plot_timeseries(date, value, fname[, ...])	plots a time series
plot_timeseries_comp(date1, value1, date2, ...)	plots 2 time series in the same graph
plot_sun_retrieval_ts(sun_retrieval, ...)	plots a time series
get_colobar_label(field_dict, field_name)	creates the colorbar label using field metadata

pyrad.graph.**get_colobar_label**(*field_dict*, *field_name*)

creates the colorbar label using field metadata

Parameters **field_dict** : dict

dictionary containing field metadata

field_name : str

name of the field

Returns **label** : str

colorbar label

pyrad.graph.**plot_bscope**(*radar*, *field_name*, *ind_sweep*, *prdcfg*, *fname*)

plots a B-Scope (angle-range representation)

Parameters **radar** : Radar object

object containing the radar data to plot

field_name : str

name of the radar field to plot

ind_sweep : int

sweep index to plot

prdcfg : dict

dictionary containing the product configuration

fname : str

name of the file where to store the plot

Returns fname : str

the name of the created plot file

pyrad.graph.**plot_cappi**(*radar*, *field_name*, *altitude*, *prdcfg*, *fname*)

plots a Constant Altitude Plan Position Indicator CAPPI

Parameters radar : Radar object

object containing the radar data to plot

field_name : str

name of the radar field to plot

altitude : float

the altitude [m MSL] to be plotted

prdcfg : dict

dictionary containing the product configuration

fname : str

name of the file where to store the plot

Returns fname : str

the name of the created plot file

pyrad.graph.**plot_ppi**(*radar*, *field_name*, *ind_el*, *prdcfg*, *fname*, *plot_type='PPI'*, *step=None*, *quantiles=None*)

plots a PPI

Parameters radar : Radar object

object containing the radar data to plot

field_name : str

name of the radar field to plot

ind_el : int

sweep index to plot

prdcfg : dict

dictionary containing the product configuration

fname : str

name of the file where to store the plot

plot_type : str

type of plot (PPI, QUANTILES or HISTOGRAM)

step : float

step for histogram plotting

quantiles : float array

quantiles to plot

Returns **fname** : str

the name of the created plot file

pyrad.graph.**plot_quantiles**(*quant*, *value*, *fname*, *labelx='quantile'*, *labely='value'*, *titl='quantile'*)

plots quantiles

Parameters **quant** : array

quantiles to be plotted

value : array

values of each quantie

fname : str

name of the file where to store the plot

labelx : str

The label of the X axis

labely : str

The label of the Y axis

titl : str

The figure title

Returns **fname** : str

the name of the created plot file

pyrad.graph.**plot_rhi**(*radar*, *field_name*, *ind_az*, *prdcfg*, *fname*, *plot_type='PPI'*, *step=None*, *quantiles=None*)

plots an RHI

Parameters **radar** : Radar object

object containing the radar data to plot

field_name : str

name of the radar field to plot

ind_az : int

sweep index to plot

prdcfg : dict

dictionary containing the product configuration

fname : str

name of the file where to store the plot

plot_type : str

type of plot (PPI, QUANTILES or HISTOGRAM)

step : float

step for histogram plotting

quantiles : float array

quantiles to plot

Returns **fname** : str

the name of the created plot file

pyrad.graph.**plot_sun_retrieval_ts**(*sun_retrieval*, *data_type*, *fname*)

plots a time series

Parameters **date** : datetime object

time of the time series

value : float array

values of the time series

fname : str

name of the file where to store the plot

labelx : str

The label of the X axis

labely : str

The label of the Y axis

label1 : str

The label of the legend

titl : str

The figure title

period : float

measurement period in seconds used to compute accumulation. If 0 no accumulation is computed

Returns **fname** : str

the name of the created plot file

pyrad.graph.**plot_timeseries**(*date*, *value*, *fname*, *labelx='Time [UTC]'*, *labely='Value'*, *label1='Sensor'*, *titl='Time Series'*, *period=0*)

plots a time series

Parameters **date** : datetime object

time of the time series

value : float array

values of the time series

fname : str

name of the file where to store the plot

labelx : str

The label of the X axis

labely : str

The label of the Y axis

label1 : str

The label of the legend

titl : str

The figure title

period : float

measurement period in seconds used to compute accumulation. If 0 no accumulation is computed

Returns **fname** : str

the name of the created plot file

pyrad.graph.**plot_timeseries_comp**(*date1*, *value1*, *date2*, *value2*, *fname*, *labelx='Time [UTC]'*, *labely='Value'*, *label1='Sensor 1'*, *label2='Sensor 2'*, *titl='Time Series Comparison'*, *period1=0*, *period2=0*)

plots 2 time series in the same graph

Parameters **date1** : datetime object

time of the first time series

value1 : float array

values of the first time series

date2 : datetime object

time of the second time series

value2 : float array

values of the second time series

fname : str

name of the file where to store the plot

labelx : str

The label of the X axis

labely : str

The label of the Y axis

label1, label2 : str

legend label for each time series

titl : str

The figure title

period1, period2 [float] measurement period in seconds used to compute accumulation. If 0 no accumulation is computed

Returns **fname** : str

the name of the created plot file

CHAPTER

SIX

UTILITIES (PYRAD.UTIL)

Functions to read and write data and configuration files.

6.1 Radar Utilities

create_sun_hits_field(rad_el, rad_az, ...)	creates a sun hits field from the position and power of the sun hits
create_sun_retrieval_field(par, imgcfg)	creates a sun retrieval field from the retrieval parameters
compute_quantiles_sweep(field, ray_start, ...)	computes quantiles of a particular sweep
compute_histogram_sweep(field, ray_start, ...)	computes histogram of the data in a particular sweep

pyrad.util.**compute_histogram_sweep**(*field*, *ray_start*, *ray_end*, *field_name*, *step=None*)

computes histogram of the data in a particular sweep

Parameters **field** : ndarray 2D

the radar field

ray_start, ray_end : int

starting and ending ray indexes

field_name: str

name of the field

step : float

size of bin

Returns **bins** : float array

interval of each bin

values : float array

values at each bin

pyrad.util.**compute_quantiles_sweep**(*field*, *ray_start*, *ray_end*, *quantiles=None*)

computes quantiles of a particular sweep

Parameters **field** : ndarray 2D

the radar field

ray_start, ray_end : int

starting and ending ray indexes

quantiles: float array

list of quantiles to compute

Returns **quantiles** : float array

list of quantiles

values : float array

values at each quantile

pyrad.util.**create_sun_hits_field**(*rad_el*, *rad_az*, *sun_el*, *sun_az*, *data*, *imgcfg*)

creates a sun hits field from the position and power of the sun hits

Parameters **rad_el, rad_az, sun_el, sun_az** : ndarray 1D

azimuth and elevation of the radar and the sun respectively in degree

data : masked ndarray 1D

the sun hit data

imgcfg: dict

a dictionary specifying the ranges and resolution of the field to create

Returns **field** : masked ndarray 2D

the sun hit field

pyrad.util.**create_sun_retrieval_field**(*par*, *imgcfg*)

creates a sun retrieval field from the retrieval parameters

Parameters **par** : ndarray 1D

the 5 retrieval parameters

imgcfg: dict

a dictionary specifying the ranges and resolution of the field to create

Returns **field** : masked ndarray 2D

the sun retrieval field

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

p

C

G

M

P

R

W