Organisation of data in standardized HDF5 Brillouin spectra containers

Pierre Bouvet

November 18, 2024

1 Introduction

The evolution of Brillouin Light Scattering (BLS) applications has lead to an increasing number of publications and research on this domain, resulting in a vast collection of data obtained with different techniques and setups by different laboratories. The evolution of this domain has reached the point where to allow it to grow outside of dedicated laboratories, a common approach to storing, treating and interpreting the data is needed. In an effort to facilitate this endeavour, we here propose a new normalized file structure based on the HDF5 file format, and introduce a dedicated Python Library: HDF5_BLS to facilitate the use of the format and encourage its use as well as normalized treatment approaches.

In this document, we will propose a normalized structure of the HDF5 files and present the concept behind the treatment of BLS spectra by the proposed Python library.

2 Storing raw data in HDF5 wrapers

2.1 Concept

The first need of the proposed format is to easily store data inside a HDF5 file. To do so, we propose to place the raw data in a dedicated object inside the HDF5 file, in the form of dataset:

```
HDF5 file
└─ Data (type:  object)
   └─ Raw Data (type:  Dataset)
```

From there the goal of the HDF5 file format is to store together with the data, the parameters used for the experiment. These parameters will be stored in the form of text attributes to the HDF5 file:

```
HDF5 file
├─ Attributes (type:  HDF5 attributes stored as text)
│
└─ Data (type:  object)
   └─ Raw Data (type:  Dataset)
```

Because entering all the attributes of a spectrometer for each measure is time consuming, we propose to rely on a standardized spreadsheet that is imported to create the attributes. Aside from standardizing the nomenclature of the attributes, this will also allow researchers to create

spreadsheets explicitely for their instrument, and reuse these documents each time the instrument is used for a new measure, with minimal changes.

2.2 Use of the file format to wrap data

Using the proposed file structure allows us to add to the file, different data for different uses. For instance, a calibration curve of the spectrometer made with a sample of reference, a measure of its impulse response or any other data relevant to the experiment can be added to the file easily following this structure:

```
HDF5 file
├─ Attributes (type:  HDF5 attributes stored as text)
│
├─ Data (type:  object)
│  ├─ Raw spectrum (type:  dataset)
│  │
│  ├─ Calibration spectrum (type:  dataset)
│  │
│  ├─ Impulse response (type:  dataset)
│  │
│  └─ Other data (type:  dataset)
```

Creating different groups inside the same file can also be used to store data taken in different conditions. In that case, we propose to number the groups as follows:

```
HDF5 file
├─ Attributes (type:  HDF5 attributes stored as text)
│
├─ Data (type:  object)
│  ├─ Data_0 (type:  object)
│  │  ├─ Data_0_0 (type:  object)
│  │  │  ├─ Data_0_0_0 (type:  object)
│  │  │  │
│  │  │  ├─ Data_0_0_1 (type:  object)
│  │  │  └─ ...
│  │  │
│  │  ├─ Data_0_1 (type:  object)
│  │  └─ ...
│  │
│  ├─ Data_1 (type:  object)
│  └─ ...
```

3 Data types

To allow the storage of the largest type of data possible, we propose to store raw data as n-dimensional arrays of floating point numbers. This choice forces however to set a certain order on the dimensions, that we propose as follows:

- Nth dimension: spectral channels
- (N-1)th dimension: (if reported) z dependence
- (N-2)th dimension: (if reported) y dependence
- (N-3)th dimension: (if reported) x dependence
- (N-4)th dimension: (if reported) time dependence
- (N-5)th dimension: (if reported) radial angular dependence
- (N-6)th dimension: (if reported) azimutal angular dependence
- (N-7)th dimension: (if reported) temperature dependence
- (N-8)th dimension: (if reported) concentration dependence
- (N-9)th dimension and further: user-specific dimensions

We propose that in case of redundancy (for example a dependance on both time and position), the user chooses the relevant dimension.

4 List of attributes

One key condition for this format to be useful for meta-studies, is to allow users to access the same parameters of the measures in different files. To do so, we propose a normalization of the name of the attributes. This list will evolve and can be found on the dedicated GitHub page of the project in "Spreadsheets/Attributes.xlsx".

These attributes are divided in three sections:

- Measure: all the parameters that are measure-dependent and that are not strictly linked to the tool used to measure the BLS spectra
- Spectrometer: all the parameters that are instrument-dependent. This includes all the choices made at the moment of developping the spectrometer. This separation is particularly interesting for manufacturers of standard tools as they can directly refer the model of their device here and adapt their treatments to this single parameter.
- File properties: all the file-related parameters that are linked to the numerical storage of the data and are not dependent neither on the measure nor the instrument.

This list is prone to evolve and we encourage everyone interested in this project to share with us the parameters they feel are needed for their particular application in order to add it to the standard.

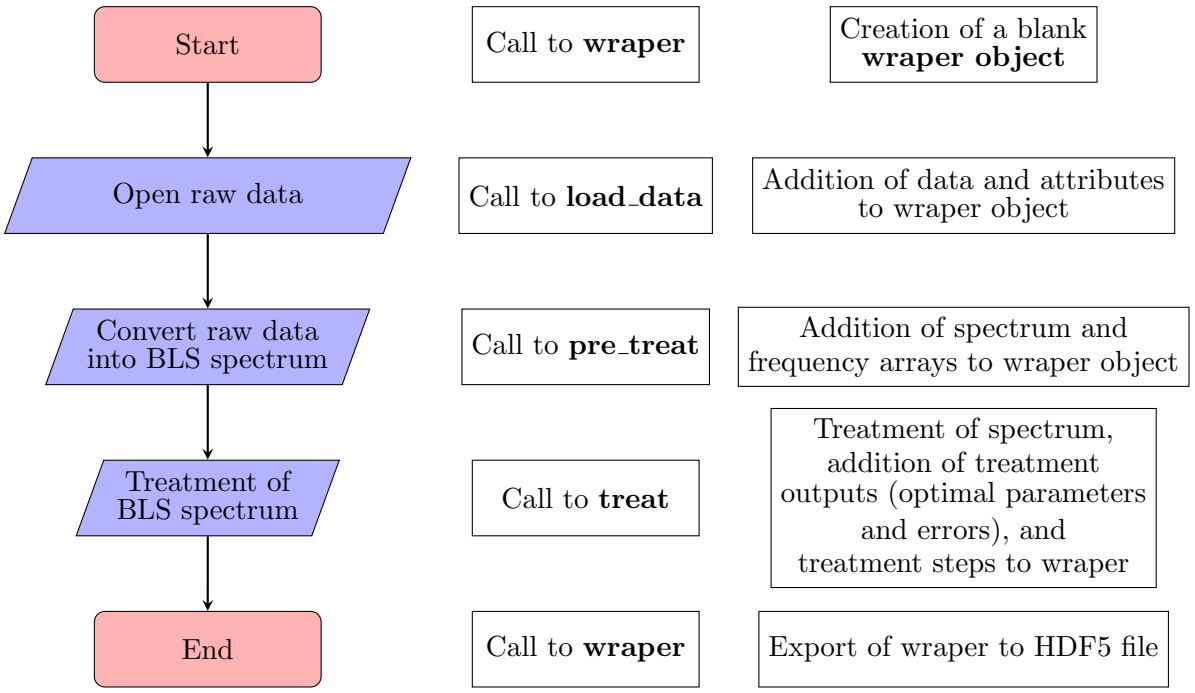
5 HDF5_Brillouin_creator modules structure

The module presented here aims at allowing the user to import, convert, organize and treat the data. As such the module is divided in four sub-modules:

- load_data: the sub-module dedicated to extract data and attributes from a file.

- wraper: the sub-module dedicated to creating the object where both the data and attributes will be stored, where the treatment steps will be recorded and that can export to a HDF5 file.
- pre_treat: the sub-module dedicated to obtain a BLS spectrum from the raw data, a step dependent on the type of technique and the spectrometer being used to acquire the spectrum.
- treat: the sub-module dedicated to treating a Brillouin spectrum.

The library is meant for unifying the storage and treatment of spectra which can be schematized by the following pipeline:



6 Independent use of the module

The module can be used independently of the wraper module, by simply calling the desired functions of the three sub-modules "load_data", "pre_treat" and "treat".