lidar processing Documentation

Release 0.1.0

Ioannis Binietoglou

May 20, 2022

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

This module collects basic processing routines for aerosol lidar systems. Its aim is to act as a repository of pre-processing and optical processing routines, that can be used as a basic building block for any atmospheric lidar processing software.

To make it reusable, the module includes only the pre-processing and optical processing functions. Reading data, visualization, etc. should be handled by different modules.

Note: Here is a list of complementary lidar-related modules:

Molecular scattering The lidar_molecular module is a collection of scripts to calculate scattering parameters of molecular atmosphere.

Raw lidar files The atmospheric-lidar module contains classes to read raw lidar data files, including Licel binary files. It can be used for plotting (quicklooks) and converting raw data to SCC format.

1.1 Documentation

Each function should be documented following the Numpy doc style.

For details see the numpy documentation.

All docstrings are collected to a single documentation file using the Sphinx module. The documentation is located in the docs/ folder. The documentation is written in restructured text format.

You can rebuild the docs by running the following command from the docs folder.

```
make html
```

The documentation is also built automatically every time you push your changes to the repository. You can find it online in Read the docs.

1.2 Testing

Some tests, based on unittest2 library, are located in the lidar_processing/tests/ folder.

You can run all the test using the commands from the project directory.

```
python -m unittest discover
```

1.3 Todo

The module is still in a very early stage so most things need to be done. Here is an indicative list of things to add:

- Signal gluing
- Optical product gluing (e.g. from near and far range telescopes).
- Error propagation (Monte Carlo method).
- Klett algorithm for elastic lidar retrieval
- Raman scattering algorithms for backscatter and extinction.

Even if you don't have something to code, there are other ways to contribute, e.g:

- Review/improve this documentation.
- Test that the implemented functions work correctly.
- Suggest missing routines or other improvements.

CHAPTER

TWO

PRE PROCESSING FUNCTIONS

CHAPTER

THREE

CHECKS FOR FITTING SIGNALS

This file contains functions that check if two signals fit or not. They can be used to check a gluing or molecular fit regions.

fit_checks.**check_correlation**(*first_signal*, *second_signal*, *threshold=None*)

Returns the correlation coefficient between the two signals.

The signals can be either 1D arrays or 2D arrays containing the rolling slices of the input signals. In the 2D case, the function returns the sliding correlation between the original signals.

If a threshold is provided, returns True if the correlation is above the specified threshold.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

threshold: float or None Threshold for the correlation coefficient.

Returns

correlation: float or boolean If threshold is None, then the function returns an the correlation coefficient. If a threshold is provided, the function returns True if the correlation value is above the threshold.

fit_checks.**check_linear_fit_intercept_and_correlation**(*first_signal*, *second_signal*)

Check if the intercept of a linear fit is near zero, and the correlation coefficient of the two signals.

Performs a linear fit to the data, assuming y = ax + b, with x the first_signal and y the second_signal. It will return the value np.abs(b / np.mean(y) * 100)

If the intercept is far from zero, it indicates that the two signals do not differ from a multiplication constant.

Parameters

first_signal [array] The first signal array

second_signal [array] The second signal array

Returns

intercept_percent [float or boolean] The value of the intercept b, relative to the mean value of the second_signal.

correlation [float] Correlation coefficient between the two samples

fit_checks.**check_min_max_ratio**(*first_signal*, *second_signal*, *threshold=None*)

Returns the ration between minimum and maximum values (i.e. min / max).

The operation is performed for both signals and the minimum is returned. The aim is to detect regions of large variation e.g. edges of clouds. Similar large values will be returned when the signals are near 0, so the relative difference is large. Consequently, this test should be used in parallel with checks e.g. about signal to noise ratio.

If a threshold is provided, returns True if the reltio is above the specified threshold.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

threshold: float or None Threshold for the correlation coefficient.

Returns

minmax: float or boolean If threshold is None, then the function returns the min/max ratio. If a threshold is provided, the function returns True if the correlation value is above the threshold.

fit_checks.**check_residuals_not_gaussian**(*first_signal*, *second_signal*, *threshold=None*)

Check if the residuals of the linear fit are not from a normal distribution.

The function uses a Shapiro-Wilk test on the residuals of a linear fit. Specifically, the function performs a linear fit to the data, assuming y = ax, and then calculates the residuals r = y - ax. It will return the p value of the Shapiro-Wilk test on the residuals.

If a threshold is provided, returns True if the p value is below the specified threshold, i.e. if the residuals are probably not gaussian.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

threshold: float or None Threshold for the Shapiro-Wilk p-value.

Returns

p_value: float or boolean If threshold is None, then the function returns the p-value of the Shapiro-Wilk test on the residuals. If a threshold is provided, the function returns True if p-value is below the threshold.

fit_checks.**check_residuals_not_gaussian_dagostino**(*first_signal*, *second_signal*, *threshold=None*)

Check if the residuals of the linear fit are not from a normal distribution.

The function uses a D'agostino - Pearsons's test on the residuals of a linear fit. Specifically, the function performs a linear fit to the data, assuming y = ax, and then calculates the residuals r = y - ax. It will return the p value of the D'agostino - Pearsons's omnibus test on the residuals.

If a threshold is provided, returns True if the p value is below the specified threshold, i.e. if the residuals are probably not gaussian.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

threshold: float or None Threshold for the Shapiro-Wilk p-value.

Returns

p_value: float or boolean If threshold is None, then the function returns the p-value of the D'agostino - Pearsons's test on the residuals. If a threshold is provided, the function returns True if p-value is below the threshold.

fit_checks.**sliding_check_correlation**(*first_signal*, *second_signal*, *window_length=11*, *threshold=None*)

Returns the sliding correlation coefficient between the two signals.

If a threshold is provided, returns True if the correlation is above the specified threshold.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

window_length: int The length of the window. It should be an odd number.

threshold: float or None Threshold for the correlation coefficient.

Returns

correlation: float or boolean If threshold is None, then the function returns an the correlation coefficient. If a threshold is provided, the function returns True if the correlation value is above the threshold.

fit_checks.**sliding_check_linear_fit_intercept_and_correlation**(*first_signal*, *second_signal*, *window_length=11*)

Check if the intercept of a linear fit is near zero.

Performs a linear fit to the data, assuming y = ax + b, with x the first_signal and y the second_signal.

It will return the value np.abs(b / np.mean(y) * 100) and the correlation of the two signals.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

window_length: int The length of the window. It should be an odd number.

Returns

intercepts [float or boolean] The value of the intercept b, relative to the mean value of the second_signal.

correlations [float] Correlation coefficient between the two samples

fit_checks.**sliding_check_min_max_ratio**(*first_signal*, *second_signal*, *window_length=11*, *threshold=None*)

Returns the sliding min/max ratio for both signals

If a threshold is provided, returns True if the min/max ratio is above the specified threshold.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

window_length: int The length of the window. It should be an odd number.

threshold: float or None Threshold for the correlation coefficient.

Returns

correlation: float or boolean If threshold is None, then the function returns an the correlation coefficient. If a threshold is provided, the function returns True if the correlation value is above the threshold.

fit_checks.**sliding_check_residuals_not_gaussian**(*first_signal*, *second_signal*, *window_length*, *threshold=None*)

Check if the residuals of the linear fit are not from a normal distribution.

The function uses a Shapiro-Wilk test on the residuals of a linear fit. Specifically, the function performs a linear fit to the data, assuming y = ax, and then calculates the residuals r = y - ax. It will return the p value of the Shapiro-Wilk test on the residuals.

If a threshold is provided, returns True if the p value is below the specified threshold, i.e. if the residuals are probably not gaussian.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

window_length: int The length of the window. It should be an odd number.

threshold: float or None Threshold for the Shapiro-Wilk p-value.

Returns

p_value: array If threshold is None, then the function returns the p-value of the Shapiro-Wilk test on the residuals. If a threshold is provided, the function returns True if p-value is below the threshold.

fit_checks.**sliding_check_residuals_not_gaussian_dagostino**(*first_signal*, *second_signal*, *window_length*, *threshold=None*)

Check if the residuals of the linear fit are not from a normal distribution.

The function uses a Shapiro-Wilk test on the residuals of a linear fit. Specifically, the function performs a linear fit to the data, assuming y = ax, and then calculates the residuals r = y - ax. It will return the p value of the Shapiro-Wilk test on the residuals.

If a threshold is provided, returns True if the p value is below the specified threshold, i.e. if the residuals are probably not gaussian.

Parameters

first_signal: array The first signal array

second_signal: array The second signal array

window_length: int The length of the window. It should be an odd number.

threshold: float or None Threshold for the Shapiro-Wilk p-value.

Returns

p_value: array If threshold is None, then the function returns the p-value of the Shapiro-Wilk test on the residuals. If a threshold is provided, the function returns True if p-value is below the threshold.

CHAPTER

FOUR

ELASTIC RETRIEVALS

Retrieval of aerosol optical properties from elastic lidar signals.

Todo: Implement iterative retrieval (Di Girollamo et al. 1999)

elastic_retrievals.**klett_backscatter_aerosol**(*range_corrected_signal*, *lidar_ratio_aerosol*, *beta_molecular*, *index_reference*, *reference_range*, *beta_aerosol_reference*, *bin_length*, *lidar_ratio_molecular=8.73965404*)

Calculation of aerosol backscatter coefficient using Klett algorithm.

The method also calculates aerosol backscatter above the reference altitude using forward integration approach.

Parameters

range_corrected_signal [float.] The range corrected signal.

lidar_ratio_aerosol [float.] The aerosol lidar ratio.

beta_molecular [array_like] The molecular backscatter coefficient. (m^-1 * sr^-1)

index_reference [integer] The index of the reference height. (bins)

reference_range [integer] The reference height range. (bins)

beta_aerosol_reference [float] The aerosol backscatter coefficient on the reference height. (m^-1 * sr^-1)

bin_length [float] The vertical bin length. (m)

lidar_ratio_molecular [float] The molecular lidar ratio. Default value is $8\pi/3$ which is a typical approximation.

Returns

beta_aerosol: float The aerosol backscatter coefficient. (m^-1 * sr^-1)

Notes

We estimate aerosol backscatter using the equation.

$$\beta_{aer}(R) = \frac{A}{B - C} - \beta_{mol}(R)$$

where

$$A = S(R) \cdot exp(-2 \int_{R_0}^{R} [L_{aer}(r) - L_{mol}] \cdot \beta_{mol}(r) dr)$$
$$B = \frac{S(R_0)}{\beta_{aer}(R_0) + \beta_{mol}(R_0)}$$
$$C = -2 \int_{R_0}^{R} L_{aer}(r) \cdot S(r) \cdot T(r, R_0) dr$$

with

$$T(r, R_0) = exp(-2 \int_{R_0}^{r} [L_{aer}(r') - L_{mol}] \cdot \beta_{mol}(r') \cdot dr')$$

and

- R the distance from the source,
- R_0 the distance between the source and the reference region,
- β_{aer} the aerosol backscatter coefficient,
- β_{mol} the molecular backscatter coefficient,
- $S(R)$ the range corrected signal,
- P the signal due to particle and molecular scattering,
- L_{aer} the aerosol lidar ratio (extinction-to-backscatter coefficient),
- L_{mol} the molecular lidar ratio.

Note that *lidar_ratio_molecular* should correspond to the *beta_molecular* i.e. they should both correspond to total or Cabannes signal.

References

Ansmann, A. and Muller, D.: Lidar and Atmospheric Aerosol Particles, in Lidar: Range-Resolved Optical Remote Sensing of the Atmosphere, vol. 102, edited by C. Weitkamp, Springer, New York., 2005. p. 111.

CHAPTER

FIVE

RAMAN RETRIEVALS

CHAPTER

SIX

LINEAR DEPOLARIZATION ESTIMATION

Calculation of volume and particle depolarization coefficient.

depolarization.**calibration_constant_cross_parallel_profile**(*signal_cross_plus45*, *signal_cross_minus45*, *signal_parallel_plus45*, *signal_parallel_minus45*, *t_cross*, *t_parallel*, *r_cross*, *r_parallel*)

Calculate the calibration constant in a lidar system that is able to detect the cross-to-parallel depolarization ratio.

Parameters

signal_cross_plus45: vector The input vertical profile from the cross channel. Calibrator angle phi=45.

signal_parallel_plus45: vector The input vertical profile from the total channel. Calibrator angle phi=45.

signal_cross_minus45: vector The input vertical profile from the cross channel. Calibrator angle phi=-45.

signal_parallel_minus45: vector The input vertical profile from the total channel. Calibrator angle phi=-45.

t_cross: float Transmittance of cross component through transmitted path.

t_parallel: float Transmittance of parallel component through transmitted path.

r_cross: float Transmittance of cross component through reflected path.

r_parallel: float Transmittance of parallel component through reflected path.

Returns

v_star_mean: float Calibration constant's mean value (vertical axis).

v_star_sem: float Calibration constant's standard error of the mean (vertical axis).

Notes

The calibration constant is calculated by the following formula:

$$V^* = \frac{[1 + \delta^V tan^2(\phi)]T_p + [tan^2(\phi) + \delta^V]T_s}{[1 + \delta^V tan^2(\phi)]R_p + [tan^2(\phi) + \delta^V]R_s} \cdot \delta^*(\phi)$$

References

Freudenthaler, V. et al. Depolarization ratio profiling at several wavelengths in pure Saharan dust during SAMUM 2006. Tellus, 61B, 165-179 (2008)

depolarization.**calibration_constant_cross_total_profile**(*signal_cross_plus45*, *signal_cross_minus45*, *signal_total_plus45*, *signal_total_minus45*, *r_cross*, *r_total*)

Calculate the calibration constant profile, in a lidar system that is able to detect the cross-to-total depolarization ratio.

Parameters

signal_cross_plus45: vector The input vertical profile from the cross channel. Calibrator angle phi=45.

signal_total_plus45: array The input vertical profile from the total channel. Calibrator angle phi=45.

signal_cross_minus45: vector The input vertical profile from the cross channel. Calibrator angle phi=-45.

signal_total_minus45: vector The input vertical profile from the total channel. Calibrator angle phi=-45.

r_cross: float The transmission ratio of the cross channel (Rc).

r_total: float The transmission ratio of the total channel (Rt).

Returns

c_profile: vector The vertical profile of the calibration constant.

Notes

The calibration constant is calculated by the following formula:

$$C = \frac{1 + R_t}{1 + R_c} \cdot \sqrt{\delta'_{+45} \cdot \delta'_{-45}}$$

References

Engelmann, R. et al. The automated multiwavelength Raman polarization and water-vapor lidar Polly XT: the neXT generation. Atmos. Meas. Tech., 9, 1767-1784 (2016)

depolarization.**calibration_constant_value**(*calibration_constant_profile*, *first_bin*, *bin_length*, *lower_limit*, *upper_limit*)

Calculate the mean calibration constant and its standard error of the mean, from the calibration constant profile.

Parameters

c_profile: vector The vertical profile of the calibration constant.

first_bin: integer The first bin of the system.

bin_length: float The length of each bin. (in meters)

lower_limit: float The lower vertical limit for the calculation. (in meters)

upper_limit: float The lower vertical limit for the calculation. (in meters)

Returns

c_mean: float Calibration constant's mean value (vertical axis).

c_sem: float Calibration constant's standard error of the mean (vertical axis).

depolarization.**particle_depolarization**(*delta_m*, *delta_v*, *molecular_backscatter*, *particle_backscatter*)

Calculate the linear particle depolarization ratio.

Parameters

delta_m: vector The linear molecular depolarization ratio.

delta_v: vector The linear volume depolarization ratio.

molecular_backscatter: vector The molecular component of the total backscatter coefficient.

particle_backscatter: vector The particle component of the total backscatter coefficient.

Returns

delta_p: vector The linear particle depolarization ratio.

Notes

The linear particle depolarization ratio is calculated by the formula:

$$\delta^p = \frac{(1+\delta^m)\delta^V \mathbf{R} - (1+\delta^V)\delta^m}{(1+\delta^m)\mathbf{R} - (1+\delta^V)}$$

References

Freudenthaler, V. et al. Depolarization ratio profiling at several wavelengths in pure Saharan dust during SAMUM 2006. Tellus, 61B, 165-179 (2008)

depolarization.**volume_depolarization_cross_parallel**(*signal_cross*, *signal_parallel*, *t_cross*, *t_parallel*, *r_cross*, *r_parallel*, *v_star*)

Calculate the linear volume depolarization ratio in a lidar system that is able to detect the cross-to-parallel depolarization ratio. The calibration factor from the delta-90 calibration is being used.

Parameters

signal_cross: vector The input vertical profile from the cross channel. Normal measurement (phi=0).

signal_parallel: vector The input vertical profile from the parallel channel. Normal measurement (phi=0).

t_cross: float Transmittance of cross component through transmitted path.

t_parallel: float Transmittance of parallel component through transmitted path.

r_cross: float Transmittance of cross component through reflected path.

r_parallel: float Transmittance of parallel component through reflected path.

v_star: float The calibration constant.

Returns

delta_v: vector The linear volume depolarization.

Notes

The linear volume depolarization ratio is calculated by the formula:

$$\delta^V = \frac{\frac{\delta^*}{V^*}T_p - R_p}{R_s - \frac{\delta^*}{V^*}T_s}$$

References

Freudenthaler, V. et al. Depolarization ratio profiling at several wavelengths in pure Saharan dust during SAMUM 2006. Tellus, 61B, 165-179 (2008)

depolarization.**volume_depolarization_cross_total**(*signal_cross*, *signal_total*, *r_cross*, *r_total*, *c*)

Calculate the linear volume depolarization ratio in a lidar system that is able to detect the cross-to-total depolarization ratio. The calibration factor from the delta-90 calibration is being used.

Parameters

signal_cross: vector The input vertical profile from the cross channel. Normal measurement (phi=0).

signal_total: vector The input vertical profile from the total channel. Normal measurement (phi=0).

r_cross: float The transmission ratio of the cross channel (Rc).

r_total: float The transmission ratio of the total channel (Rt).

c: float The calibration constant.

Returns

delta_v: vector The linear volume depolarization.

Notes

The linear volume depolarization ratio is calculated by the formula:

$$\delta^V = \frac{1 - \frac{\delta'}{C}}{\frac{\delta' R_t}{C} - R_C}$$

References

Engelmann, R. et al. The automated multiwavelength Raman polarization and water-vapor lidar Polly XT: the neXT generation. Atmos. Meas. Tech., 9, 1767-1784 (2016)

CHAPTER

SEVEN

HSRL RETRIEVAL FUNCTIONS

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

INDEX