**Q**uantification of **l**idar (hardware) **unc**ertainties

**Qlunc**

Tags: wind lidar, hardware uncertainty, photonics module, optics module, OpenLidar

Author: Costa, Francisco[[1]](#footnote-1)

Introduction

Measuring uncertainty means doubt about the validity of the result of a measurement [1] or, in other words, it represents the dispersion of the values attributed to a measurand. The importance of knowing uncertainty in measurements lies on both, the quality of the measurement and understanding of the results, and it can have a huge impact on the veracity of an experiment or measuring set up, hence in decision-making processes based on the experiment outcomes. In the wind energy community wind lidar devices have been widely used to characterize the wind and research for most suitable sites, with best wind conditions, but also in wind forecasting, wind turbine power performance, loads assessment and lidar-assisted turbine control [2]. Building a wind farm entails a huge effort and investment, so knowing beforehand site and wind key parameters is important to minimize risks and optimize energy acquisition. In this sense, lidar measurement uncertainties assessment plays a crucial role, since it can determine decision-making processes and therefore the global performance of a wind facility.

What’s Qlunc

Lidar is a remote sensing measuring device and, to increase confidence in its measurements, the uncertainty of the measuring data must be assessed. This project develops and implements an open-source, freely available uncertainty model that allows us to assess lidar measurement uncertainties for *profiling lidar* and *forward-looking nacelle-mounted lidar* before a lidar is built.

Inspired by the OpenLidar architecture [3], this model is a python-based tool called Qlunc for “Quantification of Lidar UNCertainties”, that aims to estimate the uncertainty of a wind lidar device, including hardware and data processing methods. It contains models of the uncertainty contributed by individual lidar components that are then combined to estimate the total uncertainty of the lidar device.

The code (Qlunc) has an objected-oriented structure taking advantage of python features; by using python objects and simulating real lidar components, the code puts all together in modules to eventually build up a lidar digital twin. Qlunc is meant to be as modular as possible and offers to the user the possibility of creating different lidar objects on parallel, with different components, simultaneously. This allows to easily combine different modules with different characteristics simulating different lidar devices and compare them against each other. Furthermore, it allows to easily integrate different uncertainty methods or interface external codes.

Each component, pertaining to correspondent module (e.g. photodetector belongs to the photonics module) is created as a python object and enclosed in other python class, which represents the aforementioned modules. Following this procedure these modules are, in turn, included in the lidar python class, which gathers all classes corresponding to the different modules a lidar is made of, thus creating the lidar digital twin. All components are characterized by their technical parameters and their uncertainty function, which is a stand-alone function evaluating each component uncertainty.

In the context of this project, two different uncertainties must be pointed out. When talking about uncertainties in the output signal, we stand for signal noise and SNR is used to characterize the error. On the other hand, uncertainties referred to as pointing accuracy errors are expressed through distance errors. Eventually, different uncertainties will be expressed in terms of percentages, getting a global percentage value, which will be considered as the total uncertainty estimation of lidar hardware.

# Qlunc available capabilities

At this stage, hardware uncertainties coming from specific lidar modules, namely photonics and optics, are under assessment.

Qlunc can work with VAD and scanning lidar patterns. Currently, Qlunc can calculate uncertainties coming from photonics module, including photodetector (with or without trans-impedance amplifier) and optical amplifier uncertainties, as well as optics module uncertainty including scanner pointing accuracy distance errors and optical circulator uncertainties. For each module, the Guide to the Expression of Uncertainty in Measurement ([GUM](https://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf)) [1] is applied to calculate uncertainty expansion through all components and modules. Note that included components and modules are considered uncorrelated. This is an important assumption since we can disregard cross-correlations terms among different elements in the lidar.

Plots can show different signal noise contributors in the photodetector components and scanning points including their distance uncertainty.

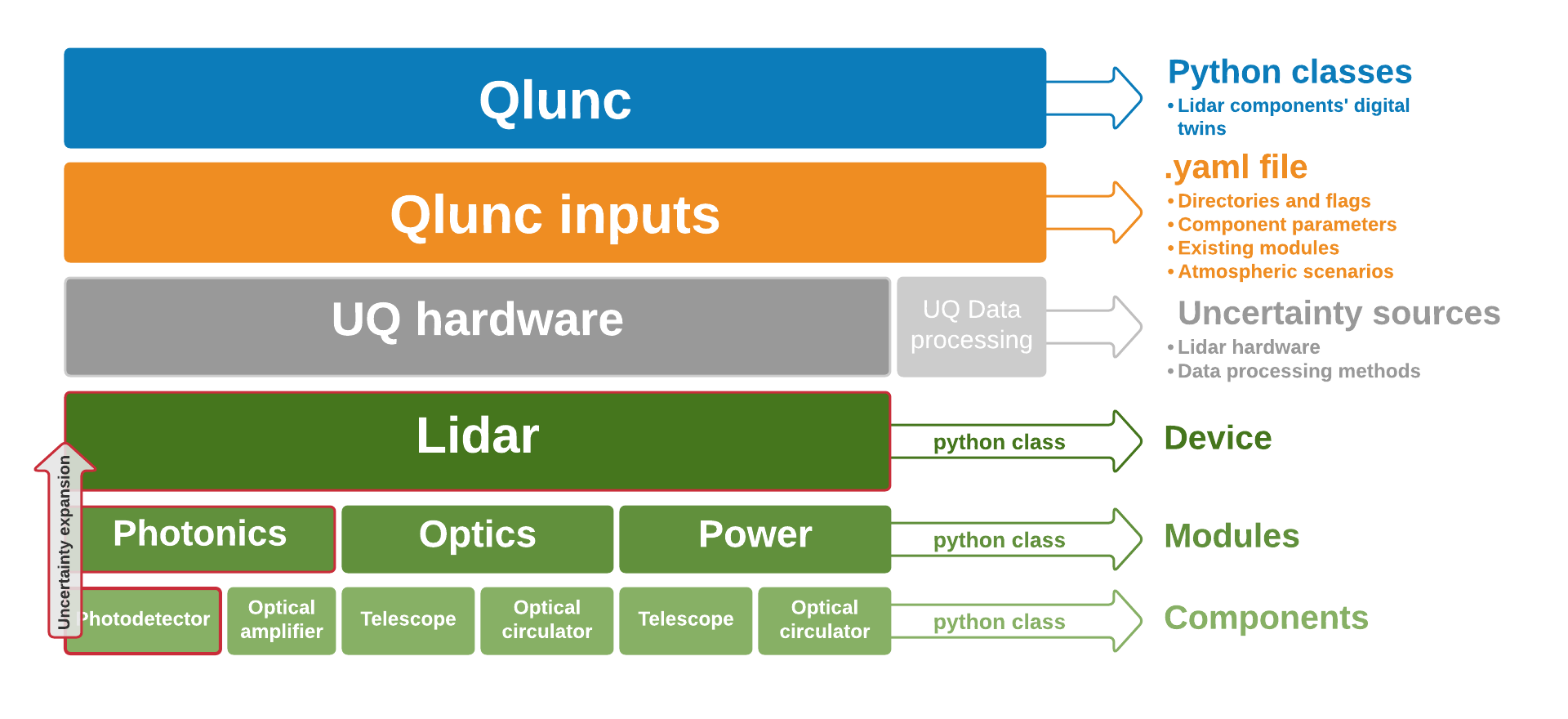


Figure 1. Qlunc basic structure

# How is Qlunc working?

Based on different scripts interconnected and stand-alone functions calculating uncertainties

The user creates the different lidar components by instantiating a python class, including its functional parameters through a yaml file, and defining the function used to obtain the specific component uncertainty. Then, each module (also python objects) is "filled" with the corresponding components and their uncertainties are computed following uncertainty expansion method according to the [GUM](https://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf) model. Once each component is 'ensembled' building up the different modules, the lidar object is created and the modules included. As a result, the desired lidar digital twin is created, the uncertainty of which is computed again by following [GUM](https://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf) suggestions about uncertainty expansion.

* Steps:
  + Creating python classes virtually representing the objects or lidar elements (blue box in [Figure 1](#Fig1))
    - Create python classes for the components (`class MyComponentA`)
    - Create python classes for the modules (`class MyModuleA`)
    - Create python classes for the lidar (`class MyLidarA`)
    - Create python class for the atmospheric scenario (`class AtmosphScen`)
  + Give values to component parameters and instantiate the objects to digitally create lidar objects. Qlunc includes a .yml file to make it user friendly and easy-readable. Thus, is easy to include the component parameter values in the system (orange box in [Figure 1](#Fig1))
    - Filling the yaml file with lidar components’ and modules’ parameter values.
    - Instantiate components, values of which are read from the yaml file:
      * `MyComponentA (paramA1,paramA2, AtmosphScen,Uncertainty1…)`
      * `MyComponentB (paramB1,paramB2,AtmosphScen,Uncertainty1…)`
      * …
    - Instantiate modules. User can define whether a module is included or not in calculations:
      * `MyModuleA (MyComponentA,MyComponentB,AtmosphScen,Uncertainty2)`
      * …
    - Instantiate lidar device:
      * `MyLidarA (MyModuleA,MyModuleB,AtmosphScen,Uncertainty3)`

`Unceratinty1` is a stand-alone python script calculating each component uncertainty. `Uncertainty2` and `Uncertainty3` calculate the first uncertainty expansion regarding the modules and the lidar global uncertainty expansion, respectively. Combined and expanded Uncertainties are calculated accordingly GUM.

# Working example and Tutorials: Do it yourself

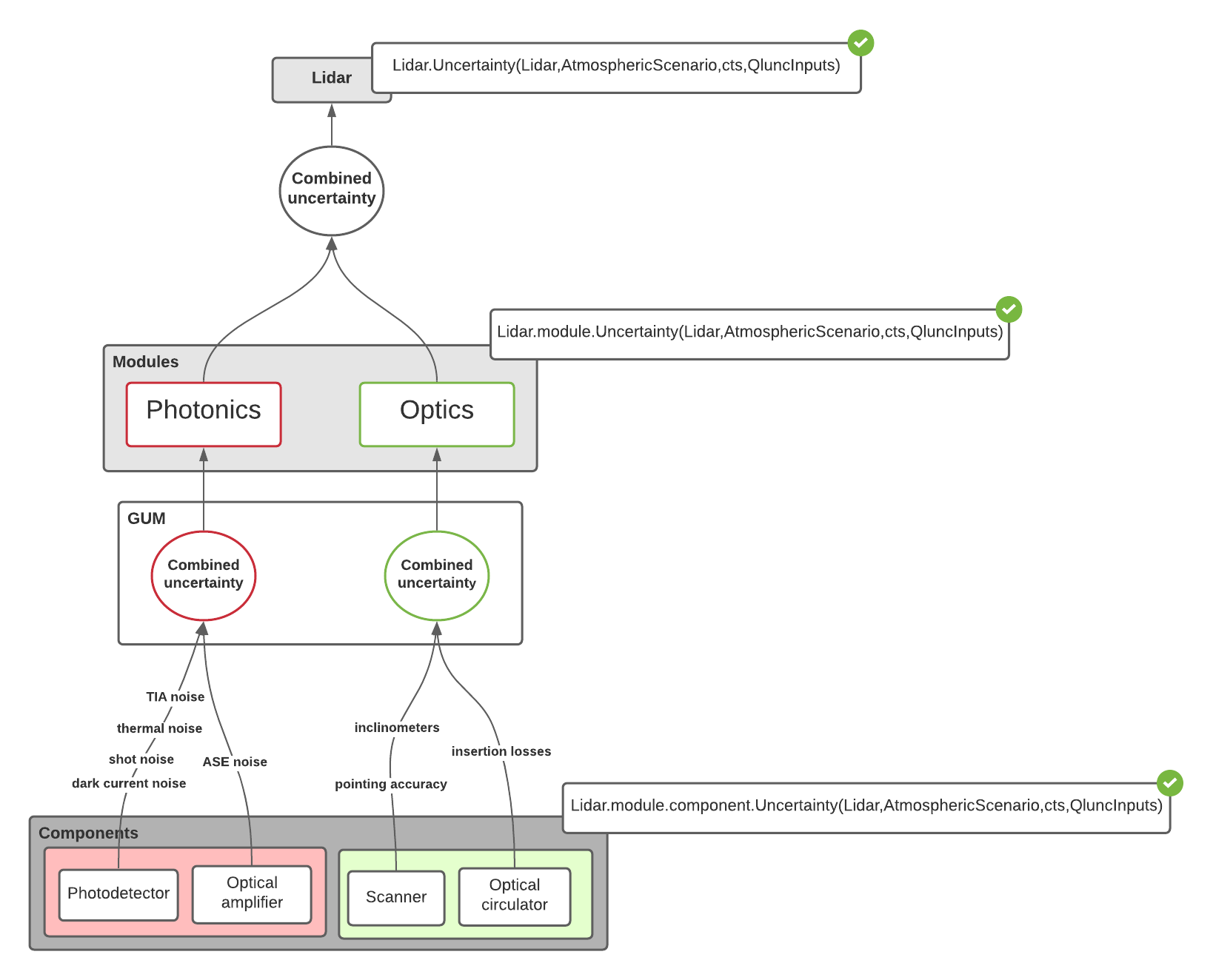
User can find tutorials on how to begin using Qlunc by downloading the Qlunc repository from [https://‌github.com/‌SWE-UniStuttgart/‌Qlunc.git](https://github.com/SWE-UniStuttgart/Qlunc.git). *Tutorial1* aims to facilitate the introduction to Qlunc. In this tutorial, the user will go through the code and create a lidar device with its modules and components. We will ask for either, lidar general or component-specific uncertainties. User will obtain some graphical interesting results, as well. Finally, an example of how to access lidar design parameter values, through *dot notation* method, will be given. Via *Tutorial1* we learn how to create a lidar device. Throughout *Tutorial2* we design two lidar devices and compare against each other. These devices differ from each other just in the scanner head, so we will build up two optic modules by changing the scanner component. Also, some graphical results will be given.

Tutorials include a yaml file for each lidar we want to create. By changing parameter input values, users can design a new lidar and begin getting familiarized with Qlunc.

Apart from the tutorials, the package includes a functional working example. More information about this working example is given in the readme attached, where the process of creating a lidar digital twin is treated in depth.

# Uncertainties estimation

As mentioned above, the code claims flexibility and aims to foster collaboration among researchers. To encourage both, flexibility and further collaborations each lidar module has its own uncertainty estimation function, which includes the components the module is made of. These stand-alone uncertainty estimation functions are easily interchangeable, just in case users want to use another uncertainty model.

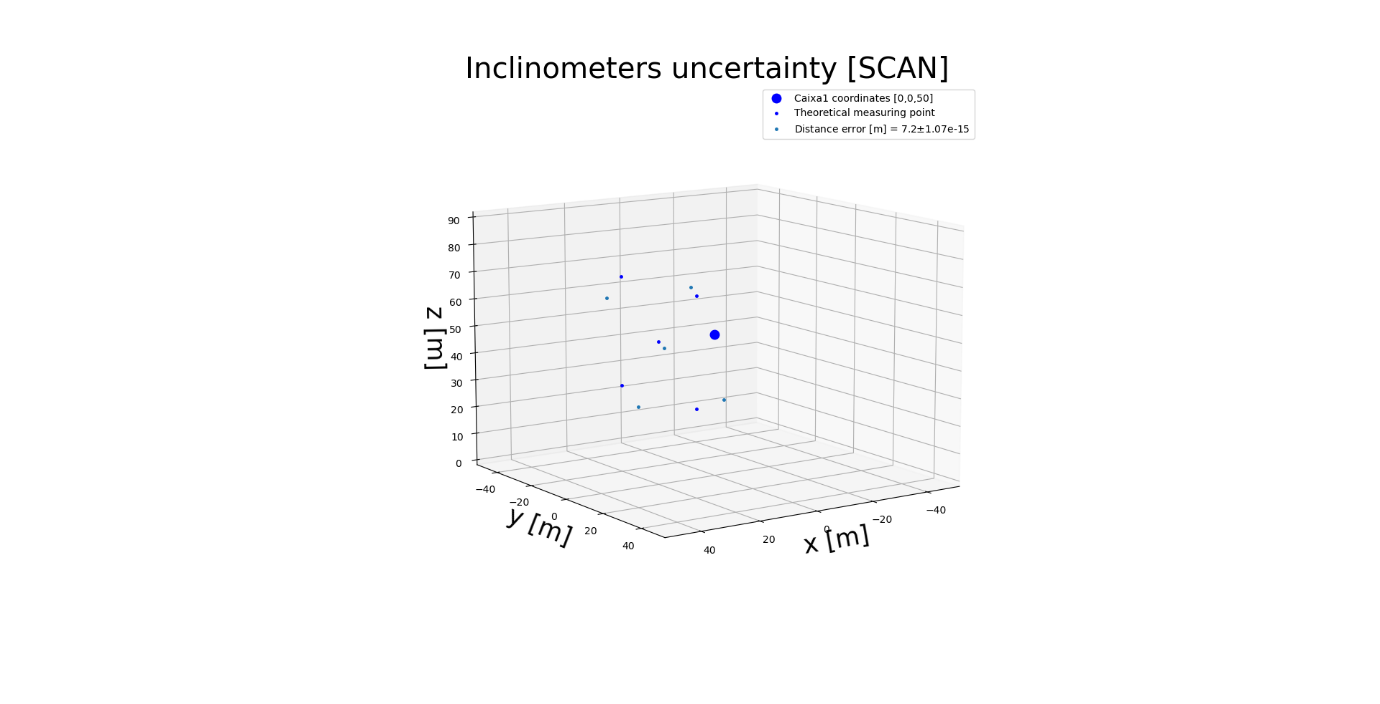
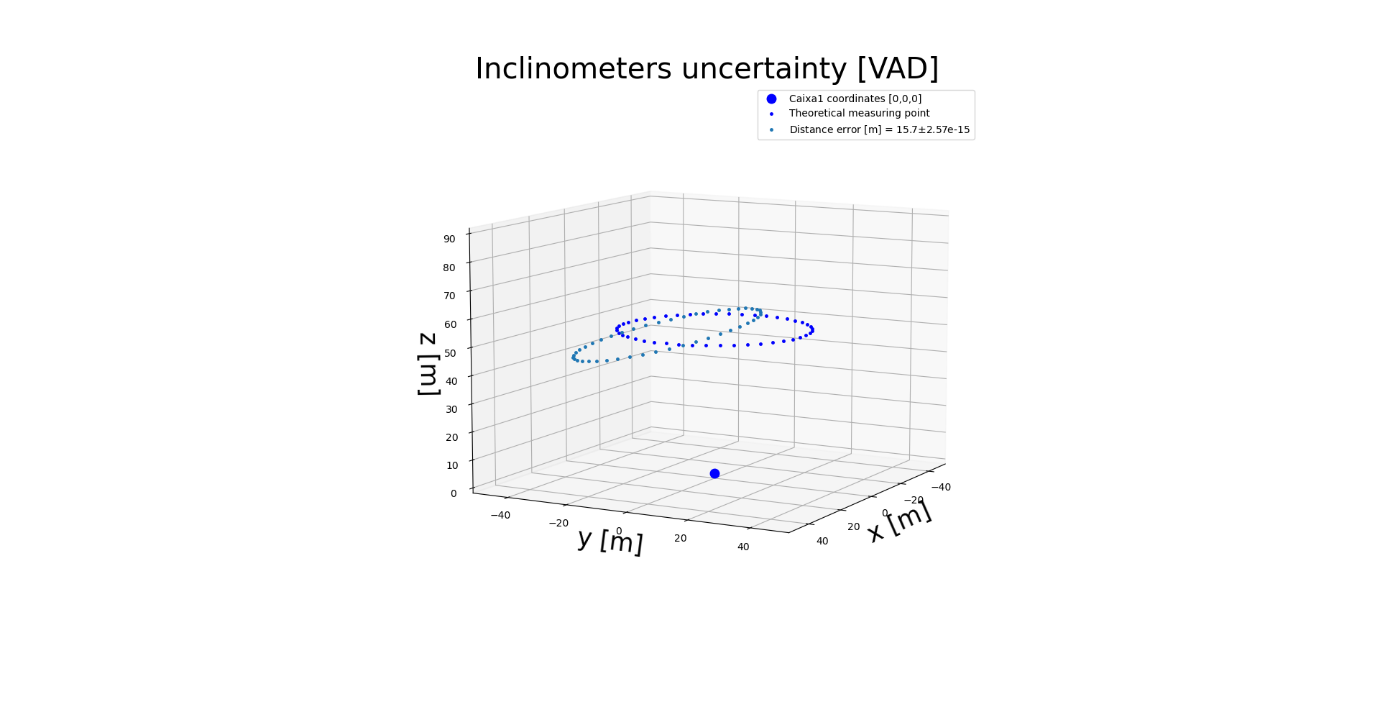
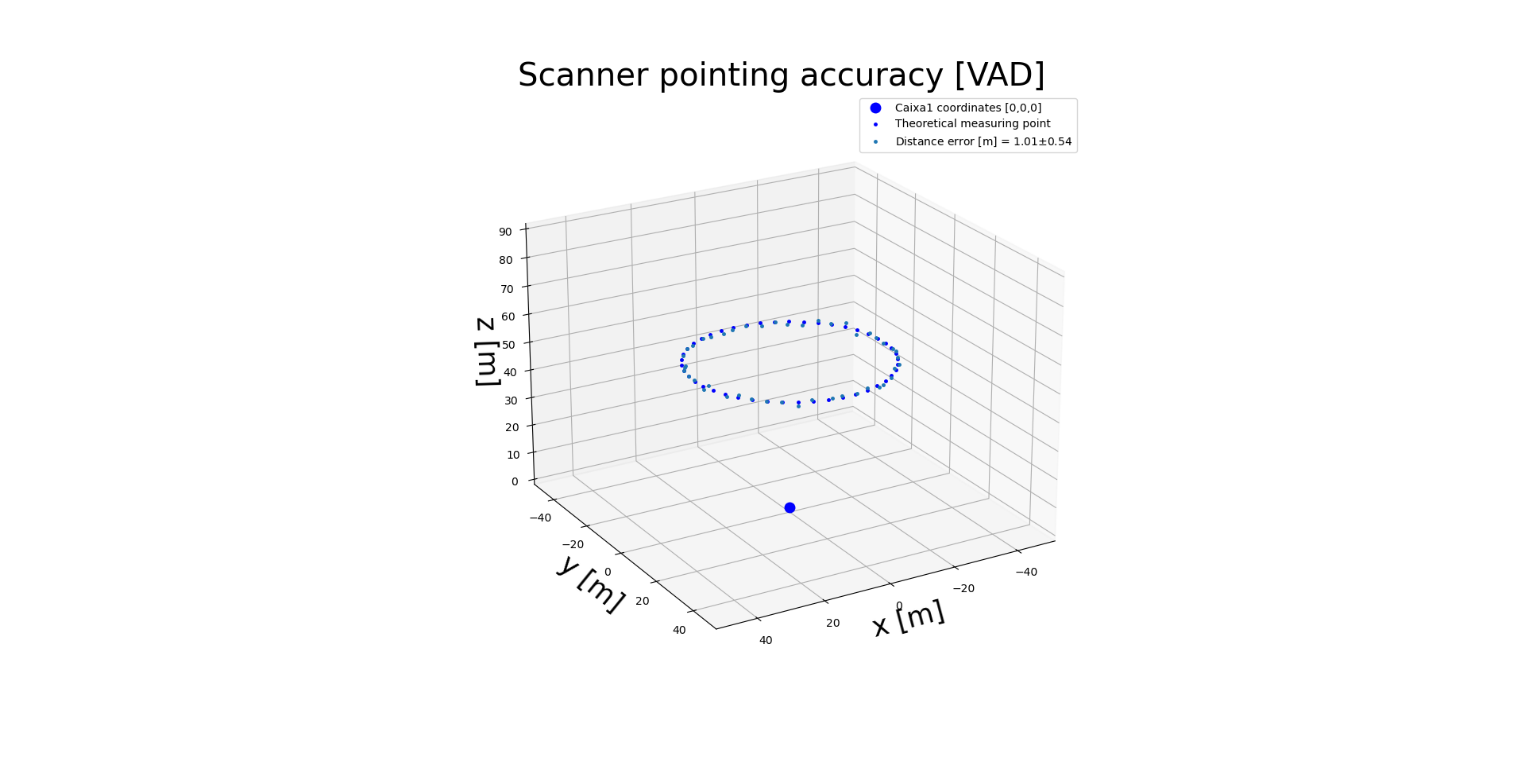
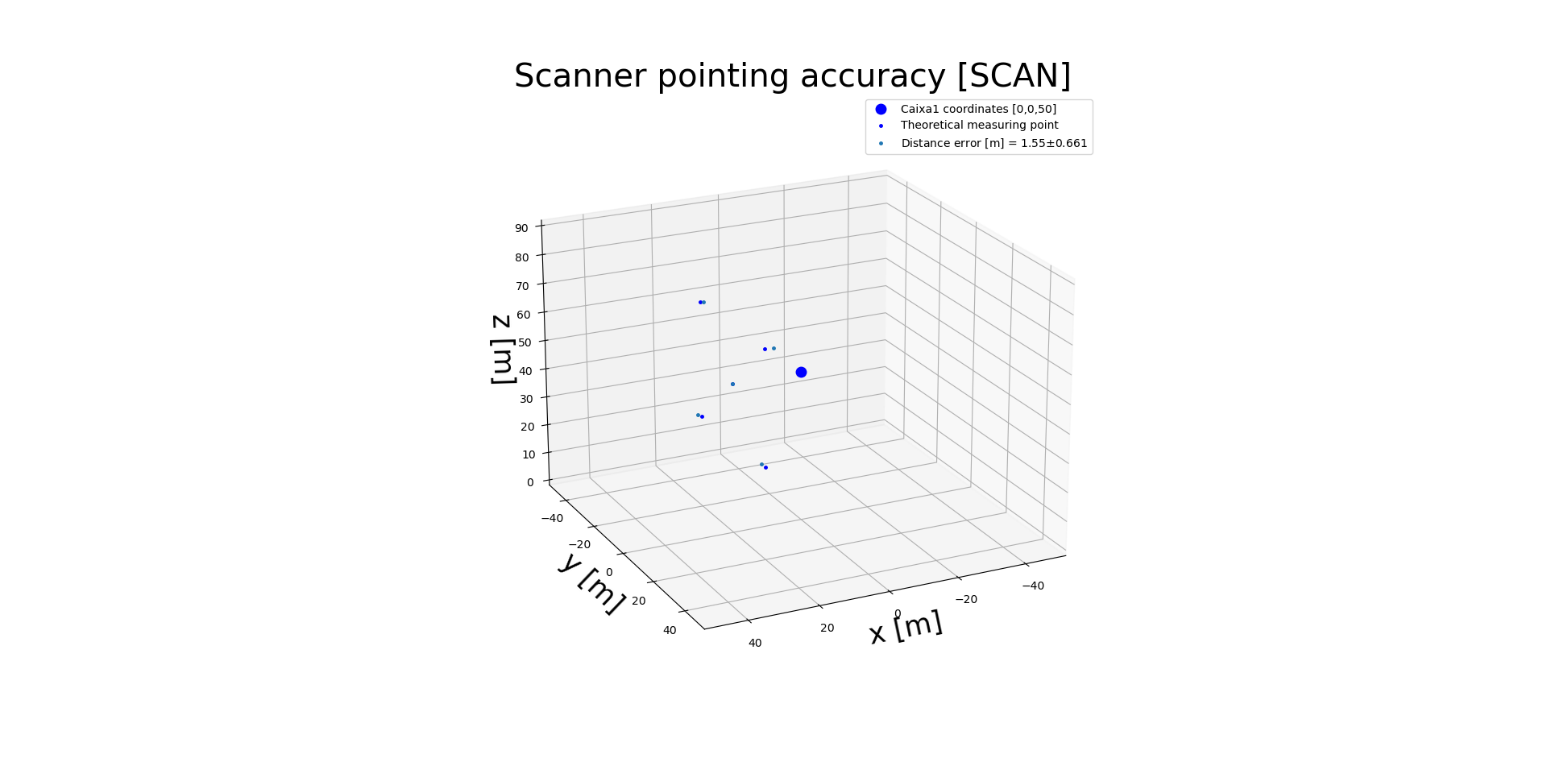


1. Optics module:
   * Scanner: As stated before, Qlunc can interpret positioning data coming from both, scanning and VAD lidar scanning methods. By introducing a 0 mean, white noise with standard deviations - accepting standard deviation as the error of each coordinate when measuring - derived from pointing accuracy parameter errors - in case of VAD, *cone angle*, *azimuth* and *focus distance* and *x*, *y* and *z* in case of scanning lidar, and their corresponding standard deviations in degrees and/or meters -, the code computes a Monte-Carlo simulation. To do so Qlunc computes 100 loops with 10.000 combination cases each loop, calculates the difference between the simulated and the theoretical coordinates’ measuring points, obtains the mean and the standard deviation for each loop and eventually calculates the mean of the 100 loops and yields an estimation of the scanner uncertainty, the error of which decreases as . is a convolution of 100 times synthetic noise addition with 10.000 coordinate combinations, given a total of 1.000.000 combinations to obtain the distance error and the standard deviation of the distance error.
   * Inclinometers: Inclinometer uncertainty contribution is estimated by applying a rotation over simulated measuring patter points obtained in the previous step, using inclinometer deployment angle errors: yaw, pitch and roll.

Both errors are calculated simultaneously, so the user obtains a distance value and a distance standard deviation in meters, . Given a coverage factor of , the user obtains a distance error interval within which 68% of the data will drop in.

* + Optical circulator: Lidar optical circulator uncertainty is assessed by taking into account the optical circulator insertion loss parameter, expressed in dB. In the near future, further investigations on optical circulator structure will enable us to introduce polarizing beam splitters, so we can account for their contributions to losses, namely transmittance and reflectance.

As can be seen, scanner and inclinometers uncertainty are considered as independent of the rest of the optical components. Indeed, when talking about distances, errors are given in meters but dB when talking about signal error contributors, or signal noise. Therefore, for good comprehension and meaningful results of uncertainty estimations, errors should be given in the correct format, accounting for the object under assessment.



a)

b)

Figure 2. a) Scanner pointing accuracy & uncertainty in distance and b) inclinometers uncertainty

1. Photonics module
   * Photodetector: Three different contributions are taken into account to estimate photodetector uncertainty: thermal noise, shot noise and dark current noise; transimpedance amplifier noise contribution, if necessary, is included in calculations.
     + Thermal noise is generated by the load resistor and follows Gaussian statistics. For a receiver with a spectral bandwidth , the mean-square noise current representing the total thermal noise power can be expressed as [4], where is the Boltzman constant, is the temperature parameter, and is the photodiode load resistor.
     + Shot noise, also known as quantum noise, is a white noise too. Has its origin in the statistic nature of the photodetection, due to the random interaction of photons with the detector, giving place to a fluctuating photocurrent because of this random nature of the photo arrival. Shot noise can be characterized by the variance of the shot noise current as [4], where q is the electron charge, ℜ is the responsivity representing the conversion efficiency and accounts for the signal power.
     + Dark current noise is the constant current in the photodetector output when no light is incident on the photodiode. The dark current noise for a photodetector with a bandwidth can be expressed as [4], where is the dark current of the photodiode.
     + Transimpedance amplifier: In an optical receiver, an electrical preamplifier is needed immediately after the photodiode to amplify the photocurrent and convert it into an electrical voltage. Based on manufactures’ photodetector design values, noise generated by the transimpedance amplifier can be estimated as
   * Optical amplifier: In erbium-doped fibre amplifiers or EDFA, the basic mechanism leading to noise in the optical amplifier is the amplified spontaneous emission or ASE [5][6]. Here ASE is computed using noise figure (NF) data from manufactures. If wavelength does not match lidar output wavelength, Qlunc interpolates among manufacture values to obtain the most accurate and closest to the lidar wavelength NF value.

where is the noise figure, is the Plank constant, is the laser wavelength, is the speed of light and is the optical amplifier gain.

Then, to estimate the total signal output noise concerning the photonics module we can use GUM suggestion about uncertainties combination:

1. Lidar device:

* Modules: Optics and photonics

To calculate the signal noise incorporated to the input signal from both, photonics and optics modules included so far we use GUM expansion uncertainty model:

To characterize the whole signal noise and account for the signal quality, SNR parameter, defined as the ratio between input power signal and noise power at the output, is useful. [Figure 3](#Fig3) depicts the contributions of each photonic module elements’ noise against the laser input optical power in an interval from -30 to 30 dBm. This figure shows intervals where specific noise can be disregarded and which of them take importance, depending on the source power input interval.

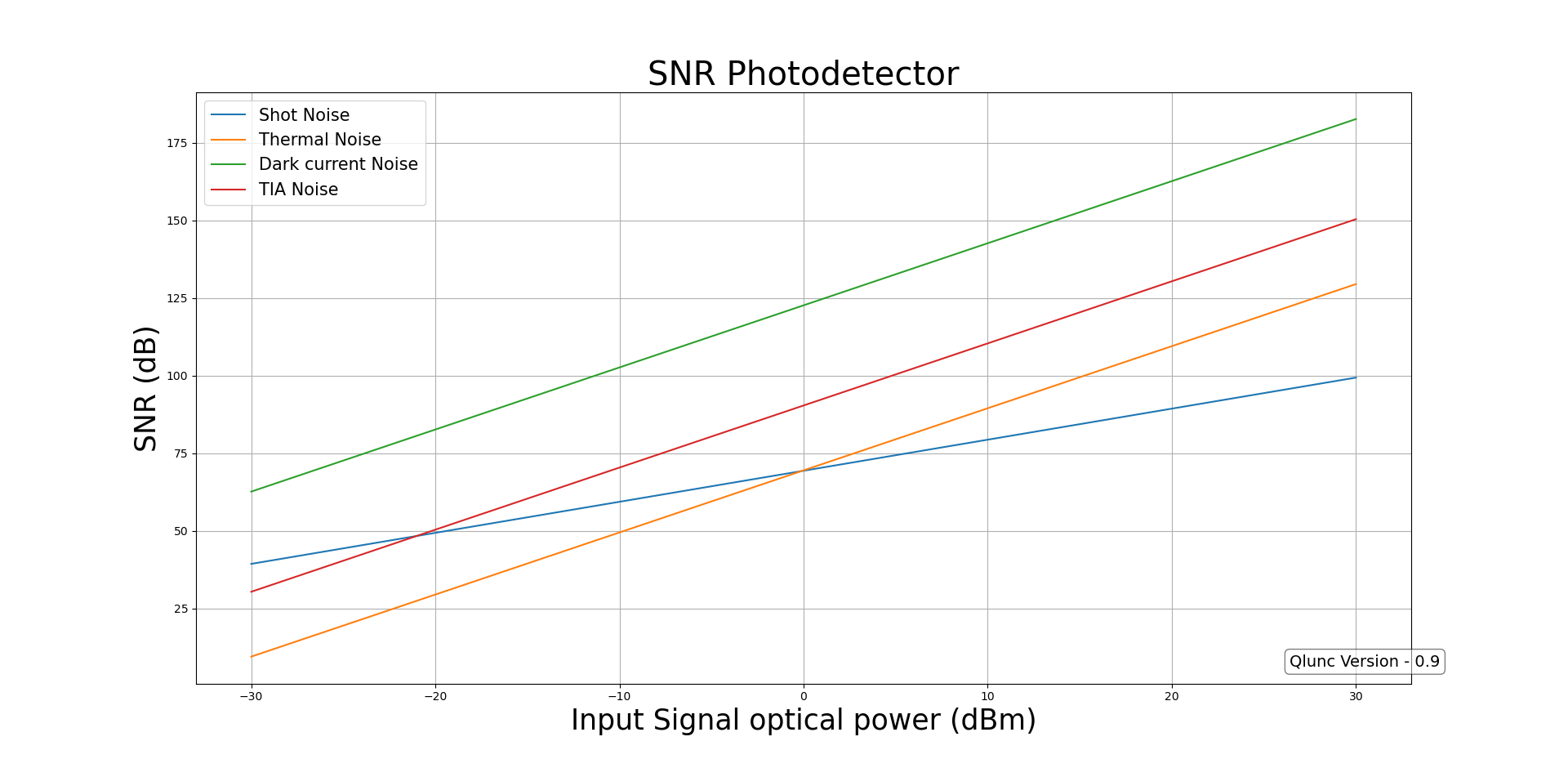


Figure 3. Photodetector noise contributors

# Results

* Dot notation. How to get lidar parameters?
* Asking for uncertainties and code modularity.
* Plots
* tutorials

# Conclusions

Despite Qlunc is yet neither tested nor validated with actual data it promises to be a useful tool to assess lidar uncertainties. The possibility of analyzing noise contributors before a lidar is built can give relevant information about which noise terms we should account for, depending on the input power interval we are working with. This can save effort, design time, and allow us to, maybe, create specific components for specific sites, e.g. on/off-shore, or measuring tasks. For now, is not a full featured application, since many components, modules and new uncertainty sources must still be applied, but offers a main structure where user can see main components, features and capabilities of a lidar measuring device.

Lidar hardware uncertainty is not relevant for lidar uncertainty assessment in comparison with uncertainties introduced by lidar data processing methods.

# Aknowledgement

# References

[1] Joint Committee For Guides In Metrology, “Evaluation of measurement data — Guide to the expression of uncertainty in measurement,” *Int. Organ. Stand. Geneva ISBN*, vol. 50, no. September, p. 134, 2008, doi: 10.1373/clinchem.2003.030528.

[2] V. Pettas, F. Costa, M. Kretschmer, A. Clifton, and P. W. Cheng, “A numerical framework for constraining synthetic wind fields with lidar measurements for improved load simulations.,” *AIAA Scitech 2020 Forum.*, pp. 1–6, 2019, doi: https://doi.org/10.2514/6.2020-0993.

[3] T. Openlidar, “The OpenLidar initiative for wind lidar hardware and software collaboration Wind lidars are complex devices that take huge expertise to develop and,” pp. 9–10, 2019, doi: 10.5281/zenodo.3414197.

[4] H. Rongqing, “Introduction to Fiber-Optic Communications,” *Elsevier*, vol. 4, pp. 125–154, 2019, doi: https://doi.org/10.1016/B978-0-12-805345-4.00004-4.

[5] B. P. Pal, *Guided wave optical components and devices: basics, technology, and applications*. 2005.

[6] G. L. Cariolaro, R. Corvaja, P. Franco, M. Midrio, and G. L. Pierobon, “Noise in optical amplifiers,” *Fiber Integr. Opt.*, vol. 13, no. 2, pp. 199–213, 1994, doi: 10.1080/01468039408202230.

1. Stuttgart Wind Energy, Institute of Aircraft Design and Manufacture, University of Stuttgart, Stuttgart, Germany - costa@ifb.uni-stuttgart.de - <https://orcid.org/0000-0003-1318-9677> [↑](#footnote-ref-1)