Customer	**ESRIN**	**Document Ref**	**:**	**BAQ-MGT-TEN-SER-018**
Contract No	**4000126749/19/I-NS**	**Issue Date**	**:**	**10Jan 2020**
WP No	**3100**	**Issue**	**:**	**1.1**

Updated LIDAR aerosol profile retrieval schemes

Abstract Technical note describing the analysis software developed for the BAQUNIN elastic LIDAR system

Author Gabriele Mevi
Serco

Approval Anna Maria Iannarelli
Serco

Distribution ESA/ESRIN EOP-GMQ
BAQUNIN Leadership Team

Serco Italia SpA
Sede Operativa: Via Sciadonna, 24-26 - Frascati (Roma)
Tel: +39 06 98354400 Fax:
www.serco.com

CHANGE HISTORY

This document shall be amended by releasing a new edition of the document in its entirety. The Amendment Record Sheet below records the history and issue status of this document.

ISSUE	DATE	REASON
1.0	30 Aug 2019	First version

INTRODUCTION

This document is a description of the LIDAR Analysis Software (LAS), dedicated to the Elastic LIDAR. The software was developed as part of the Boundary-layer Air Quality-analysis Using Network of Instruments (BAQUNIN) framework at the Atmospheric Physics Laboratory (APL) of Sapienza University.

The LIDAR data analysis is not an automatic process and requires an expert operator in order to identify the clouds and aerosol layers signature on the received signal, and the best retrieval algorithm to be employed for that particular measurement.

LAS goal is to provide robust retrievals for the aerosol extinction and backscattering coefficient in many different atmospheric conditions, and a visual interface on which observe the signal, select the analysis parameters and immediately visualize the analysis results

APPLICABLE DOCUMENTS

The following is a list of documents with a direct bearing on the content of this report. Where referenced in the text, these are identified as RD.n, where 'n' is the number in the list below:

[RD.1] Project Management Plan, (TBD), latest applicable issue

ACRONYMS

Acronym	Definition
APL	Atmospheric Physics Laboratory (at Sapienza)
BAQUNIN	Boundary-layer Air Quality-analysis Using Network of INstruments
LAS	LIDAR Analysis Software

1. OVERVIEW

LIDAR analysis software (LAS) is a tool developed for the analysis of measurements collected by the LIDAR system of the BAQUNIN project. The software receives as inputs the LIDAR signal and the vertical profiles of pressure and temperature from radiosonde data. LAS create a user-friendly interface which allows visualizing the LIDAR range-corrected signal, calculating signal background, comparing the result with the synthetic profile calculated from radiosonde data and identifying the atmospheric structures (aerosol and cloud layers). LAS retrieves the optical characteristics of each layer selected and returns as outputs the aerosol and clouds optical depth, backscattering-ratio, extinction coefficient and backscattering coefficient vertical profiles. Figure 1.1 shows the block diagram describing the program main steps.

The program language used for the coding is Python3 (documentation at https://docs.python.org/3/reference) an open source language commonly used for scientific applications. The main feature of this language is its flexibility and its object oriented nature. It is distributed with several open source libraries that allow performing scientific calculations, designing graphic user interface (GUI) and handling scientific format data files. Python is compatible with Windows, MAC and Linux systems.

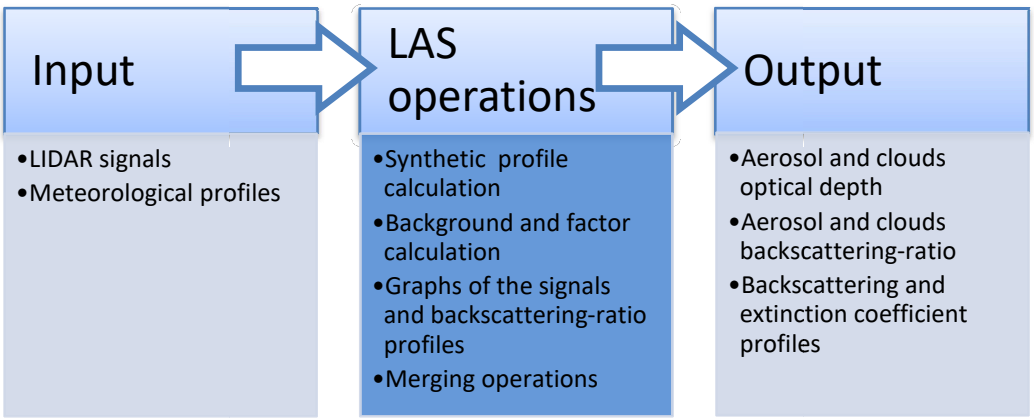

Figure 1.1: Scheme of the LAS principal characteristics

2. LAS INSTALLATION AND CONFIGURATION

In order to operate with LAS, a Python compiler should be installed on the PC. The software was developed using Anaconda (downloadable at www.anaconda.com, documentation at https://docs.anaconda.com/anaconda/), a free open source Python3 distribution, including the major part of the libraries needed to run the software, *tkinter*, *matplotlib*, *numpy* and *datetime* The links for the documentation of these libraries can be found in this document References. An additional library needs to be downloaded and installed in order to operate on netcdf file format, named *netcdf4* (documentation at https://unidata.github.io/netcdf4-python/netCDF4). This library can be done using Anaconda Prompt, with the command "*pip install netcdf4*".

LAS is composed by six different scripts (that should be in the same folder), two auxiliary codes and a configuration file, briefly described in Table 2.1.

Script Name	Description
LAS_configuration.txt	Configurations file containing the input and output directories and some graphical settings such as the character and window size.
LAS_main.py	The main script of the software, containing the operations to start the execution and the definitions of the methods that can be called by the user.
LAS_frame.py	The script containing the instructions to build the GUI.
LAS_inout.py	The script containing the input output routine.
LIDAR_info.py	This script creates a dictionary structure containing all the information about the acquisition system and the measurements.
LAS_variables.py	This script defines the variables.
LAS_computations.py	The script containing the calculation algorithms.
smooth.py	Auxiliary code. A library containing the smoothing functions.
julianday.py	Auxiliary code. A library containing functions to manage dates.

Table 2.1: the scripts composing the software with the two auxiliary codes

The **Configuration file** is an ASCII file used to properly run LAS. It contains the output and input folders paths, the library folder path and some graphical options. An example of Configuration file is shown in Appendix D. In order to correctly run LAS, the following paths should be modified on the configurations file:

- inpath: the folder containing the input files
- meteopath: the folder containing the radiosonde files
- outpath: the folder in which the output will be saved
- routinepath: the folder containing the two auxiliary codes (smooth.py and julianday.py)
- extrapath: the folder containing the measurements from other instruments useful during the analysis (if presents)

The GUI visualization parameters and graph markers can also be modified according to the user screen size and preferences. Appendix D shows an example of configuration file.

LIDAR_main.py script is the code to be executed in order to run LAS. The first part of the script imports Python3 libraries, functions and classes defined in the other scripts composing the software. The code then defines the global variables employed in all LAS architecture, as for example the output files destination folder.

LIDAR_main.py then defines a Python3 class called ***LASclass*** and all the operations that the user can execute using the analysis software as methods of the principal class (the

methods are internal functions of a class commonly used in the object-oriented programming, see Paragraph 6.7). These methods are the LAS answer to the user inputs, such as pressure of a button in the GUI. The LASclass class inherits all the classes and methods defined in the scripts LIDAR_frame.py, LIDAR_info.py and LIDAR_variables.py.

The last three command lines of this script are the command used to create an instance of the LASclass class, called LAS and run the program.

3. LIDAR RETRIEVAL ALGORITHMS

This chapter describes the algorithms used for the retrieval operations. A more complete description of the elastic LIDAR theory can be found in Kovalev and Eichinger, 2004.

3.1 The LIDAR equation

The signal $\hat{S}$ detected by the LIDAR system as function of the altitude z can be described by the LIDAR equation:

$$\hat{S}(z) = \frac{I(z)C\beta(z)T_a^2(z_0,z)^2T_m^2(z_0,z)}{z^2} + B. \quad (1)$$

The function $I(z)$ is the overlap function and it depends on the LIDAR system geometry; $I(z) < 1$ if $z < z_{over}$ and $I(z) = 1$ if $z \geq z_{over}$, where z_{over} is the overlap altitude.

C is the LIDAR calibration constant depending on laser energy, number of shoots per second, accumulation time and electronic gain. This quantity is apriori not known and need to be measured or estimated during the retrieval process.

$\beta(z) = \beta_a(z) + \beta_m(z)$ is the backscattering coefficient equal to the sum of the molecules and aerosol backscattering coefficient, respectively $\beta_m(z)$ and $\beta_a(z)$.

z_0 is the instrument altitude from sea level while B is the signal background.

$T_m^2(z_0,z)$ and $T_a^2(z_0,z)$ are the profile transmission due to molecules and aerosol respectively, and defined according to:

$$T_m^2(z_0,z) = \exp\left(-2\int_{z_0}^{z}\alpha_m(z)\,dz\right) \quad (2)$$

$$T_a^2(z_0,z) = \exp\left(-2\int_{z_0}^{z}\alpha_a(z)\,dz\right) \quad (3)$$

$\alpha_m(z)$ and $\alpha_a(z)$ are the extinction coefficient of molecules and aerosol respectively.

The Equation (1) can be rearranged subtracting the background and multiplying by the square of the altitude to produce the range corrected signal $RCS = z^2(\hat{S} - B)$.

$$RCS(z) = C\beta(z)T_a^2(z_0,z)T_m^2(z_0,z). \quad (4)$$

Equation (4) is called range corrected signal equation. In what follows the equations will be analyzed in the profile region in which $z > z_{over}$, where $I(z > z_{over}) = 1$.

A useful quantity is the **molecular range corrected signal (RCS_m)**:

$$RCS_m(z) = \beta_m(z)T_m^2(z_0,z) \quad (5)$$

RCS_m and the molecular backscattering and extinction coefficients can be calculated knowing the temperature and pressure vertical profile from the meteorological file, through the Rayleigh scattering theory.

In order to resolve Equation (4), a linear relation between aerosol backscattering and extinction coefficient is assumed:

$$\alpha_a = LR\beta_a, \quad (6)$$

where LR is defined **LIDAR Ratio,** a quantity that depends on aerosol composition, diameter and shape.

Another important function is the **backscatter ratio** R between total backscattering coefficient and molecular backscattering coefficient:

$$R(z) = \frac{\beta_a(z)+\beta_m(z)}{\beta_m(z)} \quad (7)$$

$$\beta_a(z) = (R(z) - 1)\beta_m \quad (8)$$

The goal of the retrieval process is to obtain C, β_a and α_a that are unknown.

3.2 LAS retrieval scheme

Figure 3.1 displays the retrieval scheme; it is composed by three steps: the **Signal properties characterization**, the **Retrieval type selection** and the **Definition of aerosol and cloud layers**.

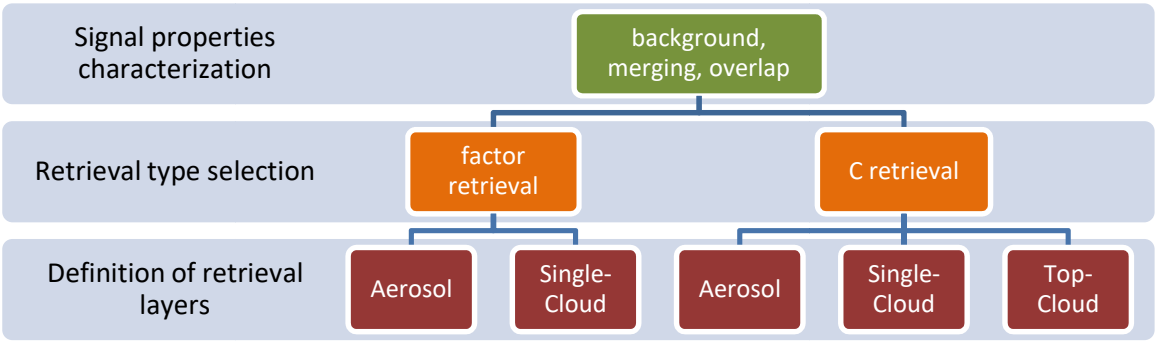

Figure 3.1: scheme of the retrieval algorithm.

The **Signal properties characterization** step involves the operations to characterize the signal properties, such as obtain the background value B (Paragraph 5.4), select the overlap and altitude scale layer (Paragraph 3.3 and 5.5), and eventually merge two signals (Paragraph 5.3).

Afterwards the retrieval scheme has to be chosen. LAS calculates the backscattering and extinction coefficients of the aerosol and clouds layers using two different retrieval algorithms. The first algorithm called **factor retrieval** can be employed if the followings conditions are verified:

1. there are not clouds inside or at the top of the ground-based aerosol layer;
2. there is a region where the aerosol contribution to the scattering is negligible. This region is called **factor layer** and it is used to estimate the **factor value** (see the Paragraph 3.4).

Adding the factor layer to the profile (Paragraph 5.6) selects the factor retrieval scheme.

The **C retrieval** can be employed if it is not possible to identify the factor layer or if there are clouds inside or at the top of the ground-based aerosol layer.

Figure 3.2 shows an example of well distinguished layers, where the factor layer is marked by the two orange vertical lines, an aerosol layer by purple lines and a cloud layer by cyan lines. Figure 3.3 shows an example of cloud inside an aerosol layer in which the cloud bottom border cannot be easily identified.

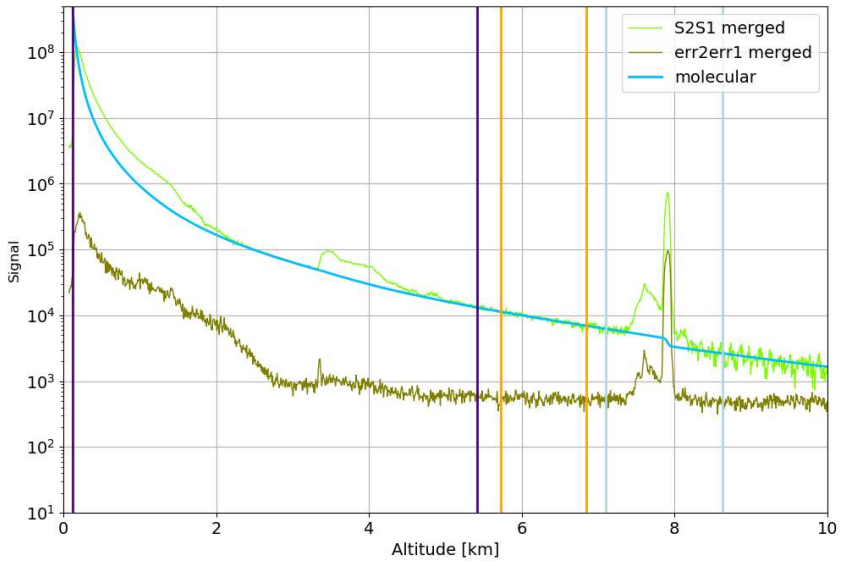

Figure 3.2: an example of well distinguished layers.

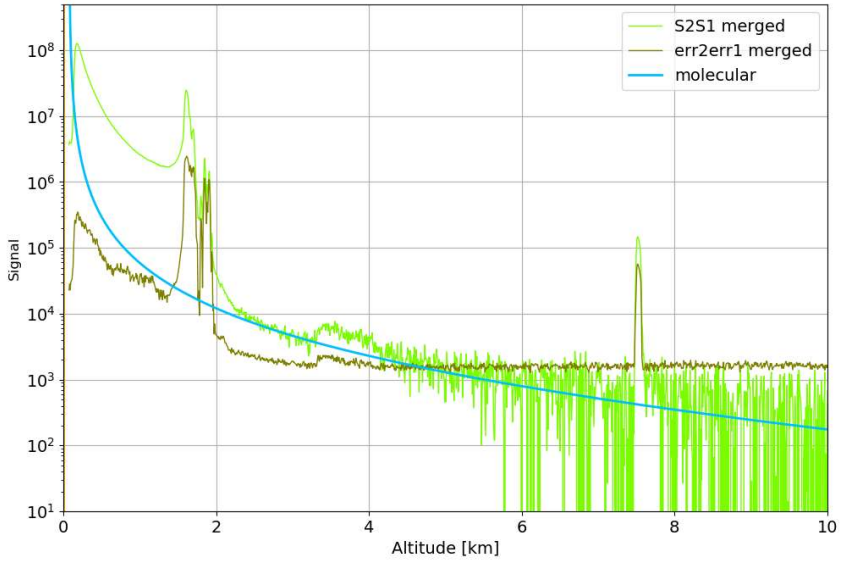

Figure 3.3: the cloud layer and the aerosol layer cannot be clearly distinguished.

The last step involves the definition of the retrieval layers (Paragraph 5.8). Three layer types can be actually selected: the aerosol layer, the single-cloud layer and the top-cloud layer. Top-cloud layers scenario can be analyzed only with the C retrieval scheme.

An **aerosol layer** is a profile region of which it is possible to apriori estimate the LIDAR Ratio, so LR is a calculation Input and is provided by the user (LR is considered to have 10% relative uncertainty).

A **single-cloud layer** is a cloudy region surrounded by regions where the aerosol concentration is negligible. This region is usually characterized by strong absorption due

to water droplet or ice crystals that can be measured observing the signal decay above the layer. The cloud optical depth and the LR are carried out from the retrieval algorithm. An example of single-cloud layer is marked with cyan lines in Figure 3.2.

A **top-cloud** layer is a cloudy region that tops a ground-based aerosol layer; above the cloud a factor layer is present. In this case is not possible to use a single LR for the aerosol layer and the cloud. Figure 3.3 shows a top-cloud next to an aerosol layer.

3.3 Backscattering Ratio extrapolation

The backscattering ratio R below the overlap altitude cannot be correctly calculated due to the poor knowledge of the overlap function; however the region below z_{over} can have an optical depth calculation contribution not negligible. In order to provide a better estimation of the profile optical depth τ, $R_{c/f}(z < z_{over})$ can be extrapolated according to

$$R_{c/f}(z < z_{over}) = R_{c/f}(z_{over})\exp\left(\frac{z_{over}-z}{H}\right) \quad (9)$$

Where H is an altitude scale.

3.4 The factor retrieval

The factor retrieval assumes the presence of **factor layer** (Z_{fac}) that is selected by the user.

The ratio between S and S_m in this region is called **factor** (f)

$$f \equiv \frac{RCS(z\in Z_{fac})}{RCS_m(z\in Z_{fac})} = CT_a^2(z_0, z_m), \quad (10)$$

where z_m is the factor layer bottom height. LAS calculates f averaging the values of S/S_m on the factor layer.

Outside the factor layer the ratio between S and S_m can be described using Equations (3) and (7) as

$$\frac{RCS(z)}{RCS_m(z)} = R(z)CT_a^2(z_0, z_m)T_a^2(z_m, z) \quad (11)$$

Defining the unattenuated backscatter ratio

$$R_f(z) \equiv \frac{RCS(z)}{f\,RCS_m(z)}, \quad (12)$$

Equation (11) can be rewritten as:

$$R_f(z) = R(z)T_a^2(z_m, z) \quad (13)$$

Equation (13) can be resolved with an iterative procedure; adding an aerosol or a cloud layer to the calculation, the user selects the profile region in which the equation will be solved in order to calculate the backscattering and extinction coefficients. As displayed in Figure 3.1 the algorithm changes depending of the layer type added by the user.

Note: in order to correctly operate, the factor retrieval scheme needs the knowledge of the extinction coefficient of the region **between** the factor layer and the new added layer: this implies that if more than one layer is present, the user has to add first the layers next to the factor layer.

After the layers have been resolved, LAS calculates the calibration constant C of the analyzed profile using Equation (14)

$$C = \frac{f}{T_a^2(z_0, z_m)} \tag{14}$$

This value will be updated every time a new layer is added to the calculation.

LAS assigns a default value of $R(z)$ even on the profile region with no layer type, equal to:

$$R(z) = \frac{R_f(z)}{T_a^2(z_m, z)}. \tag{15}$$

3.4.1 Aerosol layer (factor retrieval)

The aerosol algorithm requires the knowledge of LR value, see Equation (6). This value is inserted by the user. In what follows, the iteration steps will be represented by the apex.

As first step the R value is initialized to R_f

$$R^0 = R_f \tag{16}$$

β_a^0 and α_a^0 profiles are calculated using Equation (8) and Equation (6); $[T_a^2(z_m, z)]^0$ is calculated using the α_a^0 profiles. The procedure continues calculating:

$$R^{i+1}(z) = \frac{R_f(z)}{[T_a^2(z_m, z)]^i} \tag{17}$$

and using R^{i+1} to calculate β_a^{i+1}, α_a^{i+1} and $[T_a^2(z_m, z)]^{i+1}$.

The user can select an integral or a punctual convergence mode through the convergence button. If the integral convergence is selected the iteration goes on until

$$\int_{z_{bot}}^{z_{top}} \alpha_a^{i+1}(z)dz - \int_{z_{bot}}^{z_{top}} \alpha_a^{i}(z)dz < \varepsilon. \tag{18}$$

If the punctual convergence is selected, the stop condition is calculated separately at each altitude:

$$\alpha^i(z) - \alpha^{i+1}(z) < \varepsilon. \tag{19}$$

3.4.2 Single-cloud layer (factor retrieval)

A single-cloud layer is a region whose LR is not apriori known and it is surrounded by no aerosol regions.

The signal at z_{bot} and $z_{top}$is

$$RCS(z_{bot}) = C\beta(z_{bot})T_m^2(z_0, z_{bot})T_a^2(z_0, z_{\boldsymbol{bot}}) \tag{20}$$

$$RCS(z_{\boldsymbol{top}}) = C\beta(z_{top})T_m^2(z_0, z_{top})T_a^2(z_0, z_{bot})T_a^2(z_{bot}, z_{top}) \tag{21}$$

The two equations can be rewritten as

$$R_f(z_{bot}) = R(z_{bot})T_a^2(z_0, z_{bot}) \tag{22}$$

$$R_f(z_{top}) = R(z_{top})T_a^2(z_0, z_{bot})T_a^2(z_{bot}, z_{\boldsymbol{top}}) \tag{23}$$

The ratio between Equations (23) and (22) is

$$\frac{R_f(z_{top})}{R_f(z_{bot})} = \frac{R(z_{top})}{R(z_{bot})}T_a^2(z_{bot}, z_{top}) = \frac{R(z_{top})}{R(z_{bot})}e^{-2\tau_c} \tag{24}$$

Where $\tau_c = \int_{z_{bot}}^{z_{top}} \alpha_a(z)dz$ is the cloud optical depth. Since no aerosol is present at the cloud borders $R(z_{bot}) = R(z_{top}) = 1$ and the Equation (24) allows the calculation of the cloud optical depth. In order to minimize the uncertainty due to signal oscillations $R_f(z_{bot})$ and $R_f(z_{top})$ are calculated averaging together ten altitude bins respectively below z_{bot} or above z_{top}.

Note: the value of LR is an output of this type of calculation.

As first step the R value is initialized to R_f and L to the value of 10 sr.

$$R^0(z) = R_f(z),\ LR^0 = 10\ sr \tag{25}$$

$\beta_a^i(z)$ is calculated using Equation (8) from $R^i(z)$, while L^i is computed using Equation (26)

$$LR^i = \frac{\tau_c}{\int_{z_{bot}}^{z_{top}} \beta_a^i(z)dz} \tag{26}$$

The $\alpha_a^i(z)$ is calculated by:

$$\alpha_a^i(z) = LR\beta_a^i(z) \tag{27}$$

$\alpha_a^i(z)$ is used to calculate $[T_a^2(z_m, z)]^i$and $R^{i+1}(z)$

$$R^{i+1}(z) = \frac{R_f(z)}{[T_a^2(z_m,z)]^i} \tag{28}$$

The iteration stops when

$$LR^{i+1} - LR^i < \varepsilon \tag{29}$$

3.5 The C retrieval

The C retrieval algorithm can be used when the LIDAR calibration constant have been evaluated independently with respect of the analyzed profiles. Actually this is possible using the *factor_fit* method through a linear interpolation between the C values of the other profiles.

Using the Equations (4) and (5) it is possible to obtain Equation (30)

$$R_c(z) = R(z)T_a^2(z_0, z), \tag{30}$$

where

$$R_c(z) = \frac{RCS(z)}{C*RCS_m(z)}. \tag{31}$$

Note: in order to correctly operate, the C retrieval algorithm needs the knowledge of the extinction coefficient of the region **below** a new added layer: this implies that if more than one layer is present, the user has to add first the lower layers, in order of increasing altitude.

LAS assigns a default value of R even on the profile region with no layer type, equal to:

$$R(z) = \frac{R_c(z)}{T_a^2(z_0,z)}. \tag{32}$$

3.5.1 Aerosol layer (C retrieval)

The aerosol algorithm requires the knowledge of L value, see Equation (6). This value is inserted by the user. In what follows the iteration step will be represented by the apex.

As first step the R value is initialized to R_c

$$R^0 = R_c \tag{33}$$

β_a^0 and α_a^0 profiles are calculated using Equation (8) and Equation (6). $[T_a^2(z_0,z)]^0$ is calculated using the α_a^0 profiles.

The procedure continues calculating

$$R^{i+1}(z) = \frac{R_c(z)}{[T_a^2(z_0,z)]^i} \tag{34}$$

and using R^{i+1} to calculate β_a^{i+1}, α_a^{i+1} and $[T_a^2(z_0,z)]^{i+1}$.

The stop condition is calculated separately at each altitude:

$$\alpha^i(z) - \alpha^{i+1}(z) < \varepsilon. \tag{35}$$

3.5.2 Single-cloud layer (C retrieval)

As described above, the layer optical depth can be calculated by means of Equation (24) when no aerosol zones are presents at the layer borders.

Note: the value of LR is an output of the calculation.

As first step the R value is initialized to R_c and LR to the value of 10 sr.

$$R^0(z) = R_c(z),\ LR^0 = 10\ sr \tag{36}$$

$\beta_a^i(z)$ is calculated using Equation (8) from $R^i(z)$, while LR^i is computed using Equation (26)

$$LR^i = \frac{\tau_c}{\int_{z_{bot}}^{z_{top}} \beta_a^i(z)dz} \tag{37}$$

$\alpha_a^i(z)$ is calculated by:

$$\alpha_a^i(z) = LR * \beta_a^i(z) \tag{38}$$

$\alpha_a^i(z)$ is used to calculate $[T_a^2(z_0,z)]^i$and $R^{i+1}(z)$

$$R^{i+1}(z) = \frac{R_c(z)}{[T_a^2(z_0,z)]^i} \tag{39}$$

The iteration stops when

$$LR^{i+1} - LR^i < \varepsilon \tag{40}$$

3.5.3 Top-cloud layer (C retrieval)

A single-cloud layer is a profile region whose LR is not apriori known and it is characterized by no aerosol above its top border.

The top-cloud layer optical depth can be calculated using Equation (24) with $R(z_{top}) = 1$.

Note: in a correct analysis, the region below the bottom border has been resolved **before** adding the top-cloud layer; in this way the value of $R(z_{bot})$ is known (LAS averages together the R values of 5 bins below the bottom border in order to reduce the Ratio oscillations).

The iterative algorithm is the same described in the previous paragraph.

3.6 Molecular signal update

LAS shows on the principal graph V_m, the range corrected signal that should be acquired in absence of aerosol ($\alpha_m \equiv 0, \beta_m \equiv 0$).This quantity is not used directly in the calculations but it is useful to correctly identify the aerosol and cloud layers and the factor layers. It takes into account the molecular emission and the total extinction coefficient of the profile

$$V_m = C * RCS_m T_a^2(z_0, z) \tag{41}$$

C and transmission values on Equation (41) are apriori not known. V_m is so updated with every calculation according to Equation (41), if a C retrieval has been executed or according to Equation (42) if a factor calculation has been executed.

$$V_m = f * RCS_m T_a^2(z_m, z) \tag{42}$$

3.7 Uncertainty calculation

The uncertainties of the variables are saved in the LAS output file and represented in the graphs. This chapter describes how the uncertainties are calculated; the notation $\sigma(a)$ refers to the uncertainty of the quantity a.

In what follows the uncertainties of α, β, R will be classified following the guidelines provided by *Evaluation of measurement data—Guide to the expression of uncertainty in measurement* (2008). The obtained systematic uncertainties refer to the non-random uncertainties type described in the cited guidelines and will be marked by the apex “sys”, instead the random uncertainties refers to the random uncertainty type described in the same document and marked by the apex “ran”. The total uncertainty will be characterized by no apex.

Note: if a smooth is applied on signal or Ratio, the correspondent uncertainty will decrease of a factor $\sqrt{n}$, with n the number of smooth points.

3.7.1 Signal, background and molecular signal uncertainty

$RCS(z)$ is the sum of all the counts measured by the front-end electronics during the accumulation time. The total signal uncertainty $\sigma(RCS)$ takes into account the uncertainty introduced by front-end electronics described by Russell et al. (1978), the signal oscillations due to atmospheric variations during the accumulation process and the background value uncertainty. An estimation of the first two contributions is provided by standard deviation of the signal during the accumulation time multiplied by a factor $\sqrt{n_{shot}}$, where n_{shot} is the accumulated laser shoots number. The background uncertainty is estimated by the background calculation methods. $\sigma(B)$ calculated using the average on the last 100 points of the signal is equal to the standard deviation divided by the square root of the points number.

$\sigma(B)$ calculated using the *background_fit* method is equal to the linear fit parameters uncertainty described in Equation (43).

$$\sigma(B) = \sqrt{\frac{\sum_i \left[\sigma\left(RCS(z_i)\right)^2 RCS_m(z_i)^2\right]}{N\sum RCS_m(z_i)^2 - \left[\sum_i RCS_m(z_i)\right]^2}} \tag{43}$$

If the background value is modified using a multiplicative parameter by the user, the background uncertainty will change proportionally. The total signal uncertainty is so equal to:

$$\sigma(S) = \sqrt{\sigma_p(RCS)^2 + \sigma(B)^2} \tag{44}$$

The molecular signal uncertainty and molecular backscattering uncertainty depends on meteorological profile uncertainties and the distance between instrument and radiosonde station. For the Rome LIDAR these quantities have been estimated to have 3% relative uncertainties.

3.7.2 Factor uncertainty

If the factor procedure is selected, the factor uncertainty $\sigma(f)$ is calculated as the standard deviation of the ratio between signal and molecular signal on the factor layer, divided by the square root of the number of altitude bins of the factor layer.

If the factor is calculated using the *background_fit* method $\sigma(f)$ is the fit slope uncertainty calculated using Equation (45):

$$\sigma(f) = \sqrt{\frac{\sum_i \left[\sigma(RCS(z_i))^2\right]}{N\sum RCS_m(z_i)^2 - \left[\sum_i RC_m(z_i)\right]^2}} \tag{45}$$

3.7.3 C and transmission uncertainty

The uncertainty of C, is calculated as an output of the factor retrieval and it depends on the factor uncertainty and transmission uncertainty, as displayed in Equation (14). It can be evaluated using the uncertainty statistical propagation theory. If C is interpolated using the *factor_fit* method $\sigma(C)$ is a linear interpolation between the $\sigma(C)$ of the profiles analyzed using the factor retrieval.

The transmission uncertainty is estimated as:

$$\sigma\left(T_a^2(z_0,z_m)\right) = \frac{1}{2}\left|\exp\left(-2\int_{z_0}^{z_m}\left(\alpha_a(z)+\left|\sigma\left(\alpha_a(z)\right)\right|\right)dz\right) - \exp\left(-2\int_{z_0}^{z_m}\left(\alpha_a(z)-\left|\sigma\left(\alpha_a(z)\right)\right|\right)dz\right)\right| \tag{46}$$

where $\sigma(\alpha_a(z))$ is the extinction coefficient uncertainty (see below).

3.7.4 Retrieval uncertainty calculation

In order to estimate the uncertainty of the quantities calculated using the factor/C retrieval, $\sigma(R_f)/\sigma(R_c)$ needs to be estimated first. These quantities are calculated using the uncertainty statistical propagation theory from the $\sigma(f)/\sigma(C)$, $\sigma(S)$ and $\sigma(RCS_m)$.

$$\sigma^{sys}\left(R_{f/c}\right) = R_{f/c}\frac{\sigma\left(RCS_m\right)}{RCS_m} \tag{47}$$

$$\sigma^{ran}\left(R_{f/c}\right) = R_{f/c}\sqrt{\left(\frac{\sigma(c/f)}{c/f}\right)^2 + \left(\frac{\sigma(RCS)}{RCS}\right)^2} \tag{48}$$

$$\sigma\left(R_{f/c}\right) = R_{f/c}\sqrt{\left(\frac{\sigma(c/f)}{c/f}\right)^2 + \left(\frac{\sigma(RCS)}{RCS}\right)^2 + \left(\frac{\sigma(RCS_m)}{RCS_m}\right)^2} \tag{49}$$

Aerosol retrieval

A 10% uncertainty is assumed affecting the LR used for the calculation. In order to evaluate $\sigma_{LR}(R)$, the contribution of $\sigma(LR)$ to $\sigma(R)$, the iterative algorithm is executed with $LR \pm |\sigma(LR)|$ as inputs, producing R^+ and R^- respectively as outputs. This leads to the calculation of:

$$\sigma_{LR}(R) = \frac{1}{2}|R^+ - R^-| \tag{50}$$

The total uncertainty on the backscattering ratio is then:

$$\sigma(R) = \sqrt{[\sigma_{LR}(R)]^2 + \left[R\frac{\sigma(R_{f/c})}{R_{f/c}}\right]^2}, \tag{51}$$

$$\sigma^{ran}\left(R\right) = R\frac{\sigma^{ran}\left(R_{f/c}\right)}{R_{f/c}} \tag{52}$$

$$\sigma^{sys}\left(R\right) = \sqrt{\left[\sigma_{LR}\left(R\right)\right]^2 + \left[R\frac{\sigma^{sys}\left(R_{f/c}\right)}{R_{f/c}}\right]^2} \tag{53}$$

where the second term on the right side changes if factor or C retrieval has been used.

$\sigma(\beta)$ and $\sigma(\alpha)$ are evaluated using the uncertainty statistical propagation theory from $\sigma(R)$ and $\sigma(LR)$ (note that $\sigma(\beta_m)$ is included in $\sigma(R)$ due to $\sigma(RCS_m)$ contribution:

$$\sigma(\beta_a) = (\sigma(R)\beta_m), \tag{54}$$

$$\sigma^{sys}\left(\beta_a\right) = \sigma^{sys}\left(R\right)\beta_m \tag{55}$$

$$\sigma^{ran}\left(\beta_a\right) = \sigma^{ran}\left(R\right)\beta_m \tag{56}$$

$$\sigma(\alpha_a) = \sqrt{(\sigma(LR)\beta_a)^2 + (\sigma(\beta_a)LR)^2}. \tag{57}$$

$$\sigma^{sys}\left(\alpha_a\right) = \sqrt{\left(\sigma\left(LR\right)\beta_a\right)^2 + \left(\sigma^{sys}\left(\beta_a\right)LR\right)^2} \tag{58}$$

$$\sigma^{ran}\left(\alpha_a\right) = \sigma^{ran}\left(\beta_a\right)LR \tag{59}$$

The layer optical depth uncertainty $\sigma(\tau_a)$ is calculated accordingly to:

$$\sigma\left(\tau_a\right) = \int_{z_{bot}}^{z_{top}} \sigma\left(\alpha_a\left(z\right)\right)dz \tag{60}$$

Cloud retrieval

The Cloud retrieval is based on the estimation of the cloud layer optical depth τ_c using Equation (24), both in "single-cloud" and "top-cloud" modes. $\sigma\left(R(z_{top})\right)$, $\sigma\left(R_f(z_{top})\right)$, $\sigma\left(R_f(z_{bot})\right)$ if factor retrieval has been used, or $\sigma\left(R_c(z_{top})\right)$ and $\sigma\left(R_c(z_{bot})\right)$ if C retrieval has been used used, are estimated by the standard deviation on a region of 5 bins

below/above z_{bot}/z_{top} divided by the square root of the bins number. $\sigma(\tau_c)$ is calculated then using the uncertainty statistical propagation theory.

In order to evaluate $\sigma_\tau(R)$, the contribution of $\sigma(\tau_c)$ to $\sigma(R)$, the iterative algorithm is executed with $\tau_c \pm |\sigma(\tau_c)|$ as inputs, producing R^+ and R^- respectively as outputs.

$$\sigma_\tau(R) = \frac{1}{2}\left|R^+ - R^-\right| \tag{61}$$

The total uncertainty on the backscattering ratio is then:

$$\sigma(R) = \sqrt{\left[\sigma_\tau(R)\right]^2 + \left[R\frac{\sigma\left(R_{f/c}\right)}{R_{f/c}}\right]^2}\ , \tag{62}$$

$$\sigma^{ran}(R) = R\frac{\sigma^{ran}\left(R_{f/c}\right)}{R_{f/c}} \tag{63}$$

$$\sigma^{sys}(R) = \sqrt{\left[\sigma_\tau(R)\right]^2 + \left[R\frac{\sigma^{sys}\left(R_{f/c}\right)}{R_{f/c}}\right]^2} \tag{64}$$

where the second term on the right side in Equations (62) and (64) depends if factor or C retrieval has been used.

$\sigma(\beta)$ and $\sigma(\alpha)$ are calculated according to the previous paragraph, while $\sigma(LR)$ is:

$$\sigma(LR) = \sqrt{\left(\frac{\sigma(\tau_c)}{\int_{z_{bot}}^{z_{top}}\beta_a(z)dz}\right)^2 + \left(\frac{\int_{z_{bot}}^{z_{top}}\sigma(\beta_a(z))dz}{\int_{z_{bot}}^{z_{top}}\beta_a(z)dz}LR\right)^2} \tag{65}$$

4. LAS GRAPHICAL INTERFACE

LAS GUI (graphical user interface) allows the user to interact with the analysis software and to visualize the results of each operation (Figure 4.1). This chapter describes the different buttons and graphic elements with their function.

Figure 4.1 shows the GUI; the different parts of the window are marked with letters and described below.

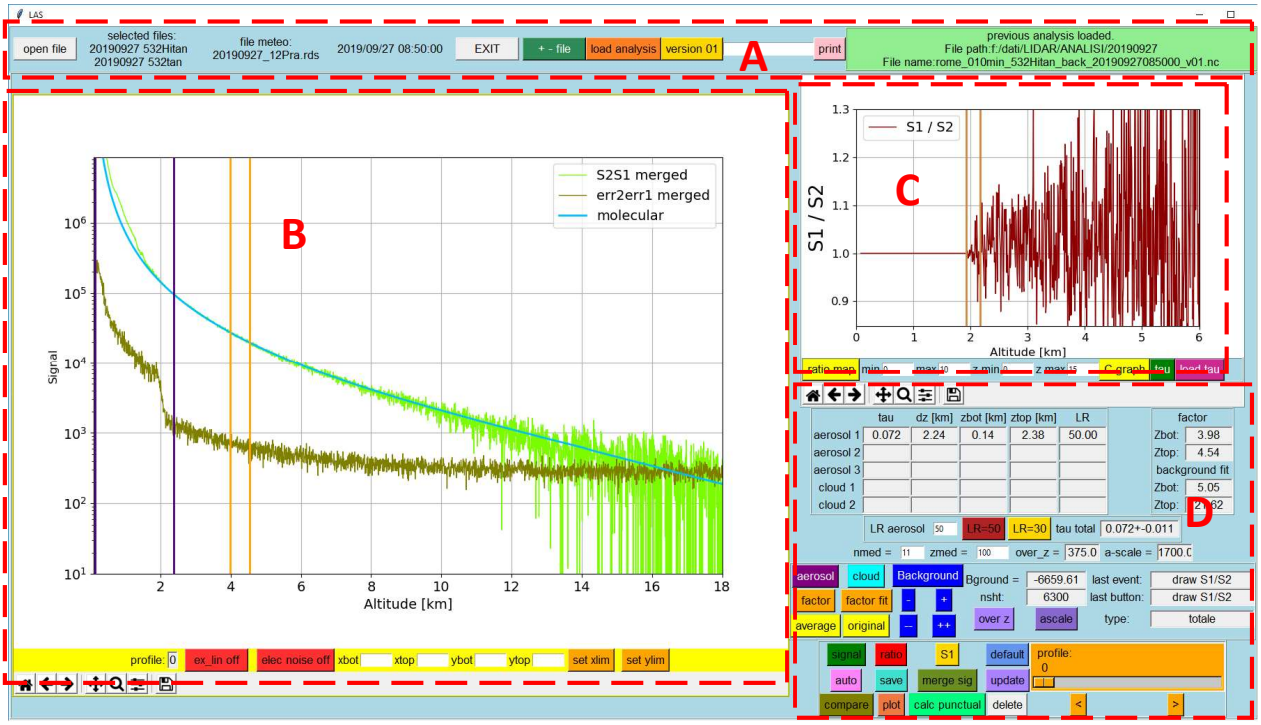

Figure 4.1: The visual interface of LAS.

A. The top part of the window contains the buttons used to handle the input-output operations and some information about the files analyzed. This part of the GUI is defined **info frame.**
B. The left part of the window hosts the signal graph; the x-axis represents the signal altitude while the y-axis the signal intensity in term of LIDAR counts. This graph is called **principal graph** and allows the user to select the profile region in which perform the various operations. Several buttons are used to zoom in the graphic or to shift the x or y-axis. It is also possible to save a single graph as a .png file through the save graph button, represented as a small floppy. This part of the GUI is defined **principal graph frame.**
C. The right part of the window presents a smaller graph in which can be displayed the ratio between two loaded signals, the backscattering ratio colour map of the analyzed profiles and the graph of LIDAR calibration constant as function of time. This part of the GUI is defined **auxiliary graph frame.**
D. The bottom right of the window hosts several buttons and indicators, used to interact with the software and observe the computation results. The functions of the various buttons will be described later in this document. This part of the GUI is defined **operational frame.**

The size of the different objects may change in order to adapt to different screen resolution. In what follows the GUI elements will be described shortly; for a more detailed description of LAS operations see the Chapter 5.

In what follows there is a description of the buttons, indicators and entries of LAS.

The info frame

- **Open file button:** open a window in which the user can select a signal file to analyze.
- **File names indicators:** the names of the signal files loaded.
- **Meteorological file name indicator:** the name of the meteorological file used to calculate the molecular signal.
- **Date-time indicator:** the date and time of current visualized profile start measurement.
- **Exit button**: used to terminate the program.
- **Additional file button:** this button opens a pin-up menu that allows the user to load or remove from memory additional signal files. It is used in the merging and synchronization operations.
- **Load analysis button:** this button opens a pin-up menu that allows the user to load the results and parameters of a previously saved analysis, update the calculation or save all the profiles contained in a signal file.
- **Version button:** this button open a pin-up menu that allows to change the output file name and version; because the output file name is automatically assigned this button is used to create new output files without overwriting the previous ones.
- **Command entry and print button:** these elements are used as debugging tools. Typing in the entry the name of some LAS variable and pressing **print,** the software will print on the command shell its value. For detailed information about LAS data structure see Chapter 6.
- **Message box:** this box is used to visualize LAS messages and warnings.

The principal graph frame

- **Principal graph:** signal, molecular signal, backscattering ratio, backscattering and absorption coefficients of the visualized profile are displayed here. Other analysis parameters such as bottom and top altitudes of the aerosol and cloud layers can be visualized here by vertical lines.
- **Profile indicator:** the number of the visualized profile.
- **Electronic noise button:** pressing this button visualizes the level of noise due to electronic affecting the signal on the principal graph.
- **Extra lines button:** this button shows/hides some indicators such as the merging layer bottom and top altitudes on the principal graph.
- **Graph limits entries and buttons:** entries and buttons to set the minimum and maximum values of the axis.
- **Matplotlib graph interaction buttons:** a set of Python built in buttons to zoom, shift, revert to previous visualization and save the graph as a jpeg or png image.

The auxiliary graph frame

- **The auxiliary graph:** this graph is useful to visualize useful information about the analysis, such as Calibration constant and total optical depth over time, the ratio between two loaded signals (for the merging operation), the calculated backscattering ratio map as function of altitude and time.
- **Ratio map button and entries:** this button shows the backscattering Ratio as function of altitude and time on the auxiliary graph. The entries can be used to set the minimum and maximum value of the colour-bar and of the altitude axis.
- **C graph button:** it draws the value of the Calibration constant as function of time.
- **Tau button:** it draws the value of optical depth estimated from the analysis as function of time.
- **Tau load button:** it allows the user to load the optical depth values measured by other instruments (actually CIMEL or PREDE-POM) in order to compare with the

analysis results. See Appendix E for an example of CIMEL or PREDE-POM optical depth files.

- **Matplotlib graph interaction buttons:** see above.

The operational frame

- **Layers indicators**: information regarding the layers added to the calculations.
- **LR aerosol entry:** the LIDAR-Ratio value currently used for the aerosol layer calculation.
- **LR 30 and 50 buttons:** these buttons quickly set the LR equal to 30 or 50.
- **Tau total indicator:** the optical depth of the visualized profile (equal to the sum of the optical depth of each aerosol and cloud layer).
- **Average point entry "nmed":** used to set the number of points on which perform the smoothing average.
- **Average minimum altitude entry "zmed":** used to set the altitude above which smooth on signal or backscattering ratio.
- **Overlap altitude indicator:** the overlap altitude z_{over} used in the calculation.
- **Altitude scale "a-scale":** the altitude scale H used in the calculation to extrapolate R below z_{over}.
- **Aerosol button:** this button opens a pin-up menu that allows adding or removing an aerosol layer to the calculation.
- **Cloud button**: this button opens a pin-up menu that allows adding or removing a cloud layer to the calculation.
- **Background button**: this button opens a pin-up menu that allows calculating the background value B, using different methods.
- **Background "+" or "-" buttons:** manually increase or decrease the B value.
- **Factor button**: this button allows adding the factor layer for the calculation of f.
- **Factor fit button**: this button allows estimating the calibration constant C.
- **Average button**: perform a smoothing average of *nmed* points on signal or R.
- **Original button:** undo the smoothing process.
- **Over z button:** select z_{over} clicking on the principal graph.
- **Altitude scale button**: select the altitude scale H clicking on the principal graph.
- **Background indicator:** the value of B.
- **Shoots number indicator:** the number of laser shoots accumulated for the measurement of the visualized profile.
- **Last event indicator:** the last action executed.
- **Last button indicator:** last button pressed.
- **Calculation type indicator:** type of calculation (elastic, Raman). Actually not used.
- **Signal button:** visualize the signal on the principal graph.
- **Ratio button:** visualize R on the principal graph.
- **Signal selector button:** select the signal on which perform the background calculation.
- **Default button:** save the visualized profile analysis parameters in the Default dictionary.
- **Auto button:** analyze the visualized profile using the parameters saved in the Default dictionary.
- **Save button:** save the visualized profile results in the output file.
- **Merge signal button:** open a pin-up menu with the merging and synchronization operations.
- **Update button:** update the analysis of the current profile.
- **Compare button**: compare the LAS and other software results. Used during software development, it will be disabled in the future versions.
- **Plot button:** plot α, β, or signal derivative on the principal graph.
- **Convergence mode:** select the convergence mode for the iterative calculation (punctual or integral convergence).

- **Delete analysis button:** delete the visualized profile analysis results and parameters.
- **Profile selector:** move the cursor or use the buttons to switch between the profiles.

5. LAS OPERATIONS

This section describes the operations that can be executed in order to analyze a signal profile, summarized in Figure 5.1. The green, orange and red panes refer to the first, second and third analysis step defined in Paragraph 3.2, respectively Signal properties characterization, Retrieval type selection and Definition of retrieval layers, while the blue panes refer to Input Output operations. The green panes operations are optional. These operations cycle, apart from the inputs file load, have to be repeated for each profile.

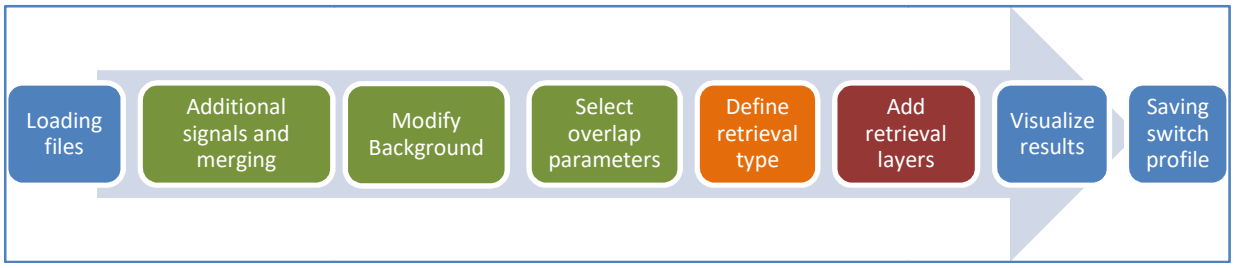

Figure 5.1: The operative steps of LAS

5.1 Loading the main file

The open file button at the top left of the LAS window is used to load a measurement file. A mouse click on the button will open a box in which the user can select the input signal file; after the selection, a second box will appear to allow the user to select the meteorological file containing the information about atmospheric pressure and temperature, mandatory to analyze the signal (see Appendix B).

If the load operation was successful, the signal file is showed on the left graph, together with the signal uncertainty and the molecular signal computed according to the temperature and pressure profiles, as displayed in Figure 5.2.

If more than a single profile is saved on the signal file, the user can shift between the saved profiles moving the **orange cursor** of the profile selector of the Panel D (for panels numbering, refer to Figure 4.1), and see there the number of the profile selected; alternatively, the user can use the **keyboard command Page Up and Page Down**. The **signal index** identifies the different profiles (see Paragraph 5.11 for details).

The indicators in Panel A will show the signal and meteorological file names and the date/time referred to the visualized profile. The signal file loaded using this procedure is defined in this document as the **main file** of the analysis.

At this stage LAS calculates the signal background automatically as the average of the signal last 100 points. This first background estimation is subtracted to the original signal. A new background computation can be executed through the **Background** command (see paragraph 5.4).

During the load process LAS calculates the **molecular signal** V_m (see paragraph 3.6 for detailed molecular signal calculation). The signal received is proportional to the LIDAR calibration constant C, which is a priori not known. At this stage, therefore the molecular signal showed in the graph is just a poor estimation. In order to estimate it more accurately the user can calculate the **LIDAR factor** f or to estimate the calibration constant by the factor fit operation (see paragraphs 5.6 and 5.7), using the **factor or factor fit** buttons. V_m is also updated after the calculation of each aerosol or cloud layer (Paragraph 3.6).

LAS will show the signal file with its uncertainty and the molecular signal on the main graph (Figure 5.2).

The switch button "**electronic noise on/off**", lower part of Box B in Figure 4.1, enables/disables the visualization of the estimated electronic noise intensity on the principal graph.

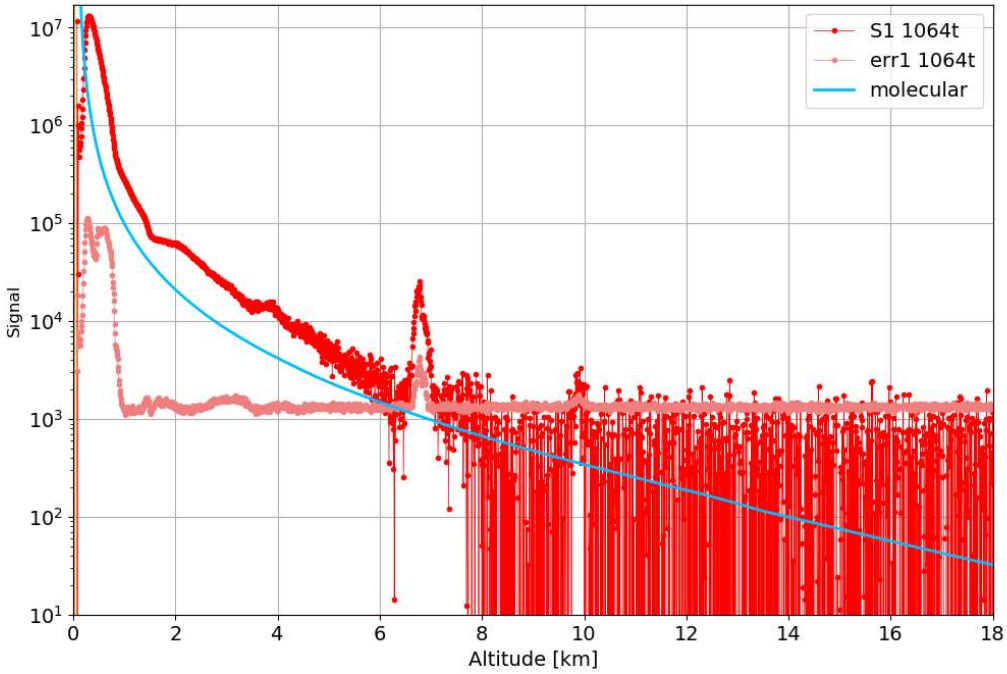

Figure 5.2: The main graph after a successful load of the signal and meteorological files. The RCS is in red, the signal uncertainty in pink and the molecular range corrected signal in light blue. Note: the colours change accordingly to the showed channel (1064 nm in this picture).

If the loaded signal was acquired through photo-counts, LAS applies to it a correction taking into account the front-end electronics dead-time. The range corrected signal after this operation RCS_{corr} will be:

$$RCS_{corr} = \frac{RCS}{1-q\,S}. \tag{66}$$

RCS is the original signal before the background value subtraction and q is a parameter equal to:

$$q = \frac{2\Delta}{c\,t}, \tag{67}$$

Where Δz is the system vertical resolution, 7.5 m in normal acquisition conditions, c the light speed and t the system dead time, estimated to be about 8 ns. Equation (55) holds if the difference between corrected and original signal is within 20%.

5.2 Loading analysis

If a previous existing analysis referred to main file is present in the Save folder, it will be automatically loaded during the main file opening. The user can also load another analysis using the **load analysis** button (box A in Figure 4.1).

Note: before loading a merged signals analysis, the user has to load the main signal (open button, Paragraph 5.1) and the additional signal used in the merging operation (Paragraph 5.3).

If the loaded results not directly refer to the current signal file, the **update** button allows applying the parameters loaded to it (example: the results of a 1064 nm analysis to a 532 nm signal file). This operation is restricted just to the visualized profile; otherwise the **update all** function (in load analysis menu) updates each profile of the file.

5.3 Loading additional signal files and merging operations

LAS is able to load and show on the main graph more signal files in order to compare or merge them with the main signal. The add file button to load or remove additional files is labelled as "**+- file**". LAS shows on the main graph the additional signals profile correspondent to the main signal index (Figure 5.3).

The extra lines button above the main graph can be used to hide/show the graphs of the additional signals.

Note: the main file is the file on which the analysis is performed. The additional files are used to perform some specific operations (see above for further information), such as merging. In order to change the main signal file, the user should use the open file button. In what follows, the main file signal is defined as **S1**, while the additional file signal as **S2**.

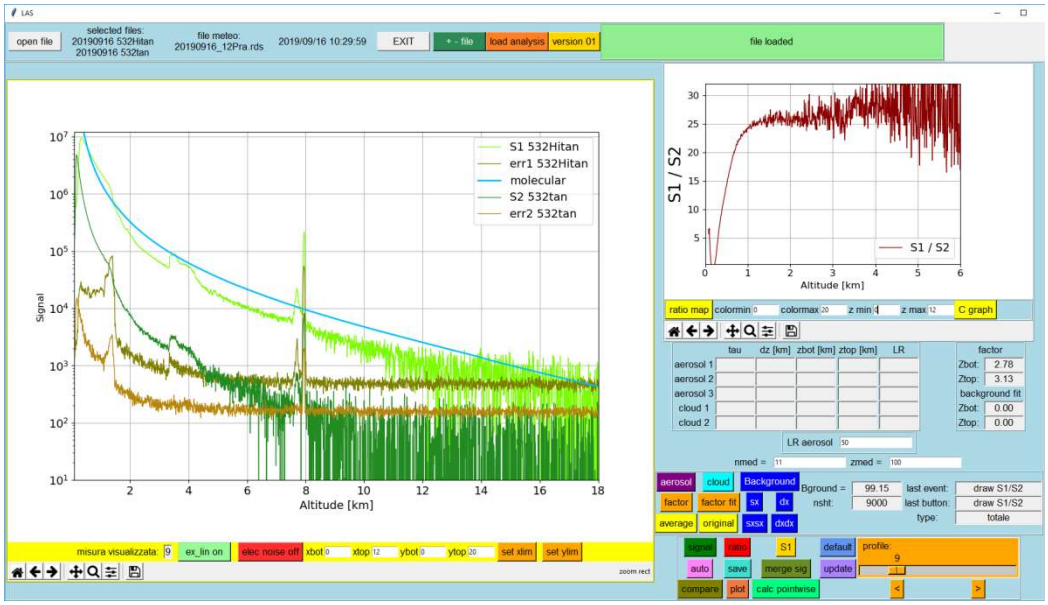

Figure 5.3: multiple signal files loaded and showed on the principal graph. The auxiliary graph shows the ratio between the two signals.

The user can merge two profiles using the **merge signal** button, which opens a pin-up menu with several voices.

Pressing **draw S1/S2** voice on the merging menu, LAS draws on the auxiliary graph the ratio between the two signals (auxiliary graph in Figure 5.3).

If an altitude bias is present between the two signals the user can select **synchronize S1** or **synchronize S2**. This operation applies a translation $T(a)S(z) = S(z+a)$ on the first/second signal in order to maximize the correlation coefficient between the two. The altitude layer on which the correlation coefficient is computed has to be selected by the user, clicking on the graphs to define the top and bottom borders. The translation will be applied to all the profiles contained in the file. The correlation coefficient calculated as function of a will be showed on the auxiliary graph.

Pressing **merge S1 S2** or **merge S2 S1** voice on the merging menu, the user selects the region in which the two profiles are merged, called **merging layer,** clicking on the graphs to define the top and bottom borders (the region will be marked by dark red vertical lines). The ratio between the two signals in the merging layer should be constant within the oscillations due to signals uncertainties.

S_1 and its uncertainty $\sigma(S_1)$ will be substitute by a new signal S_{12} with its uncertainty $\sigma(S_{12})$ composed merging together S_1 and S_2. Defined M as the mean S_1/S_2 in the merging layer:

- if merge S1 S2 was pressed, $S_{12} = S_1$, $\sigma(S_{12}) = \sigma(S_1)$ below the merging layer and $S_{12} = MS_2$, $\sigma(S_{12}) = M\sigma(S_2)$ above;
- if merge S2 S1 was pressed, $S_{12} = MS_2$, $\sigma(S_{12}) = M\sigma(S_2)$ below the merging layer and $S_{12} = S_1$, $\sigma(S_{12}) = \sigma(S_1)$ above.

On the merging layer, S_{12} linearly connect S_1 and MS_2 in order to avoid signal discontinuity according to:

$$S_{12}\left(z \in [z_{bot}, z_{top}]\right) = S_1(z)\frac{z_{top} - z}{z_{top} - z_{bot}} + S_2(z)\frac{z - z_{bot}}{z_{top} - z_{bot}}$$

Where z_{bot} and z_{top} are the altitudes of respectively the bottom and top borders.

The voice **split signals** can be used to undo the merging operation on the current profile and restore S1 to its original value.

The voice **auto-merging** can be used to automatically apply to the current profile the merging operations (synchronization and merging) used in the last analyzed signal.

5.4 Modifying Background value

The blue buttons on the bottom right of the LAS window are used to modify the signal background. A first background calculation of the profile is performed automatically when it is visualized on the main graph (Paragraph 5.1 and 5.11).

A click on the background button will open a pin-up menu with two different background calculation methods. The first method is the average of the last 100 points of the signal, the default calculation method, described above.

The second method is called **background fit** and allows the background calculation through a linear fit between the molecular and the measured signal at the upper altitudes, above all aerosol and cloud layers, in a region in which the aerosol and clouds density is close to zero. In this region the RCS depends linearly on the RCS_m (see Paragraph 3.1 and Equation (11) for detailed calculation):

$$RCS = f\,RCS_m + B. \tag{68}$$

In equation (68) f is the LIDAR factor and B the signal background; the fit process provides an estimation of both these two quantities. The estimated LIDAR factor will be used to update the molecular signal (see Paragraph 3.6).

After a click on the **background fit** voice on the pin up menu, the user has to select two altitudes clicking on the main graph. This operation selects the region where the fit is calculated. It is important in order to obtain accurate results, to select a region without cloud or aerosol layers. Two orange lines on the graph will mark the region selected.

The background value calculated is showed on the indicator next to the background button. If the background fit option was been selected the bottom and top altitudes of the region selected for the fit is showed in the LAS indicators (see Figure 5.3).

The value of B can be manually modified using the background "+" or "-". These buttons increase or decrease the original value of 0.2% for the buttons labelled "+" or "-" and of 1% for the buttons labelled "**++**" and "**--**".

LAS subtracts the background value to the measured signal and updates the principal graph accordingly.

The additional signals background can be calculated separately, selecting the signal file on which operate using the **Signal selector** button on Panel D (S1 is referred to the main signal, S2, S3, etc... to the other signal files loaded) and perform the background computation as described above.

The background uncertainty is calculated according to the chosen background operation

- **last 100 points**: the background uncertainty is the standard deviation calculated on the last 100 points of the signal divided by the root mean square of the number of the points (10);
- **background fit**: the background uncertainty and the factor uncertainty (the two fit parameters, see also Paragraph 5.6) are calculated according to the equations:

$$\sigma_B^2 = \frac{\sum_i^N \sigma_{S_i}^2 RCS_{m_i}^2}{N\sum_i^N RCS_{m_i}^2 - \left(\sum_i^N RCS_{m_i}\right)^2}$$

$$\sigma_f^2 = \frac{\sum_i^N \sigma_{S_i}^2}{N\sum_i^N RCS_{m_i}^2 - \left(\sum_i^N RCS_{m_i}\right)^2}$$

5.5 Select overlap parameters

The overlap function of the LIDAR system can be poorly known by the user. In order to provide a better estimation of R below the overlap altitude (see Paragraph 3.3) the user can extrapolate its value; z_{over} and H can be selected using the overlap altitude and overlap altitude scale buttons and clicking on the principal graph. The overlap altitude and overlap scale selected will be applied automatically to the next analyzed profile if it has no values for these two parameters.

5.6 Adding the factor layer

Once the background subtraction is performed, the user can request the calculation of the LIDAR factor f as the mean value of the ratio between RCS and RCS_m in a region in which the aerosol and clouds density is negligible defined **factor layer**. The factor uncertainty σ_f is the standard deviation divided by the root mean square of the number of the points in the factor layer.

LAS allows this calculation through the orange button labelled "**factor**". After the factor button pressure, the user selects the bottom and top altitudes of the calculation region

clicking on the principal graph. The selected region is marked by two orange lines (Figure 5.4). The factor calculation updates the RCS_m visualization on the principal graph (see Paragraph 3.6). The bottom and top altitudes used to calculate the factor are showed by the indicators on the right of the GUI, marked in red (Figure 5.3). Note that the factor value and uncertainty calculated using this method overwrites each f and σ_f from other operation such as background fit. The calculation of a new value for f deletes any aerosol or cloud layer added to the analysis (Paragraph 5.8).

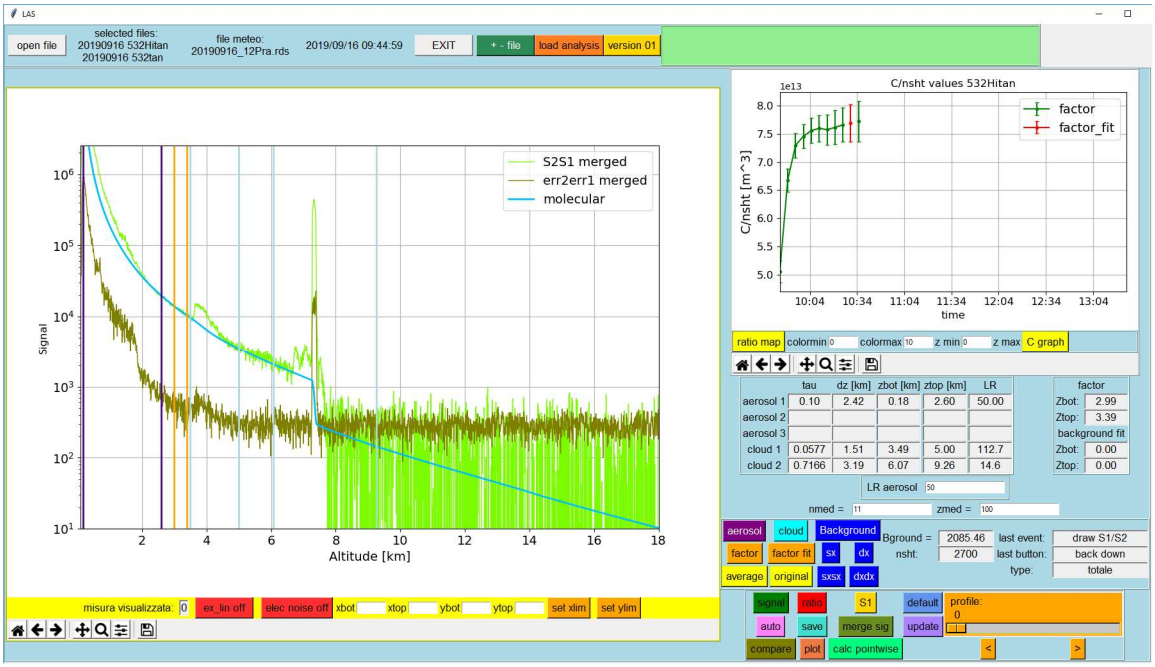

Figure 5.4: LAS window after the background and factor calculation. The orange indicators mark the region in which the factor has been calculated. The auxiliary graph shows the values of the LIDAR calibration value C divided by current profile shoot number with its uncertainty. The aerosol layer borders are marked in the principal graph by two purple lines while the cloud layers are marked by cyan lines. The indicators report some information about the results.

5.7 The LIDAR calibration and factor fit procedure

The LIDAR calibration value C changes over time. It can be retrieved for each measured profile by LAS through the factor measurement obtained by the LIDAR factor procedure. In order to obtain an accurate estimation of C, LAS has to resolve the aerosol and cloud layers eventually present **below** the factor layer. The C value computed for each profile normalized by the profile shot number and its uncertainty are showed in the auxiliary graph by pressing the **C graph** button as a green dot with green error bar. The C values interpolated using the factor fit procedure are shown as red dots with red error bar.

In some circumstances the user cannot clearly identify a region in which no aerosol or clouds are present in order to select the factor layer. In such cases the user can employ the **factor fit** procedure using the factor fit button. This procedure calculates the LIDAR calibration value C through a linear interpolation between the C values computed for the other analyzed profiles and **set the C retrieval mode** for the calculation. The user should use the factor fit procedure even in case a top-cloud layer is present on the profile

(Paragraph 3.5.3), a layer type that cannot be resolved properly using the factor retrieval. The knowledge of C allows the retrieving of the backscattering and extinction profile without needing of an estimation of f. The details of the calculation are in Paragraph 3.5.

This procedure assumes a linear variation of the parameter C over time and implies that the factor layer can be identified in at least one of the profiles contained in the file.
Two orange dotted lines will be displayed in the principal graph in order to show the mean values of bottom and top limits of the factor layers of the profiles used to interpolate C.

5.8 Adding or removing aerosol and clouds layers

In order to retrieve the backscattering Ratio and the backscattering and extinction profiles the user has to add layers of aerosol and clouds to the calculation. Adding a layer to a profile means to retrieve the results for that specific profile region using a specific algorithm. Actually there are three **layer types: aerosol, single-cloud and top-cloud**. The retrieval algorithms are described in details in Chapter 3.

In order to add an aerosol layer, the user must select the purple button marked as "**aerosol**". The button pressure opens a pin up menu with three different commands: "**+**", "**-**" and "**reset**".

The "**+**" command add an aerosol layer to the computation. The user has to select the bottom and top altitudes of the layer through two clicks on the graphs. Two purple lines on the principal graph will mark the region selected (Figure 5.3). The new aerosol layer added is characterized by a value of the LIDAR Ratio selected by the user through the **LR aerosol** entry on the D panel of the window (Figure 4.1). Each aerosol layer added to the calculation can be characterized by a different value of LIDAR Ratio. Note that it is not allowed to overlap the factor layer with cloud or aerosol layers.

The D panel presents also several indicators that display the optical depth, the width and the bottom and top altitudes and the LIDAR Ratio of the first three aerosol layers added to the calculation (the corresponding indicators will remain empty if less aerosol layers are presents).

The "**-**" command on the pin up menu will delete the last aerosol layer added to the calculation while the "reset" command will delete all the aerosol layers.

The single-cloud and top-cloud layers can be added to the calculation in the same way through the cyan button "**cloud**" and its pin up menu. The two types of cloud calculation are marked as "single" and "top-cloud".

After the layer type selection, the user has to select a region on the graphs that will be marked by cyan lines on the principal graph. A clouds layer can be removed through the "**-**" and "**reset**" functions in the same way of the aerosol layers.

The optical depth, the width, the bottom and top altitudes and the LIDAR Ratio of the first two cloud layers added to the calculation will be displayed in the indicators in the D panel of the window.

Note that for both the single and top-cloud layers the LIDAR Ratio is a product of the calculation.

The **single-cloud calculation** can be used to resolve isolated clouds which borders can be clearly identified from the aerosol layers borders. This kind of calculation can be performed in **both** factor and factor fit mode. A single-cloud is showed in Figure 5.5.

The **top-cloud calculation** can be used to resolve a cloud which top edge can be clearly identified, while its bottom edge cannot, due to the proximity to an aerosol layer. In order to employ this kind of calculation the user **has to select the factor fit procedure** due to the needing of the profile C value as apriori knowledge. A top-cloud is showed in Figure 5.5.

Important: the order in which the layers are added to the calculation depends on the calculation procedure used to provide an estimation of the factor.

- If f was calculated using the **factor calculation** or the **background fit** procedure, in order to resolve a layer, the software needs the information of the region between the layer itself and the factor layer. This constrains the user to order the layers by decreasing altitudes for the profile region below the factor layer and by increasing altitude for the profile region above the factor layer
- If the "factor fit" procedure was used, the layers have to be ordered by increasing altitude from the ground. Note that in this case the factor layer is not present.

LAS employs two different kinds of convergence conditions for some algorithms: **punctual** and **integral** convergence. The user can select the condition by the **Convergence mode button** (box D in Figure 4.1).

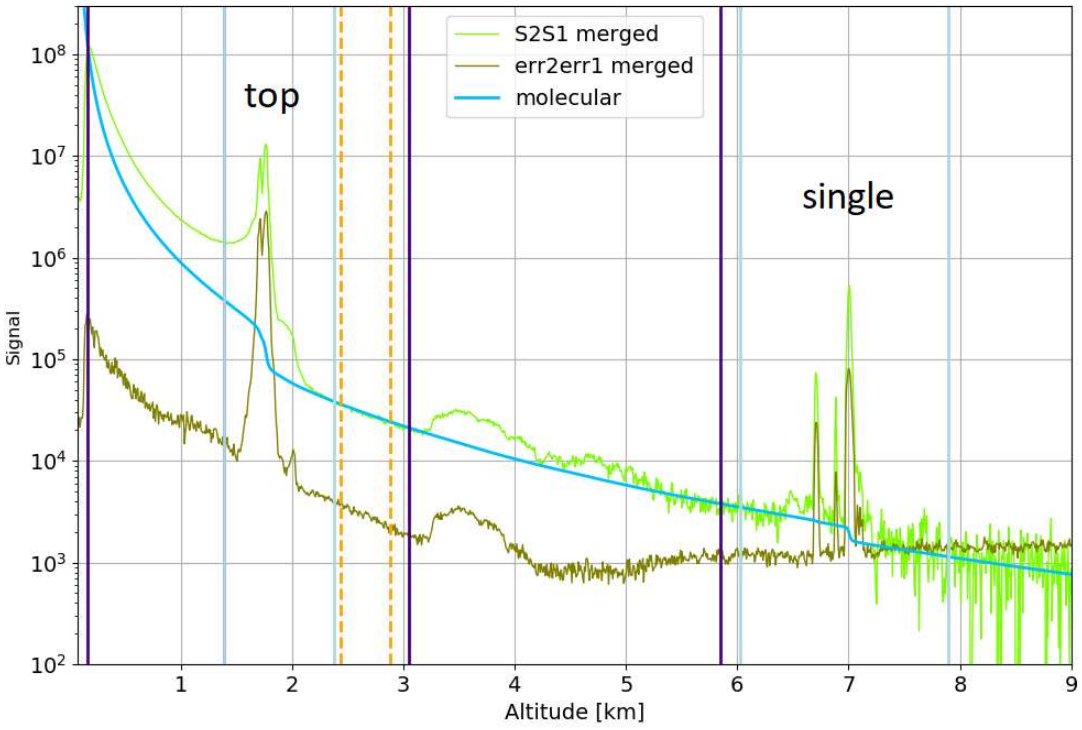

Figure 5.5: the two different kinds of clouds: single and top-cloud.

5.9 LAS graphs

Pushing the red button marked as "**ratio**", the user can visualize on the principal graph the backscattering ratio profile (blue line) and its uncertainty (cyan line) resulting from calculation. Purple, cyan and orange lines mark respectively the aerosol layers, the cloud layers and the region in which the factor was calculated (see Figure 5.5). This function is

useful to evaluate the goodness of the data analysis. The user can visualize again the signal through the green button “signal”. LAS can also show the backscattering coefficient β, the extinction coefficient α with their uncertainties and the derivative of the signal logarithm pressing the **plot** button (box D in Figure 4.1).

It is also possible to visualize the backscattering ratio as function of time and altitude in the auxiliary graph pressing the **Ratio map** button. Note that the map shows only the backscattering ratio of the analyzed profiles.

The value of the calibration value obtained from the retrieval can be visualized using the C graph button. The green dots marks the C values calculated by the factor retrieval while the red dots the values obtained using the factor fit procedure.

LAS can visualize on the auxiliary graph the optical depth retrieved; it is also possible to visualize the values measured by other instruments (actually CIMEL and PREDE photometers) using the plot tau button.

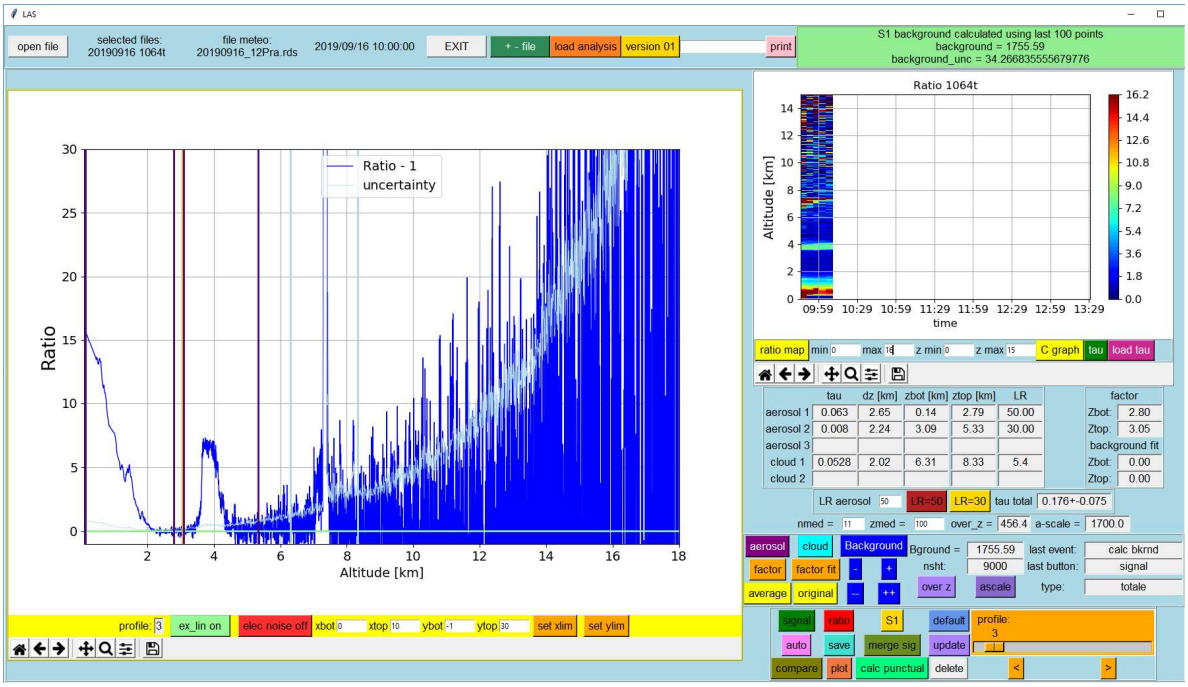

Figure 5.5: The backscattering ratio resulting from the analysis. The aerosol layer borders are marked by purple lines, the clouds borders are marked by cyan lines while orange lines mark the region in which the factor has been calculated. The cyan line represents the uncertainty. The auxiliary graph shows the backscattering ratio value as function of time and altitudes.

5.10 Save or delete analysis

The current profile analysis can be saved through the pressure of the save button. The first time the user pushes the save button, LAS creates a NETCDF4 file containing all the global attributes, the variables and their attributes (as described in Appendix C) and all the parameters used in the calculation such as the top and bottom altitude used in the factor calculation, the background value, the number and all the information about the layer used. In addition to these, also the results obtained for the current analyzed profile. The output file name is assigned automatically by the software taking into account the location, the analyzed channel, the merging operation eventually performed, the date and time of the observations, the measurement integration time. Below an example of an output file name:

rome_005min_532MTAN_back_20190916094459_v01.h5

The different colours represent the different part of the file name: measurement location, integration time in minutes, channel, file type (back means elastic LIDAR analysis), measurement start date in format yyyymmddHHMMSS and version number.

If a previous analysis already exists, the software will update the output file with the new results. The user can prevent this changing the **output file version** selecting a different version by the pressure of **version button** (the box A in Figure 4.1). A description of an output file is provided in the Appendix C. The output file metadata and variable names are inspired to Generic Earth Observation Metadata Standard (GEOMS) guidelines, available at https://evdc.esa.int/documentation/geoms/.

Note: saving is not an automatic process, and the user can decide not to save the analysis of a single profile if it is not correct.

The current analysis can be deleted and analysis parameters restored to their default values through the "**Delete"** button. Note that this procedure does not modify eventually saved profile in the output file.

5.11 Switch profile and automatic analysis

A signal file usually contains n different profiles S^i. The orange cursor on the bottom left of the window allows the user to slide between the profiles contained in the input file. LAS maintains in memory the analysis parameters and the results of each signal profile until the program termination.

If no background value is in memory for the new visualized profile S^j, LAS automatically calculates B^j: if the previous B^i was calculated using the "last 100 points" method or using the background fit operation, B^j will be obtained using the same operation (in case of background fit, the fit layer will be the same).

If z^j_{over} and H^j variables of S^j have no value assigned, LAS will assign to them from the previous visualized profile.

If the profile S^j shows a similar shape of an already analyzed profile S^i, the user can decide to automatically analyze S^j using the **auto button**. LAS analyzes S^j using the same parameters of S^i. The software automatically updates the principal graph and the indicators according to the analysis results.

Using the **Default button** the user can define the current profile parameters as default parameters for automatic analysis. This operation is useful to apply to a measured profile the same kind of analysis of an already resolved profile (not necessarily the last one).

5.12 Signal/Ratio smoothing average

LAS can smooth the signal or backscattering ratio profiles through the average function, called by the yellow button labelled "**average**". The smoothed signal or backscattering ratio profile will be plotted in the principal graph. The yellow button labelled "**original**" undo the smoothing process. The average operation allows the user to select the number of points to average; it is also possible to smooth only a certain region of the measured profile, from an altitude z_{med} to above. This procedure reduces the signal or ratio uncertainties by a factor $\sqrt{n}$, where n is the smoothing points number.

5.13 LAS results comparison

The **compare button** is used for testing, comparing the results of the previous analysis software with LAS results.

6. LAS DATA STRUCTURES, METHODS AND FUNCTIONS

This chapter describes the main information clusters managed by LAS and the several methods, instances and functions used to handle the operations described in the previous chapter.

According to Python programming, a **function** is an independent part of code that receives a data flow as input, performs some operations and can return some data as output. The LAS functions are defined in two libraries: LAS_inout and LAS_computation.

A **method** is an internal function while an **instance** is an internal attribute of a Python *class.* With the exception of the functions, the other objects described below are all instances or methods of the class named **LASclass**, defined in the main code. LAS source code uses extensively the Python construct "**self**". This construct allows the data to flow between the different codes, methods and instances composing LAS without the needing to strictly define inputs and outputs variables. The names of the different data structures and methods described below are all preceded by the prefix "self.", except for the method definition. This prefix is omitted for clarity sake.

In order to invoke a data structure or a method defined in LAS, the user has to type its name on the Python console, substituting the prefix "self." with the prefix "LAS.".

6.1 Main data structures

The data, flags and information managed by LAS are stored in Python structures called dictionaries described below. In order to show the content of these structures, the user needs to exit the program and types on the Spider command box the command *LAS.dict_name["dict_voice"].* Alternatively it is possible to use the **Command entry and print button** (Paragraph 4) typing *self.dict_name["dict_voice"]* without exit the program.

LAS works using three main families of data, called Input, ParamAll and OutputAll. The Input dictionary contains all the information read from the input signal and meteorological files. OutputAll and ParamAll are lists of dictionaries; each element contains results and analysis parameters respectively of the single profiles. The single OutputAll element and ParamAll element are named respectively as Output and Param dictionaries.

In case of several files loaded, the additional files information is contained in a list called MultiInput. Each element of this list contains the same data structure and variables of the Input dictionary. Each additional file refers also to an element of a second list of dictionaries called MultiParam, containing the information needed to merge the files.

The Default dictionary is used in order to perform automatic analysis. It has the same voices and variables of the Param dictionary. Modifying a Param voices of the current profile, for example adding a cloud layer or a factor layer, will change also the content of the correspondent voice of the Default dictionary. Changing profile will not change the Default content. The Default information allows applying the same analysis parameters to different profiles by pressing the auto button.

The Extra dictionary contains the aerosol optical depth and other measurements of other instruments that can be loaded in order to improve the analysis.

The INFO dictionary contains the instrumentation specifics and other useful information. It is created by LAS_info.py.

The Input, Param, Output and MultiParam and Extra dictionaries will be described in the following paragraphs. In what follows the several data type are described as:

- array(n): python numpy.array object with length n
- list(n): python list object with length n
- double/int/string: a single floating point/integer/string variable.

Moreover n defines the number of points of the altitude grid, k the total number of profiles contained in the file, a the number of aerosol layers added to the calculation and c the number of cloud layers added to the calculation.

6.2 Input structure

The Input dictionary contains the information loaded from the main signal files. The Input voices are described below.

- **Input["alpha_m"]**: array(n). The molecular extinction coefficient α_m
- **Input["Altitude"]**: double. The instrument altitude from the sea level
- **Input["analog_fullscale"]**: Double. The scale width used to digitalize the signal in the acquisition process
- **Input["AverageTimeMinute"]**: double. The number of minutes of accumulation to measure the visualized profile
- **Input["beta_m"]**: array(n). The molecular backscattering coefficient β_m
- **Input["channel"]**: string. The name of the channel that acquired the signal
- **Input["electronic_noise"]**: array(n). The uncertainty profile due to electronic noise (it is a component of the uncertainty calculated in Input["err"])
- **Input["err"]** : array(n). The current signal profile uncertainty, taking into account the background uncertainty.
- **Input["err_ori"]** : array(n). The original signal profile uncertainty.
- **Input["file_path"]**: string. The complete path of the input signal file
- **Input["lambda"]**: double. The wavelength in nm of the analyzed LIDAR channel
- **Input["Latitude"]**: double. The intrument latitude
- **Input["Licel_channel"]**: string. The Licel used to acquire the signal
- **Input["Longitude"]**: double. The instrument longitude
- **Input["nsht"]**: array(k). List containing the number of laser shoots accumulated of all the profiles contained in the signal files
- **Input["RepetitionRate"]**: double. The laser shoot frequency (number of shoots / sec)
- **Input["sig"]:** array(n). The current signal profile minus the background value calculated
- **Input["sig_m"]**: array(n). The molecular signal showed in the principal graph. This quantity is used just for graphical representation: $S_{show}(z) = CS_m(z)e^{-2\tau_a(z)}$, where C is the LIDAR calibration constant and τ_a the aerosol optical depth (see paragraph 3.6)
- **Input["sig_m_ori"]**: array(n). The molecular signal S_m computed using the meteo temperature and pressure profiles; $S_m(z) = \beta_m(z)e^{-2\tau_m(z)}$
- **Input["sig_ori"]**: array(n). The original signal, before merging or smoothing operations or background subtractions
- **Input["tau_m"]**: array(n). The molecular optical depth τ_m at the altitude z computed as $\tau_m = \int_{z_0}^{z} \alpha_m dz$, where z_0 is the instrument altitude from the sea level
- **Input["time"]:** array(k). The time in Julian day of the profiles contained in the signal files
- **Input["Vertical_resolution_m"]**: double. The vertical resolution in meters of the current signal profile.
- **Input["Z"]**: array(n). The current profile altitudes grid

6.3 Param structure

The Param dictionary contains the current profile analysis parameters.

- **Param["aero_mode"]:** list(a). Flag list referring to the current profile aerosol layers retrieval mode. Actually not used

- **Param["aerosol_zbottom"]:** list(a). List containing the bottom borders altitudes of the aerosol layers
- **Param["aerosol_ztop"]:** list(a). List containing the top borders altitudes of the aerosol layers
- **Param[“ascale”]:** double. The value of the overlap altitude scale H
- **Param["bgrnd_mod"]:** double. Multiplicative constant used to change the computed background; default = 0
- **Param["bgrnd_z"]:** list(2). List of two elements containing the bottom and top border altitudes of the background layer. If not background layer is present it is equal to [0,0]
- **Param["cloud_mode"]:** list(c). Flag list referring to the current profile cloud layers retrieval mode: 0 refers to a “single-cloud” retrieval, 2 to a “topcloud” retrieval. The value 1 refers to a “mixed” retrieval but is actually not implemented
- **Param["cloud_zbottom"]:** list(c). List containing the bottom borders altitudes of the clouds layers
- **Param["cloud_ztop"]:** list(c). List containing the top borders altitudes of the clouds layers
- **Param["computation_mode"]:** string. Flag equal to 'factor' or “factor_fit” depending on which of these two operations was used to analyze the profile
- **Param["convergence_mode"]:** string. Flag equal to “punctual” or “integral” depending on which computation algorithm was used to analyze the profile
- **Param["fac_z"]:** list(2). List of two elements containing the bottom and top border altitudes of the factor layer
- **Param["LR_aero"]:** list(a). List contaning the Lidar Ratio assigned by the user to each aerosol layer
- **Param["LR_aero_unc"]:** list(a). List contaning the Lidar Ratio uncertainty of each aerosol layer. Actually it is not used
- **Param["merge_z"]:** list(2). List of two elements containing the bottom and top border altitudes of the merging layer. If not merging layer is present it is equal to [0,0]
- **Param["merged_signals"]:** string. Flag equal to "no" if no merging operation was performed or “S1S2” or “S2S1” if the user selected one of the operations merge S1 S2 or merge S2 S1
- **Param[“over_z”]:** double. The overlap altitude z_{over}
- **Param[“saved_flag”]:** integer. Flag equal to 1 if the profile analysis has been saved, 0 otherwise
- **Param["smooth_bins"]:** int. The number of points used for the moving average (default equal to 0)
- **Param["smooth_z"]**: double. The altitude above which the moving average was calculated
- **Param["synch_binshift"]**: int. The number of bins the signal was eventually shifted
- **Param["synch_signals"]**: string. Flag equal to "no", if no synchronization between signal was requested, or “S1”/“S2” if the main/additional signal profile has been synchronized
- **Param["synch_z"]**: list(2). List of two elements containing the bottom and top border altitudes of the synchronization layer. If not synchronization layer is present it is equal to [0,0]

6.4 Output structure

The output dictionary contains the analysis results. All the quantities are referred to their respective profile

- **Output["alpha"]:** array(n). The current extinction coefficient α profile
- **Output["alpha_unc"]:** array(n). The current extinction coefficient α profile total uncertainty
- **Output[“alpha_unc_ran”]:** array(n). Extinction coefficient random uncertainty
- **Output[“alpha_unc_sys”]:** array(n). Extinction coefficient systematic uncertainty
- **Output["beta"]:** array(n). The current backscattering coefficient β profile
- **Output["beta_unc"]:** array(n). The current backscattering coefficient β profile total ucertainty
- **Output["beta_unc_ran"]**: array(n). Backscattering coefficient random uncertainty
- **Output["beta_unc_sys"]**: array(n). Backscattering coefficient systematic uncertainty

- **Output["bgrnd"]**: double. The background value B calculated for the current profile. "Defined B_{ori} the result of the bacground fit or background last 100 points operations, and m the Param["bgrnd_mod"] value, $B = (1 + m)B_{ori}$
- **Output["bgrnd_unc"]**: double. The background uncertainty of the current profile
- **Output["C"]**: double. The LIDAR calibration factor of the current profile **normalized** by the profile shoots number
- **Output["C_unc"]**: double. The LIDAR calibration factor uncertainty of the current profile
- **Output["factor"]**: double. The factor value of the current profile
- **Output["factor_unc"]**: double. The factor uncertainty of the current profile
- **Output["LR_cloud"]**: list(c). List containing the LIDAR Ratio of the cloud layers of the current profile. **Note**: the LR value of a cloud layer is a product of the retrieval
- **Output["LR_cloud_unc"]**: list(c). List containing the LIDAR Ratio uncertainty of the cloud layers of the current profile. **Note**: the LR value of a cloud layer is a product of the retrieval
- **Output["Ratio"]**: array(n). Backscattering ratio R profile
- **Output["Ratio_0"]**: array(n). Profile of R_f or R_c depending of the type of calculation (see paragraphs 3.3 and 3.5)
- **Output["Ratio_0_unc"]**: array(n). R_f or R_c total uncertainty profile, depending of the type of calculation (see Chapter 3)
- **Output["Ratio_0_unc_ran"]**: array(n). R_f or R_c random uncertainty profile, depending of the type of calculation (see Chapter 3)
- **Output["Ratio_unc"]**: array(n). R total uncertainty profile
- **Output["tau_aero"]**: list(a). List containing the optical depth of each aerosol layer of the current profile, computed as $\tau = \int_{z_{bot}}^{z_{top}} \alpha dz$
- **Output["tau_aero_unc"]**: list(a). List containing the optical depth uncertainty of the aerosol layers of the current profile.
- **Output["tau_cloud"]**: list(c). List containing the optical depth of each cloud layer of the current profile, computed as $\tau = \int_{z_{bot}}^{z_{top}} \alpha dz$
- **Output["tau_cloud_unc"]**: list(c). List containing the optical depth uncertainty of the cloud layers of the current profile
- **Output["tau_total"]:** double. Total optical depth of the profile, calculated as the sum of the optical depth of each aerosol and cloud layer added to the calculation
- **Output["tau_total_unc"]:** double. Total optical depth ucertainty, calculated as the square sum of the optical depth uncertainties of each aerosol and cloud layer added to the calculation

6.5 MultiParam structure

The elements of this list contain the following voices referred to the current profile of the additional file:

- **MultiParam["bgrnd"]**: double. The background value B subtracted to the signal profile. Defined B_{ori} the "bgrnd_ori" value, and m the "bgrnd_mod" value, $B = (1 + m)B_{ori}$
- **MultiParam** [**"bgrnd_mod"]**: double. Multiplicative constant used to change the computed background; default = 0
- **MultiParam["bgrnd_ori"**]: double. The background value B, **before** it was eventually modified by the user. It is equal to "bgrnd" if the user did not modify the value. It is the result of the background fit or background last 100 points procedure
- **MultiParam**["**bgrnd_unc"]**: double. The background uncertainty
- **MultiParam["bgrnd_z"]**: list(2). Two elements list containing the background layer bottom and top altitudes used in the background fit procedure. If the background has been calculated using the last 100 points it is [0, 0]
- **MultiParam["merged_signals"]**: int. Flag equal to 1 if the signal has been merged, elsewhere equal to 0
- **MultiParam["synch_binshift"]**: int. The number of bins the signal has been eventually shifted (default = 0)

- **MultiParam[“synch_signals”]**: int. Flag equal to 1 if the signal has been shifted, elsewhere equal to 0

6.6 Extra structure

The Extra dictionary actually can contain the optical depth measured by CIMEL and PREDE-POM spectrophotometers. In what follows l/p is the number of measurement contained in the CIMEL/PREDE-POM file.

- **Extra[“CIMEL_datetime”]**: list(l). List of python datetime referring to the CIMEL measurements time.
- **Extra[“CIMEL_lambda”]:** double. The wavelength measured by CIMEL.
- **Extra[“CIMEL_tau”]:** array(l). The optical depth measured by CIMEL.
- **Extra[“CIMEL_time”]:** array(l). The date and time of CIMEL measurements in julian day.
- **Extra[“POM_datetime”]:** list(p). List of python datetime referring to the POM measurements time.
- **Extra[“POM_lambda”]:** double. The wavelength measured by POM.
- **Extra[“POM_tau”]:** array(p). The optical depth measured by POM.
- **Extra[“POM_time”]:** array(p). The date and time of POM measurements in julian day.

6.7 LAS main methods

Below, in Table 6.1, are shortly described the different methods defined in LAS_main.py code.

name	Description
_quit	Exit the program and close the LAS window
add_file	Load an additional file
add_file_operation	The operations needed to load an additional file
aerosol_command	Prepare the LAS flags for the aerosol computation
auto_merging	Merge the profiles using the same Param[“merge_z”] of the previous visualized profile
automatic_analysis	Analyze automatically a profile according to the setting in Default dictionary
average_ratio	Average the Ratio following the parameters inserted by the user
average_signal	Average the signal following the parameters inserted by the user
background_down	Reduce the background value by 0.2%
background_downdown	Reduce the background value by 1%
background_up	Increase the background value by 0.2%
background_upup	Increase the background value by 1%
calculate_background_100points	Calculate the background of the selected signal using the average of the last 100 points
calculate_background_fit	Enter in the background fit computation mode for the selected signal
calculate_factor	Calculate the factor value
change_profile	Change the visualized profiles
clear_ratiomap	Clear the auxiliary graph from the Ratio map
cloud_command	Prepare the LAS flags for the cloud computation
compare_results	Compare results of LAS with previous IDL program used in BAQUNIN
debug	Use this function to help debugging the code

DefaulteqParam	This function is used to make Default = Param and it is necessary to avoid the alias creation by Python
define_color	Define the colour of the graph lines
delete_aero_layer	Delete the last aerosol layer added to the analysis
delete_analysis	Use this to delete the current profile analysis
delete_cloud_layer	Delete the last cloud layer added to the analysis
derivative_calculation	Compute the derivative of the signal
draw_Cgraph	Show the value of C for the various profiles on the auxiliary graph
draw_ratiomap	Draw the Ratio map on the auxiliary graph
draw_tau	Draw the optical depth calculated by LAS or measured by other instruments on the auxiliary graph
elecnoise_change	Show/hide the electronic noise on the principal graph on signal mode
extraline_change	Show/hide the vertical lines marking the Multifile operations
factor_fit	Factor fit computations
fill_aerosol_indicator	Update the aerosol indicators on the GUI
fill_cloud_indicator	Update the cloud indicators on the GUI
graph_main_limsave	Method that stores in memory the current y and x axes limits of the principal graph
graph_aux_limsave	Method that stores in memory the current y and x axes limits of the auxiliary graph
init_Output_Param_cluster	Initialize the InputAll and ParamAll dictionaries lists
load_analysis	Load results and parameters from a previous analysis
load_meteo	Load the meteorological file used to calculate the molecular signal
load_nc	Load a signal file
loadOutput	This function is used to load the Output dictionary from the OutputAll structure and it is necessary to avoid the alias creation by Python
loadParam	This function is used to load the Param dictionary from the ParamAll structure and it is necessary to avoid the alias creation by Python
load_previous_analysis	Load previous analysis results and parameters
LR30	Set LR_aerosol = 30
LR50	Set LR_aerosol = 50
merge_operation	The operations required in order to merge the profiles
merge_S1S2	Merge the main signal (bottom) and the first additional signal (top)
merge_S2S1	Merge the main signal (top) and the first additional signal (bottom)
mixed_command	Prepare the LAS flags for the mixed computation of a cloud inside two layers of aerosol. Actually disabled
multifile_check	Check if more than one profile is loaded and perform the multifile operations according to Param
nextprofile	Switch to the next profile
onclick	Method that handles the mouse clicks on the graphs, recognizes the operation requested and stores the calculations results in memory
open_nc	Open the main signal file
original	Restore signal and ratio to their original vertical resolution
Param_eq_Default	This function is used to make Param = Default and it is necessary to avoid the alias creation by Python
plot_alpha	Use it to draw alpha on the principal graph
plot_beta	Use it to draw beta on the principal graph

plot_merge_ratio	Plot the merged signal
plot_S1S2ratio	Plot the ratio between the main signal and the first additional signal
plot_signal	Draw the signal on the principal graph
plot_vertical_lines	Plot the vertical lines on the graphs
popup_aerosol	Open the aerosol calculation menu
popup_average	Open the average menu
popup_background	Open the background menu
popup_calcmode	Open the convergence-mode menu
popup_cloud	Open the cloud calculation menu
popup_file	Open the add/remove file menu
popup_merge	Open the merge menu
popup_opsig	Open the menu to choose the signal on which operate
popup_plotmode	Open the plot menu
popup_version	Open the version menu
prevprofile	Switch to the previous profiles
ratio	Method that draws the Ratio on the principal graph
read_CIMEL_data	Read and select the optical depth from CIMEL data
read_POM_data	Read and select the optical depth from POM data
recalculate_background	Change the background value by pressing the background up or down buttons
remove_file	Remove last additional file loaded from memory
reset_aerosol	Delete all the aerosol layers added to the analysis
reset_cloud	Delete all the cloud layers added to the analysis
reset_Extra	Initialize/reset the Extra dictionary
reset_Param_Output	Initialize the Output and Param dictionaries
save	Save current profile analysis
save_Default	Use this to save the current analysis parameter as Default analysis that can be recalled using auto button
saveOutput	This function is used to copy the Output dictionary into the OutputAll structure and it is necessary to avoid the alias creation by Python
saveParam	This function is used to copy the Param dictionary into the ParamAll structure and it is necessary to avoid the alias creation by Python
select_ascale	Select the value of aerosol vertical scale by clicking on the graphs
select_over_z	Select the value of overlap altitude by clicking on the graphs
set_calcmode_integral	Set the integral convergence mode
set_calcmode_punctual	Set the punctual convergence mode
set_graphxlim	Set the x limit of the principal graph
set_graphylim	Set the y limit of the principal graph
set_operative_signal	Set the signal on which you are currently operating (for example for background operations)
set_savefilename	Set the name of the output file
shift_signal	Shift the signals (translation on the z axis)
signal	Methods called by signal button. It draws the signal on the principal graph
split_signals	Split a merged signal.

synchronize_signal	Synchronize the loaded signals
topcloud_command	Prepare LAS flags for the top-cloud computation.
update	Update calculations, flags and graphs
update_background	Update the background value according to Param
update_calculation	Update the calculation according to Param
update_flag_Zindicator	Update the Z flag on the screen
update_graph	Update the graph according to self.graphmode flag
update_graph_aux	Update the auxiliary graph
version_down	The output file version will be changed by -1
version_up	The output file version will be changed by +1

Table 6.1: the LAS methods

6.8 LAS functions

This paragraph briefly describes the functions contained in LAS_computation.py and LAS_inout.py. The data structures in Table 6.2 and Table 6.3 are referred to the those defined in the previous paragraphs; Zbot and Ztop are referred to the top and bottom altitude of the layer to be calculated by the function, numprofile is the profile number. The equal signal (=) after a quantity indicates its default value. The square brackets "[]" indicates a list of elements; the Python dictionaries are explicitly indicated with their keys.

Name	in	out	Description
aerosol_core	Input, Output, Param, Zbot, Ztop, LR, numprofile	Output	Retrieval process for aerosol layers
background_fit	Input, [Ztop, Zbot], sig_m	factor, background, background_unc, factor_unc	compute the background through the background_fit procedure
calc_factor	Input, Output, [Ztop, Zbot]	factor, factor_unc	factor calculation process
calc_molecular	meteo, z, wavelength, parallel_flag=False	beta_m, alpha_m, tau_m	calculate beta_m, alpha_m, Sig_m, tau_m
calc_num_molecules	Pressure_profile, Temperature_profile	Molecular density	calculate the number of molecules used for the molecular signal calculation
calc_tau_Trasmission	z, alpha, Zbot, Ztop	tau, Trasmission	calculate the layer trasmission
calculate_C	Input, Param, Output, numprofile	Output	calculate the LIDAR C
calculate_electronic_noise	Input, record, graph_flag=False	Input	calculation of the electronic noise. Set the graph flag as True to produce a graph to study the results
calculate_ratio_btwn_signals	Input1, Input2, Zbot, Ztop	ratio_btwn_sig	calculate S1/S2 with S1 and S2 two signal loaded
calculate_single_cloud	Input, Param, Output, Zbot, Ztop, numprofile, Zf=0, mode="single"	Output, Param	inversion process for cloud layers
calculate_total_tau	Output	Output	calculate the total optical depth of the profile adding together the tau_aero and tau_cloud

correlation_index_vector	Input1, Input2, Zbot, Ztop, shiftmax=20	[corr_vec, shift_vec]	calculate the correlation index used to synchronize two signals
delete_aero_layer	Input, Param, Output, Zbot, Ztop, numprofile		delete an aerosol layer
find_order_indbot_indtop	z, Zbot, Ztop	[indbot, indtop]	find the Zbot and Ztop indexes on the z vector
init_Ratio	Input, Param, Output, numprofile	Output	initialize the Ratio as Ratio_0 and the Ratio uncertainty
interpolate_C	Input, Output, ParamAll, OutputAll, numprofile	Out dictionary with the voices: LIDAR_C, msg, fac_z, factor, factor_unc, C_unc	calculate the LIDAR C through a linear interpolation between C value from others profiles
merge_two_sig	Input1, Input2, Zbot, Ztop	Sig_merged, Err_merged	the calculation to merge two signals
order_altitude	zbot_list, ztop_list, resolution, fac_zbot, fac_ztop, Param	Out dictionary with voices: msg, zbot, ztop	order Zbot, Ztop and fac_zbot, fac_ztop (factor layer borders) if presents
simple_derivative	f, z	df	calculation of the derivative df(z) of f(z)
simple_integral	f, z, Zbot, Ztop	F	calculate the integral F of f(z) from Zbot to Ztop
smooth_Ratio	Output, Input, zmed, nmed	Output	Ratio smoothing process (nmed points from zmed altitude)
smooth_signal_from	Input, zmed, nmed		signal smoothing process (nmed points from zmed altitude)
update_sig_m	Input, Param, Output, numprofile	Input	update sig_m, molecular signal showed on the graph, according to the factor and the trasmission computed.
update_tau	Input, Param, Output	Output	calculate the optical depths of the aerosol and cloud layers (integral of alpha from Zbot to Ztop)

Table 6.2: LAS_computations functions.

Name	In	out	Description
dead_time_correction	Input, numprofile	Input	calculate the dead time correction of the digital signals
load_analysis	filefullpath, ParamAll, OutputAll	OutputAll, ParamAll	Load previous analysis
open_nc	inpath	filenamelist	create a GUI in order to select a file to open
prepare_plot	plotreference, start_z_limit=12, labsize=14	Plot_reference	prepare the principal graph to visualize the signal
profile_time	Input, numprofile	datestr	convert the julian day time to string
read_CIMEL_tau	CIMELpath	Out_dictionary with voices: tau340, tau400, tau500, tau675, tau870, tau1020, time	read a CIMEL file, extracting the aerosol optical depth (tau)
read_LIDAR_signal	Filename, numprofile	Input	read a single LIDAR profile in a signal file
read_POM_tau	POMpath	Out dictionary with voices: tau340, tau400, tau500, tau675, tau870, tau1020, time	read a POM file, extracting the aerosol optical depth (tau)
read_num_LIDAR_profiles	filepath	Profile_total_num	read the number of profiles in the file
read_rdsfile	meteofile	Out dictionary with voices: P, T, z, RH, mixR, dwpT, drcT, SKNT, THAT, THTE, THTV	read a radiosonde file
save_Output	Input, Param, Output, numprofile, firstsave, INFO, MultiInput=[], MultiParam = []	None	save the analysis of the current profile

Table 6.3: LAS_inout.py functions

7. LAS INPUT OUTPUT FILES

This chapter briefly describes LAS input and output files.

7.1 Input signal file

The input signal file is a NETCDF file containing all information about location, system and measurement as global attributes, and the averaged signal profiles.

The LIDAR collects profiles with temporal resolution of the order of seconds; in order to improve the signal to noise ratio is useful to average together several minutes of measurement in a single profile.

The name of a signal file contains information about the measurement location, the time interval for the average process, the wavelength detected and the date and time of the first profile contained in the file (see Appendix A for detailed information of a typical input signal file).

This file contains two 2D matrices named “ch” and “err” respectively, containing the binned profiles measured at different times and their uncertainty. The number of the point of the grid along the vertical direction (i.e. the altitude) and the vertical resolution are both attributes of “ch”. The two vectors “nsht” and “time” contain respectively the information about the average number of laser shots and the date and time of the binned profiles.

7.2 The meteorological file

The meteorological file contains the measurement collected by radiosonde. The pressure and temperature vertical profiles are used to retrieve molecular signal, the signal produced by the atmospheric molecules. The meteorological file is an ASCII file with a specific format; Appendix B displays an example of meteorological file.

7.3 Output file

The LAS analysis is saved in a NETCF file. The output file contains information about location, instrument and measurement in addition to the variables retrieved during analysis process: vertical profiles of backscattering, backscattering ratio and extinction coefficient, optical depths, the bottom and top altitudes of aerosol layers and clouds and L of each aerosol or cloud layer used in the retrieval process.

All the parameters involved in the retrieval process, such as the LIDAR factor and background calculated, and all the information about the measurements (such as the date and time) and about acquiring system are also saved.

(See Appendix C for detailed information of the output file).

7.4 Configuration file

The Configuration file is an ASCII file used to properly run LAS. It contains the output and input folders paths, the library folder path and some graphical options. An example of Configuration file is shown in Appendix D.

7.5 CIMEL and PREDE-POM atmospheric optical depth files

LAS can visualize CIMEL and PREDE-POM measured atmospheric optical depth (AOD) on the auxiliary graph, selecting tau button. Each instrument has its own AOD file in netcdf format. Examples of these files are shown in Appendix E and F.

8. APPENDIX

8.1 Appendix A: Input Signal file

The input signal file is a NETCDF file containing all information about location, system and measurement as global attributes, and the averaged signal profiles.

File "rome_010min_532Hitan_20190927084900.nc"

File type: NetCDF-3/CDM

```
{
 dimensions:
  npnt = 3000;
  nrec = UNLIMITED;   // (29 currently)
 variables:
  double time(nrec=29);
   :Units = "MJD2K";
   :LongName = "Time";

  double startime(nrec=29);
   :LongName = "StartTime";
   :Units = "MJD2K";

  double endtime(nrec=29);
   :LongName = "EndTime";
   :Units = "MJD2K";

  int nsht(nrec=29);
   :Units = " ";
   :LongName = "LaserShots";

  float ch(nrec=29, npnt=3000);
   :Units = "a.u.";
   :LongName = "AveragedSignal";
   :VPMT_V = 1000; // int
   :Wavelength_nm = 532; // int
   :Polarisation = 0; // int
   :bin_number = 3000; // int
   :Vertical_resolution_m = 7.5; // double
   :Range_Discriminator = 1; // int
   :Licel_channel = 2; // int
   :Trigger_delay = 1.0; // double

  float err(nrec=29, npnt=3000);
   :LongName = "SignalStandardDeviation";
   :Units = "a.u.";

 // global attributes:
 :System = "UNIROMA1 LIDAR         ";
 :Location = "Rome-Italy             ";
 :Longitude = "41.9                    ";
 :Latitude = "12.5                    ";
 :Altitude = "75.                     ";
 :SystemDescription = "Rayleigh-Raman Monostatic LIDAR      ";
 :Laser = "Quanta Ray          ";
 :RepetitionRate = "30                       ";
 :Energyperpulse = "NA                        ";
 :Emissionwavelength = "1064-532-355                  ";
```

```
 :Detectedwavelength = "532\t\t              ";
 :Zenithangle = "0                          ";
 :Acquisitionsystem = "LICEL-TR20-160               ";
 :chdescription = "532 total high range analog    ";
 :telescope = "100mm Cassegrain                  ";
 :filter = "0.3 nm BW filters              ";
 :Comments = "TEST 2018                    ";
 :_CoordSysBuilder = "ucar.nc2.dataset.conv.DefaultConvention";
}
```

8.2 Appendix B: the meteorological file

16245 LIRE Pratica Di Mare Observations at 12Z 24 Sep 2018

```
-------------------------------------------------------------------------------
PRES  HGHT  TEMP  DWPT  RELH  MIXR  DRCT  SKNT  THTA  THTE  THTV
hPa     m     C     C     %  g/kg   deg  knot     K    KK
-------------------------------------------------------------------------------
1011.    32  26.4  19.4    65 14.23   190     4 298.6 340.2 301.2
1000.   127  25.2  19.2    69 14.21   195     8 298.4 339.9 300.9
929.0   769  19.2  17.0    87 13.30   115     6 298.6 337.4 300.9
...
 7.1 33508 -39.9 -80.9     0  0.08   318    11 958.8 960.0 958.9
 7.0 33604 -40.2 -81.2     0  0.08   325    10 961.7 962.8 961.7
```

Station information and sounding indices

```
                    Station identifier: LIRE
                      Station number: 16245
                     Observation time: 180924/1200
                     Station latitude: 41.65
                     Station longitude: 12.43
                     Station elevation: 32.0
                      Showalter index: 2.08
                        Lifted index: 0.10
    LIFT computed using virtual temperature: -0.58
                        SWEAT index: 187.00
                          K index: 22.70
                    Cross totals index: 20.10
                  Vertical totals index: 21.70
                   Totals totals index: 41.80
      Convective Available Potential Energy: 627.37
           CAPE using virtual temperature: 724.36
                  Convective Inhibition: -7.07
           CINS using virtual temperature: -3.75
Equilibrum Level: 209.85
Equilibrum Level using virtual temperature: 209.73
               Level of Free Convection: 879.18
           LFCT using virtual temperature: 887.53
                  Bulk Richardson Number: 18.50
         Bulk Richardson Number using CAPV: 21.37
  Temp [K] of the Lifted Condensation Level: 290.61
Pres [hPa] of the Lifted Condensation Level: 911.31
     Mean mixed layer potential temperature: 298.44
           Mean mixed layer mixing ratio: 13.98
           1000 hPa to 500 hPa thickness: 5733.00
Precipitable water [mm] for entire sounding: 34.99
```

8.3 Appendix C: the Output file

The LAS's output file contains information about location, instrument and measurement in addition to the variables retrieved during analysis process

File "rome_010min_532MTAN_back_20190927085000_v03.nc"

File type: NetCDF-3/CDM

```
{
 dimensions:
  nbin = 3000;
  nprof = UNLIMITED;   // (29 currently)
  maxnlayers = 5;
  factor_bottom_top = 2;
 variables:
  double DATETIME(nprof=29);
   :VAR_NAME = "DATETIME";
   :VAR_SIZE = "29";
   :VAR_DESCRIPTION = "Date and time of the measured profiles in ut";
   :VAR_NOTES = "universal time in days since 2000-01-01";
   :VAR_DEPEND = "DATETIME";
   :VAR_DATA_TYPE = "DOUBLE";
   :VAR_UNITS = "MJD2K";
   :VAR_SI_CONVERSION = "0.0;86400.0;s";
   :VAR_VALID_MIN = -9.99999999999E10; // double
   :VAR_VALID_MAX = 9.99999999999E10; // double
   :VAR_FILL_VALUE = -9.999E15; // double

  float ALTITUDE(nbin=3000);
   :VAR_NAME = "ALTITUDE";
   :VAR_SIZE = "3000";
   :VAR_DESCRIPTION = "Altitude of the profile elements";
   :VAR_NOTES = "The altitude from the sea level ...";
   :VAR_DEPEND = "ALTITUDE";
   :VAR_DATA_TYPE = "REAL";
   :VAR_UNITS = "m";
   :VAR_SI_CONVERSION = "0.0;1.0;m";
   :VAR_VALID_MIN = 0.0f; // float
   :VAR_VALID_MAX = 22567.5f; // float
   :VAR_FILL_VALUE = -9999.9f; // float

  float INTEGRATION.TIME(nprof=29);
   :VAR_NAME = "INTEGRATION.TIME";
   :VAR_SIZE = "29";
   :VAR_DESCRIPTION = "Integration Time of the single profiles";
   :VAR_NOTES = "";
   :VAR_DEPEND = "DATETIME";
   :VAR_DATA_TYPE = "REAL";
   :VAR_UNITS = "h";
   :VAR_SI_CONVERSION = "0.0;3600.0;s";
   :VAR_VALID_MIN = 0.0f; // float
   :VAR_VALID_MAX = 300000.0f; // float
   :VAR_FILL_VALUE = -9999.9f; // float

  float LATITUDE.INSTRUMENT;
   :VAR_NAME = "LATITUDE.INSTRUMENT";
   :VAR_SIZE = "1";
   :VAR_DESCRIPTION = "Latitude of the LIDAR";
   :VAR_NOTES = "";
   :VAR_DEPEND = "CONSTANT";
```

:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "deg";
:VAR_SI_CONVERSION = "0.0;1.74533E-2;rad";
:VAR_VALID_MIN = -90.0f; // float
:VAR_VALID_MAX = 90.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

float LONGITUDE.INSTRUMENT;
:VAR_NAME = "LONGITUDE.INSTRUMENT";
:VAR_SIZE = "1";
:VAR_DESCRIPTION = "longitude of the LIDAR";
:VAR_NOTES = "";
:VAR_DEPEND = "CONSTANT";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "deg";
:VAR_SI_CONVERSION = "0.0;1.74533E-2;rad";
:VAR_VALID_MIN = -180.0f; // float
:VAR_VALID_MAX = 180.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

float ALTITUDE.INSTRUMENT;
:VAR_NAME = "ALTITUDE.INSTRUMENT";
:VAR_SIZE = "1";
:VAR_DESCRIPTION = "altitude of the LIDAR";
:VAR_NOTES = "";
:VAR_DEPEND = "CONSTANT";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0.0;1.0;m";
:VAR_VALID_MIN = -9999.0f; // float
:VAR_VALID_MAX = 99999.0f; // float
:VAR_FILL_VALUE = -99999.9f; // float

double DATETIME.START(nprof=29);
:VAR_NAME = "DATETIME.START";
:VAR_SIZE = "29";
:VAR_DESCRIPTION = "start time of the measurements";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "MJD2K";
:VAR_SI_CONVERSION = "0.0;86400.0;s";
:VAR_VALID_MIN = -9.99999999999E10; // double
:VAR_VALID_MAX = 9.99999999999E10; // double
:VAR_FILL_VALUE = -9999.9; // double

double DATETIME.STOP(nprof=29);
:VAR_NAME = "DATETIME.STOP";
:VAR_SIZE = "29";
:VAR_DESCRIPTION = "stop time of the measurements";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "MJD2K";
:VAR_SI_CONVERSION = "0.0;86400.0;s";
:VAR_VALID_MIN = -9.99999999999E10; // double
:VAR_VALID_MAX = 9.99999999999E10; // double
:VAR_FILL_VALUE = -9999.9; // double

int WAVELENGTH_EMISSION;
:VAR_NAME = "WAVELENGTH_EMISSION";

:VAR_SIZE = "1";
:VAR_DESCRIPTION = "emitted wavelenght";
:VAR_NOTES = "";
:VAR_DEPEND = "CONSTANT";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "nm";
:VAR_SI_CONVERSION = "0;1E-9;m";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 99999; // int
:VAR_FILL_VALUE = -9999; // int

int WAVELENGTH_DETECTION;
:VAR_NAME = "WAVELENGTH_DETECTION";
:VAR_SIZE = "1";
:VAR_DESCRIPTION = "detected wavelenght";
:VAR_NOTES = "";
:VAR_DEPEND = "CONSTANT";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "nm";
:VAR_SI_CONVERSION = "0;1E-9;m";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 99999; // int
:VAR_FILL_VALUE = -9999; // int

double VOLUME.BACKSCATTER.RATIO(nprof=29, nbin=3000);
:VAR_NAME = "VOLUME.BACKSCATTER.RATIO";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "(molecular backscattering + aerosol backscattering)/ molecular backscattering";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0.0;1;1";
:VAR_VALID_MIN = -1000.0; // double
:VAR_VALID_MAX = 9999.0; // double
:VAR_FILL_VALUE = -9999.9; // double

float AEROSOL.EXTINCTION.COEFFICIENT(nprof=29, nbin=3000);
:VAR_NAME = "AEROSOL.EXTINCTION.COEFFICIENT";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "aerosol absorption coefficient";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "m-1";
:VAR_SI_CONVERSION = "0.0;1.0;m-1";
:VAR_VALID_MIN = -999.0f; // float
:VAR_VALID_MAX = 9999.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

float AEROSOL.BACKSCATTER.COEFFICIENT(nprof=29, nbin=3000);
:VAR_NAME = "AEROSOL.BACKSCATTER.COEFFICIENT";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "aerosol backscattering coefficient";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "m-1 sr-1";
:VAR_SI_CONVERSION = "0.0;1.0;sr-1 m-1";
:VAR_VALID_MIN = -999.0f; // float
:VAR_VALID_MAX = 9999.0f; // float

:VAR_FILL_VALUE = -9999.9f; // float

float AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.COMBINED.STANDARD(nprof=29, nbin=3000);
:VAR_NAME = "AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.COMBINED.STANDARD";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "aerosol backscattering coefficient uncertainty";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "m-1 sr-1";
:VAR_SI_CONVERSION = "0.0;1.0;sr-1 m-1";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 9999.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

double VOLUME.BACKSCATTER.RATIO_UNCERTAINTY.COMBINED.STANDARD(nprof=29, nbin=3000);
:VAR_NAME = "VOLUME.BACKSCATTER.RATIO_UNCERTAINTY.COMBINED.STANDARD";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "backscattering ratio uncertainty";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0.0;1;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 9999.0; // double
:VAR_FILL_VALUE = -9999.9; // double

float AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.RANDOM.STANDARD(nprof=29, nbin=3000);
:VAR_NAME = "AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.RANDOM.STANDARD";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "backscattering coefficient random uncertainty";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "m-1 sr-1";
:VAR_SI_CONVERSION = "0.0;1.0;sr-1 m-1";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 9999.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

float AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.SYSTEMATIC.STANDARD(nprof=29, nbin=3000);
:VAR_NAME = "AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.SYSTEMATIC.STANDARD";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "backscattering ratio systematic uncertainty";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "m-1 sr-1";
:VAR_SI_CONVERSION = "0.0;1.0;sr-1 m-1";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 9999.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

float AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.COMBINED.STANDARD(nprof=29, nbin=3000);
:VAR_NAME = "AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.COMBINED.STANDARD";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "aerosol absorption coefficient uncertainty";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "REAL";

:VAR_UNITS = "m-1";
:VAR_SI_CONVERSION = "0.0;1.0;m-1";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 999.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

float AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.RANDOM.STANDARD(nprof=29, nbin=3000);
:VAR_NAME = "AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.RANDOM.STANDARD";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "aerosol absorption coefficient random uncertainty";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "m-1";
:VAR_SI_CONVERSION = "0.0;1.0;m-1";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 999.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

float AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.SYSTEMATIC.STANDARD(nprof=29, nbin=3000);
:VAR_NAME = "AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.SYSTEMATIC.STANDARD";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "aerosol absorption coefficient systematic uncertainty";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "m-1";
:VAR_SI_CONVERSION = "0.0;1.0;m-1";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 999.0f; // float
:VAR_FILL_VALUE = -9999.9f; // float

double SIGNAL(nprof=29, nbin=3000);
:VAR_NAME = "SIGNAL";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "Signal analyzed";
:VAR_NOTES = "The signal analyzed to produce the different results";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "";
:VAR_SI_CONVERSION = "0.0;1;";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 22567.5f; // float
:VAR_FILL_VALUE = 0.0f; // float

double SIGNAL_UNCERTAINTY.COMBINED.STANDARD(nprof=29, nbin=3000);
:VAR_NAME = "SIGNAL_UNCERTAINTY.COMBINED.STANDARD";
:VAR_SIZE = "29;3000";
:VAR_DESCRIPTION = "Signal analyzed total uncertainty";
:VAR_NOTES = "";
:VAR_DEPEND = "DATETIME;ALTITUDE";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "";
:VAR_SI_CONVERSION = "0.0;1;";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 22567.5f; // float
:VAR_FILL_VALUE = 0.0f; // float

double CLOUD.TOP.HEIGHT(nprof=29, maxnlayers=5);
:VAR_NAME = "CLOUD.TOP.HEIGHT";
:VAR_DESCRIPTION = "The altitudes used as top of the clouds in the calculation";

:VAR_NOTES = "The top altitude of each cloud used in the profile calculation. The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

double CLOUD.BOTTOM.HEIGHT(nprof=29, maxnlayers=5);
:VAR_NAME = "CLOUD.BOTTOM.HEIGHT";
:VAR_DESCRIPTION = "The altitudes used as bottom of the cloud layer in the calculation";
:VAR_NOTES = "The bottom altitude of each cloud layer used in the profile computation. The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

double ALTITUDE.OVERLAP(nprof=29);
:VAR_NAME = "ALTITUDE.OVERLAP";
:VAR_DESCRIPTION = "The altitudes used as top for extrapolation to ground of the aerosol ratio ";
:VAR_NOTES = "The extrapolation is used only for the AOD calculation ";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

double AEROSOL.SCALE(nprof=29);
:VAR_NAME = "AEROSOL.SCALE";
:VAR_DESCRIPTION = "The altitudes scale used for extrapolation to ground of the aerosol ratio ";
:VAR_NOTES = "The extrapolation is used only for the AOD calculation ";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

double AEROSOL.LAYER.BOTTOM.HEIGHT(nprof=29, maxnlayers=5);
:VAR_NAME = "AEROSOL.LAYER.BOTTOM.HEIGHT";
:VAR_DESCRIPTION = "The altitudes used as bottom of the aerosol layers in the calculation";
:VAR_NOTES = "The bottom altitude of each aerosol layer used in the profile calculation. The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";

:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

double AEROSOL.LAYER.TOP.HEIGHT(nprof=29, maxnlayers=5);
:VAR_NAME = "AEROSOL.LAYER.TOP.HEIGHT";
:VAR_DESCRIPTION = "The altitudes used as top of the aerosol layers in the calculation";
:VAR_NOTES = "Top altitude of each aerosol layer used in the profile calculation. The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

double AEROSOL.LIDAR.FACTOR.HEIGHT(nprof=29, factor_bottom_top=2);
:VAR_NAME = "AEROSOL.LIDAR.FACTOR.HEIGHT";
:VAR_DESCRIPTION = "Respectively the bottom and top altitudes of the vertical profile layer used to compute the LIDAR factor in the calculation";
:VAR_NOTES = "The botttom (first element) and top (second element) altitude of the layer used in order to compute the factor";
:VAR_SIZE = "29;2";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

float AEROSOL.LIDAR.RATIO(nprof=29, maxnlayers=5);
:VAR_NAME = "AEROSOL.LIDAR.RATIO";
:VAR_DESCRIPTION = "The LIDAR Ratio used in the computation for each aerosol layer";
:VAR_NOTES = "The LIDAR Ratio used in the computation for each aerosol layer (sorted according to increasing altitude). The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "ALTITUDE";
:VAR_DATA_TYPE = "REAL";
:VAR_UNITS = "sr";
:VAR_SI_CONVERSION = "0.0;1.0;sr";
:VAR_VALID_MIN = 0.0f; // float
:VAR_VALID_MAX = 9999.9f; // float
:VAR_FILL_VALUE = -9999.9f; // float

double CLOUD.LIDAR.RATIO(nprof=29, maxnlayers=5);
:VAR_NAME = "CLOUD.LIDAR.RATIO";
:VAR_DESCRIPTION = "The LIDAR Ratio retrieved in the computation for each cloud layer";
:VAR_NOTES = "The LIDAR Ratio retrieved in the computation for each cloud layer (sorted according to increasing altitude). The clouds\' LR is a result of the computation and not a parameter.The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "sr";
:VAR_SI_CONVERSION = "0.0;1.0;sr";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 9999.9; // double

:VAR_FILL_VALUE = -9999.9; // double

double CLOUD.LIDAR.RATIO_UNCERTAINTY.COMBINED.STANDARD(nprof=29, maxnlayers=5);
:VAR_NAME = "CLOUD.LIDAR.RATIO_UNCERTAINTY.COMBINED.STANDARD";
:VAR_DESCRIPTION = "The uncertainty of the LIDAR Ratio retrieved in the computation for each cloud layer";
:VAR_NOTES = "The uncertainty of the LIDAR Ratio retrieved in the computation for each cloud layer (sorted according to increasing altitude). The clouds\' LR is a result of the computation and not a parameter.The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0.0;1.0;sr";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 9999.9; // double
:VAR_FILL_VALUE = -9999.9; // double

int CLOUD.FLAG(nprof=29, maxnlayers=5);
:VAR_NAME = "CLOUD.FLAG";
:VAR_DESCRIPTION = "The computation mode used to retrieve the cloud";
:VAR_NOTES = "0: single-cloud computation; 1: cloud between two aerosol layer; 2: cloud at the top of the aerosol layer";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 2; // int
:VAR_FILL_VALUE = -1; // int

double CLOUD.OPTICAL.DEPTH(nprof=29, maxnlayers=5);
:VAR_NAME = "CLOUD.OPTICAL.DEPTH";
:VAR_DESCRIPTION = "The optical depth retrieved in the computation for each cloud layer";
:VAR_NOTES = "The optical depth retrieved in the computation for each cloud layer (sorted according to increasing altitude). The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 10.0; // double
:VAR_FILL_VALUE = -9999.9; // double

double CLOUD.OPTICAL.DEPTH_UNCERTAINTY.COMBINED.STANDARD(nprof=29, maxnlayers=5);
:VAR_NAME = "CLOUD.OPTICAL.DEPTH_UNCERTAINTY.COMBINED.STANDARD";
:VAR_DESCRIPTION = "The uncertainty optical depth calculated for each cloud layer";
:VAR_NOTES = "The uncertainty of the optical depth calculated for each cloud layer (sorted according to increasing altitude). The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 10.0; // double
:VAR_FILL_VALUE = -9999.9; // double

double AEROSOL.OPTICAL.DEPTH(nprof=29, maxnlayers=5);
:VAR_NAME = "AEROSOL.OPTICAL.DEPTH";
:VAR_DESCRIPTION = "The optical depth retrieved in the computation for each aerosol layer";
:VAR_NOTES = "The optical depth retrieved in the computation for each aerosol layer (sorted according to increasing altitude). The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 10.0; // double
:VAR_FILL_VALUE = -9999.9; // double

double AEROSOL.OPTICAL.DEPTH_UNCERTAINTY.COMBINED.STANDARD(nprof=29, maxnlayers=5);
:VAR_NAME = "AEROSOL.OPTICAL.DEPTH_UNCERTAINTY.COMBINED.STANDARD";
:VAR_DESCRIPTION = "The uncertainty of the optical depth retrieved in the computation for each aerosol layer";
:VAR_NOTES = "The uncertainty of the optical depth retrieved in the computation for each aerosol layer (sorted according to increasing altitude). The number of layers can be different for different profiles, depending on atmosphere evolution during measurements.";
:VAR_SIZE = "29;5";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 10.0; // double
:VAR_FILL_VALUE = -9999.9; // double

double SIGNAL.BACKGROUND(nprof=29);
:VAR_NAME = "SIGNAL.BACKGROUND";
:VAR_DESCRIPTION = "The background of the signal measured for each profile";
:VAR_NOTES = "";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 9.9E24; // double
:VAR_FILL_VALUE = -9999.9; // double

double SIGNAL.BACKGROUND_UNCERTAINTY.RANDOM.STANDARD(nprof=29);
:VAR_NAME = "SIGNAL.BACKGROUND_UNCERTAINTY.RANDOM.STANDARD";
:VAR_DESCRIPTION = "The background\'s standard deviation of the signal measured for each profile";
:VAR_NOTES = "";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 9.9E24; // double
:VAR_FILL_VALUE = -9999.9; // double

double LIDAR_CALIBRATION.FACTOR(nprof=29);
:VAR_NAME = "LIDAR_CALIBRATION.FACTOR";
:VAR_DESCRIPTION = "The LIDAR constant";
:VAR_NOTES = "";
:VAR_SIZE = "29";

:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 9.9E24; // double
:VAR_FILL_VALUE = -9999.9; // double

double LIDAR_CALIBRATION.FACTOR_UNCERTAINTY(nprof=29);
:VAR_NAME = "LIDAR_CALIBRATION.FACTOR_UNCERTAINTY";
:VAR_DESCRIPTION = "The LIDAR constant uncertainty";
:VAR_NOTES = "";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 9.9E24; // double
:VAR_FILL_VALUE = -9999.9; // double

double AEROSOL.LIDAR.FACTOR(nprof=29);
:VAR_NAME = "AEROSOL.LIDAR.FACTOR";
:VAR_DESCRIPTION = "The LIDAR factor calculated in order to analyze the profile";
:VAR_NOTES = "";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 9.9E24; // double
:VAR_FILL_VALUE = -9999.9; // double

double AEROSOL.LIDAR.FACTOR_UNCERTAINTY(nprof=29);
:VAR_NAME = "AEROSOL.LIDAR.FACTOR_UNCERTAINTY";
:VAR_DESCRIPTION = "The LIDAR factor standard deviation";
:VAR_NOTES = "";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 9.9E24; // double
:VAR_FILL_VALUE = -9999.9; // double

float TOTAL.OPTICAL.DEPTH(nprof=29);
:VAR_NAME = "TOTAL.OPTICAL.DEPTH";
:VAR_DESCRIPTION = "The total optical depth";
:VAR_NOTES = "";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "FLOAT";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 999.0; // double
:VAR_FILL_VALUE = -1.0; // double

float TOTAL.OPTICAL.DEPTH_UNCERTAINTY.COMBINED.STANDARD(nprof=29);

:VAR_NAME = "TOTAL.OPTICAL.DEPTH_UNCERTAINTY.COMBINED.STANDARD";
:VAR_DESCRIPTION = "The total optical depth uncertainty";
:VAR_NOTES = "";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "FLOAT";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 999.0; // double
:VAR_FILL_VALUE = -1.0; // double

int FLAG.ANALYZED(nprof=29);
:VAR_NAME = "FLAG.ANALYZED";
:VAR_DESCRIPTION = "A flag that marks if a certain profile was analyzed or not";
:VAR_NOTES = "Not yet analyzed = 0. Analyzed = 1";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 1; // int
:VAR_FILL_VALUE = -1; // int

int FLAG.COMPUTATION(nprof=29);
:VAR_NAME = "FLAG.COMPUTATION";
:VAR_DESCRIPTION = "A flag to recognize the computation mode used for the profiles";
:VAR_NOTES = "0 : normal factor computation; 1: factor fit computation";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 1; // int
:VAR_FILL_VALUE = -1; // int

int FLAG.MERGEDSIGNAL(nprof=29);
:VAR_NAME = "FLAG.MERGEDSIGNAL";
:VAR_DESCRIPTION = "A flag to recognize the merged profiles";
:VAR_NOTES = "0 : no merging operation; 1: signal merged S1S2, 2: signal merged S2S1";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 2; // int
:VAR_FILL_VALUE = -1; // int

double MERGINGLAYER.HEIGHT(nprof=29, factor_bottom_top=2);
:VAR_NAME = "MERGINGLAYER.HEIGHT";
:VAR_DESCRIPTION = "Respectively the bottom and top altitudes of the vertical profile layer used where the profiles were synchronized";
:VAR_NOTES = "The botttom (first element) and top (second element) altitude of the synch layer between the two profiles";
:VAR_SIZE = "29;2";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";

:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

int FLAG.SYNCHSIGNAL(nprof=29);
:VAR_NAME = "FLAG.SYNCHSIGNAL";
:VAR_DESCRIPTION = "A flag to recognize the synchronized profiles";
:VAR_NOTES = "0 : no merging operation; 1: signal S1 synchronized on S2, 2: signal S2 synchronized on S1";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 1; // int
:VAR_FILL_VALUE = -1; // int

double SYNCHLAYER.HEIGHT(nprof=29, factor_bottom_top=2);
:VAR_NAME = "SYNCHLAYER.HEIGHT";
:VAR_DESCRIPTION = "Respectively the bottom and top altitudes of the vertical profile layer used where the profiles were merged";
:VAR_NOTES = "The botttom (first element) and top (second element) altitude of the merging layer between the two profiles";
:VAR_SIZE = "29;2";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "DOUBLE";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -9999.9; // double

int SYNCH.BINSHIFT(nprof=29);
:VAR_NAME = "SYNCH.BINSHIFT";
:VAR_DESCRIPTION = "The shift of the signal in number of bins";
:VAR_NOTES = "Number produced by the use of the synch function on the analysis program";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0.0; // double
:VAR_VALID_MAX = 3000; // int
:VAR_FILL_VALUE = -9999.9; // double

int FLAG.CONVERGENCE(nprof=29);
:VAR_NAME = "FLAG.CONVERGENCE";
:VAR_DESCRIPTION = "A flag to recognize the convergence mode used for the profiles";
:VAR_NOTES = "0 : integral convergence; 1: punctual convergence";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 1; // int
:VAR_FILL_VALUE = -1; // int

float BACKGROUNDLAYER.HEIGHT(nprof=29, factor_bottom_top=2);
:VAR_NAME = "BACKGROUNDLAYER.HEIGHT";

:VAR_DESCRIPTION = "Respectively the bottom and top altitudes of the vertical profile layer used to calculate the background";
:VAR_NOTES = "if these values are [0, 0] the background was calulated using the last 100 points";
:VAR_SIZE = "29;2";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "FLOAT";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -1; // int

float BACKGROUND.MULTIPLIER(nprof=29);
:VAR_NAME = "BACKGROUND.MULTIPLIER";
:VAR_DESCRIPTION = "The value wich was eventually multiplied the original background computed by background fit or last 100 points procedure";
:VAR_NOTES = "if these values are [0, 0] the background was calulated using the last 100 points";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "FLOAT";
:VAR_UNITS = "%";
:VAR_SI_CONVERSION = "0;0.01;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 99999; // int
:VAR_FILL_VALUE = -1; // int

float SMOOTH.HEIGHT(nprof=29);
:VAR_NAME = "SMOOTH.HEIGHT";
:VAR_DESCRIPTION = "The altitude above which the results are eventually smoothed";
:VAR_NOTES = "if this value is 0 the entire profile is smoothed";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "FLOAT";
:VAR_UNITS = "m";
:VAR_SI_CONVERSION = "0;1;m";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = 22567.5; // double
:VAR_FILL_VALUE = -1; // int

int SMOOTH.BINS(nprof=29);
:VAR_NAME = "SMOOTH.BINS";
:VAR_DESCRIPTION = "The number of points averaged in the smoothing process";
:VAR_NOTES = "if these value is 0 no smoothing process occurred";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";
:VAR_VALID_MIN = 0; // int
:VAR_VALID_MAX = "3000";
:VAR_FILL_VALUE = -1; // int

int SHOTS.NUMBER(nprof=29);
:VAR_NAME = "SHOTS.NUMBER";
:VAR_DESCRIPTION = "The number of shots composing the signal";
:VAR_NOTES = "";
:VAR_SIZE = "29";
:VAR_DEPEND = "DATETIME";
:VAR_DATA_TYPE = "INTEGER";
:VAR_UNITS = "1";
:VAR_SI_CONVERSION = "0;1;1";

```
    :VAR_VALID_MIN = 0; // int
    :VAR_VALID_MAX = "9e+24";
    :VAR_FILL_VALUE = -1; // int

 // global attributes:
 :PI_NAME = "Cacciani;Marco";
 :PI_AFFILIATION = "Universita di Roma \"La Sapienza\" - BAQUNIN;UROMA.BAQUNIN";
 :PI_ADDRESS = "Dipartimento di Fisica, Universita Sapienza, Piazzale Aldo Moro 2;Rome, 00185;ITALY";
 :PI_EMAIL = "marco.cacciani@uniroma1.it";
 :DO_NAME = "BAQUNIN;UROMA.BAQUNIN";
 :DO_AFFILIATION = "Universita di Roma \"La Sapienza\" - BAQUNIN;UROMA.BAQUNIN";
 :DO_ADDRESS = "Dipartimento di Fisica, Universita Sapienza, Piazzale Aldo Moro 2;Rome, 00185;ITALY";
 :DO_EMAIL = "marco.cacciani@uniroma1.it";
 :DS_NAME = "Cacciani;Marco";
 :DS_AFFILIATION = "BAQUNIN;UROMA.BAQUNIN";
 :DS_ADDRESS = "Dipartimento di Fisica, Universita Sapienza, Piazzale Aldo Moro 2;Rome, 00185;ITALY";
 :DS_EMAIL = "marco.cacciani@uniroma1.it";
 :DATA_DESCRIPTION = "LIDAR measurements of aerosol backscattering  at Rome";
 :DATA_DISCIPLINE = "ATMOSPHERIC.PHYSICS;REMOTE.SENSING;GROUNDBASED";
 :DATA_GROUP = "EXPERIMENTAL;PROFILE.STATIONARY";
 :DATA_LOCATION = "ROME.SAPIENZA";
 :DATA_SOURCE = "LIDAR.AEROSOL_UROMA.BAQUNIN001";
 :DATA_VARIABLES = "
DATETIME;ALTITUDE;INTEGRATION.TIME;LATITUDE.INSTRUMENT;LONGITUDE.INSTRUMENT;ALTITUDE.INSTRUMENT;
DATETIME.START;DATETIME.STOP;WAVELENGTH_EMISSION;WAVELENGTH_DETECTION;
VOLUME.BACKSCATTER.RATIO;AEROSOL.EXTINCTION.COEFFICIENT;AEROSOL.BACKSCATTER.COEFFICIENT;
AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.COMBINED.STANDARD;
VOLUME.BACKSCATTER.RATIO_UNCERTAINTY.COMBINED.STANDARD;
AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.RANDOM.STANDARD;
AEROSOL.BACKSCATTER.COEFFICIENT_UNCERTAINTY.SYSTEMATIC.STANDARD;
AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.COMBINED.STANDARD;
AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.RANDOM.STANDARD;
AEROSOL.EXTINCTION.COEFFICIENT_UNCERTAINTY.SYSTEMATIC.STANDARD;
SIGNAL;CLOUD.TOP.HEIGHT;CLOUD.BOTTOM.HEIGHT;AEROSOL.LAYER.BOTTOM.HEIGHT;AEROSOL.LAYER.TOP.HEIGH
T;       AEROSOL.LIDAR.FACTOR.HEIGHT;AEROSOL.LIDAR.RATIO;CLOUD.LIDAR.RATIO;
CLOUD.LIDAR.RATIO_UNCERTAINTY.COMBINED.STANDARD;CLOUD.FLAG;
CLOUD.OPTICAL.DEPTH;CLOUD.OPTICAL.DEPTH_UNCERTAINTY.COMBINED.STANDARD;
AEROSOL.OPTICAL.DEPTH;AEROSOL.OPTICAL.DEPTH_UNCERTAINTY.COMBINED.STANDARD;
SIGNAL.BACKGROUND;SIGNAL.BACKGROUND_UNCERTAINTY.RANDOM.STANDARD;
LIDAR_CALIBRATION.FACTOR;LIDAR_CALIBRATION.FACTOR_UNCERTAINTY;
AEROSOL.LIDAR.FACTOR;AEROSOL.LIDAR.FACTOR_UNCERTAINTY;
FLAG.ANALYZED;FLAG.COMPUTATION;FLAG.MERGEDSIGNAL;MERGINGLAYER.HEIGHT;FLAG.SYNCHSIGNAL;
SYNCHLAYER.HEIGHT;SYNCH.BINSHIFT;FLAG.CONVERGENCE;
BACKGROUNDLAYER.HEIGHT;BACKGROUND.MULTIPLIER;       SMOOTH.HEIGHT;SMOOTH.BINS;SHOTS.NUMBER";
 :DATA_START_DATE = "20190927T085000Z";
 :DATA_STOP_DATE = "20190927T133000Z";
 :DATA_FILE_VERSION = "003";
 :DATA_MODIFICATIONS = "";
 :DATA_CAVEATS = "";
 :DATA_RULES_OF_USE = "";
 :DATA_ACKNOWLEDGEMENT = "";
 :DATA_QUALITY = "";
 :DATA_TEMPLATE = "GEOMS-TE-LIDAR-AEROSOL-004";
 :DATA_PROCESSOR = "";
 :FILE_NAME = "rome_010min_532MTAN_back_20190927085000_v03.nc";
 :FILE_GENERATION_DATE = "20191129T155414Z";
 :FILE_ACCESS = "ARCHIVE;EVDC";
 :FILE_PROJECT_ID = "";
 :FILE_DOI = "";
 :FILE_ASSOCIATION = "";
 :FILE_META_VERSION = "04R045;CUSTOM";
```

:LASER_WAVELENGHT = 532.0; // double
:SIGNAL_FILE =
"F:/dati/LIDAR/dati_mediati/20190927/rome_010min_532Hitan_20190927084900.nc;F:/dati/LIDAR/dati_mediati/20190927/rome_010min_532tan_20190927084900.nc";
:VERTICAL_RESOLUTION_M = 7.5; // double
:REPETITION_RATE = 30.0; // double
:LAS_VERSION = 14.3; // double
:_CoordSysBuilder = "ucar.nc2.dataset.conv.DefaultConvention";
}

8.4 Appendix D: Configuration file

SET THE PATHS

inpath = # the folder of the input files
f:/dati/LIDAR/dati_mediati
meteopath = # the folder containing the metereological files
f:/dati/Radiosondaggi
savefolder = # the folder in which save the output files
f:/dati/LIDAR/ANALISI/
routinepath # the folder in which are stored the python routine julianday.py and smooth.py
f:/codici_Python3/funzioni/

SET THE GRAPHS OPTIONS

start_z_limit = #the maximum default altitude on the principal graph x axis. It can be changed using the zoom buttons
18
y_minlim = #the minimum default value on the principal graph y axis. It can be changed using the zoom buttons
1e1
graphmarker = #the graphics appereance: set “-“ for continuos line, “.“ for dots markers, “-.” for continuos line and dots markers
-
linewidth = #the graphics line width
1
markerwidth = #the graphichs markers thickness
6

SET THE WINDOW OPTIONS (left blank for default value)

rootwidth = #LAS GUI width (default = 1920)

rootheight = #LAS GUI height (default = 1000)

sizescrittura = #font size (default = 12)
13
ratio_schermo = #percentage of the screen occupied by LAS GUI (default = 95)

8.5 Appendix E: CIMEL AOD file

File "20190927_20190927_Rome_La_Sapienza_CIMEL_lev15.nc"

{
dimensions:
samples = 164;
samples1 = 164;

```
variables:
 double jd_AOD(samples=164);
  :long_name = "TIME";
  :units = "MJD2K";

 double jd_SDA(samples1=164);
  :long_name = "TIME";
  :units = "MJD2K";

 double ZenithAngle(samples=164);
  :long_name = "Absolute pointing zenith angle";
  :units = "degrees";

 double AMF(samples=164);
  :long_name = "Direct sun air mass factor";
  :units = "1";

 float AOD1640(samples=164);
  :long_name = "1640 nm Aerosol Optical Depth";
  :units = "1";
  :Triplet_Variability = 3.02E-4; // double
  :exact_wav = 1.6392; // double

 float AOD1020(samples=164);
  :long_name = "1020 nm Aerosol Optical Depth";
  :units = "1";
  :Triplet_Variability = 5.3E-5; // double
  :exact_wav = 1.0193; // double

 float AOD870(samples=164);
  :long_name = "870 nm Aerosol Optical Depth";
  :units = "1";
  :Triplet_Variability = 1.75E-4; // double
  :exact_wav = 0.8689; // double

 float AOD675(samples=164);
  :long_name = "675 nm Aerosol Optical Depth";
  :units = "1";
  :Triplet_Variability = 2.04E-4; // double
  :exact_wav = 0.6747; // double

 float AOD500(samples=164);
  :long_name = "500 nm Aerosol Optical Depth";
  :units = "1";
  :Triplet_Variability = 3.46E-4; // double
  :exact_wav = 0.5009; // double

 float AOD440(samples=164);
  :long_name = "440 nm Aerosol Optical Depth";
  :units = "1";
  :Triplet_Variability = 5.48E-4; // double
  :exact_wav = 0.4392; // double

 float AOD380(samples=164);
  :long_name = "380 nm Aerosol Optical Depth";
  :units = "1";
  :Triplet_Variability = 0.001017; // double
  :exact_wav = 0.3798; // double

 float AOD340(samples=164);
  :long_name = "340 nm Aerosol Optical Depth";
```

```
    :units = "1";
    :Triplet_Variability = 0.003125; // double
    :exact_wav = 0.3405; // double

  float PW(samples=164);
    :long_name = "Precipitable_Water";
    :units = "cm";
    :Triplet_Variability = 0.029166; // double
    :exact_wav = 0.9368; // double

  float AE_440_870(samples=164);
    :long_name = "440-870_Angstrom_Exponent";
    :units = "1";

  float AE_380_500(samples=164);
    :long_name = "380-500_Angstrom_Exponent";
    :units = "1";

  float AE_440_675(samples=164);
    :long_name = "440-675_Angstrom_Exponent";
    :units = "1";

  float AE_340_440(samples=164);
    :long_name = "340-440_Angstrom_Exponent";
    :units = "1";

  float O3(samples=164);
    :long_name = "Ozone";
    :units = "Dobson";

  float NO2(samples=164);
    :long_name = "NO2";
    :units = "Dobson";

  float tau_c(samples1=164);
    :long_name = "Coarse_Mode_AOD_500nm";
    :units = "1";

  float tau_f(samples1=164);
    :long_name = "Fine_Mode_AOD_500nm";
    :units = "1";

  float eta(samples1=164);
    :long_name = "FineModeFraction_500nm";
    :units = "1";

  float Sensor_Temp(samples=164);
    :long_name = "Sensor temperature";
    :units = "degree_C";

 // global attributes:
 :_CoordSysBuilder = "ucar.nc2.dataset.conv.DefaultConvention";
```

8.6 Appendix F: PREDE-POM AOD file

File "201909_AE_POM%2311_APL_1.5.nc"

```
{
 dimensions:
   samples = 2926;
 variables:
```

```
  double DATETIME(samples=2926);
    :long_name = "Time";
    :units = "MJD2K";

  float AE(samples=2926);
    :long_name = "Angström exponent";
    :units = "1";

 // global attributes:
 :RetrievedParameter = "AE";
 :System = "Prede_Pom01L#11";
 :PI = "Monica Campanelli";
 :PI_Contact = "campanellimonica@gmail.com";
 :Location = "Sapienza University, Rome, ITA";
 :Longitude_degrees_east = "12.515773";
 :Latitude_degrees_north = "41.901695";
 :Altitude_meter_asl = "83";
 :List_variables = "Date,Time,AE";
 :Software = "dsproc_39; dsform_35";
 :Level = "1.5";
 :EvaluationMethod = "ESR sunrad pack";
 :Detection_mode = "DS: Direct sun measurements";
 :Comments = "Cloud screened";
 :DetectedWavelength_nm = "340, 400, 500, 675, 870, 1020 nm";
 :ProcessedWavelength_nm = "340, 400, 500, 675, 870, 1020 nm";
 :Equivalentbandwidths_nm = "2 nm (UV), 10 nm (VIS)";
 :_Fillvalue = -999.99; // double
 :_CoordSysBuilder = "ucar.nc2.dataset.conv.DefaultConvention";
```

8.7 Appendix G: keyboard shortcuts

```
    "<Prior>", prevprofile
    "<Next>", nextprofile
    "<Alt-f>", calculate_factor
    "<space>", automatic_analysis
    "<Alt-s>", plot_signal
    "<Alt-r>", ratio
    "<Alt-a>", aerosol_command
    "<Alt-c>", cloud_command
    "<Alt-b>", calculate_background_fit
    "<Alt-o>", open_nc
    "<Escape>", _quit
    "<Alt-Key-->", background_down
    "<Alt-+>", background_up
    "<Alt-Control-+>", background_upup
    "<Alt-Control-Key-->", background_downdown
    "<Alt-Control-Key-a>", plot_alpha
    "<Alt-Control-Key-b>", plot_beta
    "<Alt-Control-Key-s>", average_signal
    "<Alt-Control-Key-r>", average_ratio
    "<F1>", save
```

9. REFERENCES

R1. Kovalev V. A., Eichinger W. E.: *Elastic Lidar: theory, practice and analysis methods*, Wiley, 2004.

R2. Russell, P. B., Swissler, T. J., and McCormick, M. P.: *Methodology for error analysis and simulation of lidar aerosol measurements*, Appl. Opt. **18**, 3783-3797, 1979.

R3. JCGM, J.: *Evaluation of measurement data—Guide to the expression of uncertainty in measurement*. Int. Organ. Stand. Geneva ISBN, 50, 134, 2008.

R4. The Python Language reference, available at: https://docs.python.org/3/reference

R5. Anaconda 3 documentation, available at https://docs.anaconda.com/anaconda/

R6. NETCDF4 library documentation, available at https://unidata.github.io/netcdf4-python/netCDF4

R7. GEOMS documentation: https://evdc.esa.int/documentation/geoms/

R8. tkinter documentation: https://docs.python.org/3.8/library/tkinter.html

R9. Matplotlib documentation: https://docs.python.org/3.8/library/tkinter.html

R10. Datetime documentation: https://docs.python.org/3/library/datetime.html

R11. Numpy documentation: https://numpy.org/doc/

End of Document