JNanobeacons. A Jpp package for KM3NeT inter-DOM time calibration and more.

Rodrigo G. Ruiz

July 1, 2019

Abstract

This note describes the analysis of the KM3NeT nanobeacon runs to perform the inter-DOM time calibration, and the tools developed with that purpose.

1 Introduction

The time calibration of a KM3NeT Detection Unit (DU) consists on finding the t_0 for each Photo Multiplier Tube (PMT) in the DU. The t_0 of the i^{th} PMT in the DU is measured in nanoseconds. It is the time correction that should be added to the raw times of the hits recorded by the PMT in order to get physical times that are consistent with the times of the rest of the hits in the DU:

$$t^i_{phys} = t^i_{raw} + t^i_0 \tag{1}$$

The time calibration of a DU is divided into two different stages: intra-DOM time calibration, and inter-DOM time calibration. The former is performed independently for each DOM in the DU. It consists on finding the t_0 for all the PMTs in a DOM, ensuring that the physical times of the hits recorded by the 31 PMTs of the DOM will be consistent with each other. The intra-DOM calibration is performed using the optical background and it is described in reference REF. Both of them are carried out in the dark room before the DU is deployed, and periodically after the DU is deployed in the sea water. Figure 1 shows the different stages in the calibraiton of a DU.

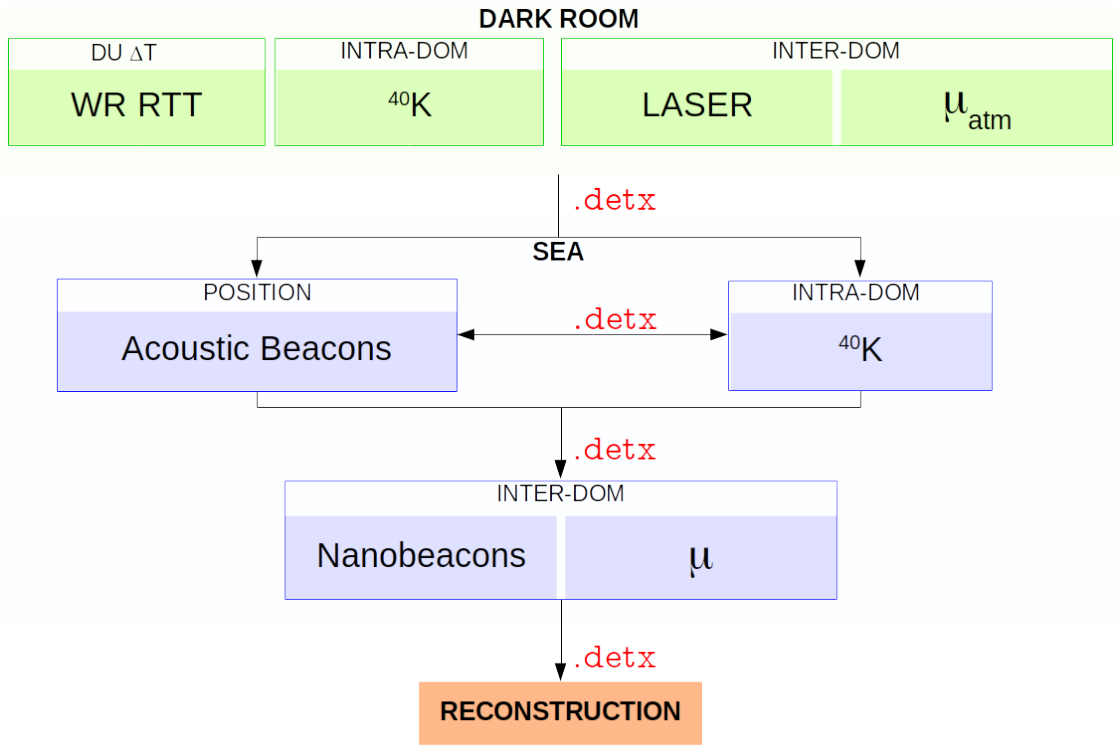


Figure 1: Different stages in the calibraiton of a DU.

After the intra-DOM time calibration has been performed, the physical times of the hits recorded by each DOM are self consistent, but the physical times of the hits recorded by two different DOMS in the DU are not consistent with each other. To make the physical times of the hits recorded by DOMS i and j consistent with each other, it is necessary to find an inter-DOM correction factor $\Delta t_0(i,j)$. The aim of the inter-DOM time calibration is to find this factor. When the DUs are in the sea water this process can be carried out by the use of the nanobeacons REF installed in each DOM. It consits on finding the correction factor $\Delta t_0(i,j)$ that makes the physical times of the hits in DOMS i and j to be consistent with the time of flight of the light from DOM i to DOM j. This note explains how this is done in the sea water, and the tools developed for this purpose. Sections 2 and 3 review the concepts of the inter-DOM time calibration with nanobeacons, and section 4 explains the tools developed for performing the corresponding analyses.

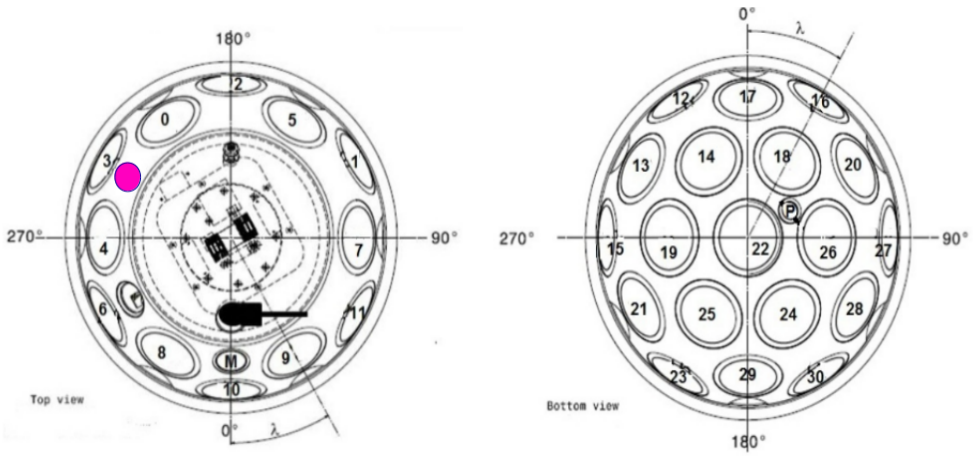


Figure 2: Plant view of the upper and lower hemispheres of a DOM. The magenta dot between PMTs 0, 3 and 4 in the upper hemisphere indicates the position of the nanobeacon

2 Inter-DOM time calibration with nanobeacons

Figure 2 shows the position where the nanobeacons are installed in each DOM. When a nanobeacon is triggered it emits a light pulse (with a width of the order of ¡10 ns) that can be detected by different PMTs in the DU. The inter-DOM calibration with nanobeacons is based on the measurement of the arrival time of a nanobeacon pulse to PMTs located on different DOMs. Assuming that the distance between the PMTs is known, one can compute the expected time difference between the arrival of the puse to both PMTs (also called time of flight). A a result of the comparison of the expected time difference with the observed time difference, a correction factor is obtained for the couple of DOMs that host these PMTs. Formally, the procedure can be described as follows:

Let i and j be two PMTs in two different DOMs from the DU, and assume that a nanobeacon pulse is detected by both of them. One can use equation 1 to compute the physical time of the hits produced by the pulse in both PMTs,

$$\begin{aligned} t^i_{phys} &= t^i_{raw} + t^i_0 \\ t^j_{phys} &= t^j_{raw} + t^j_0 \end{aligned} \tag{2}$$

Substracting both expressions one can obtain the physical time difference between the hits produced by the pulse in both PMTs as a function of the raw time differences and of the difference

between the t_0 of both PMTs,

$$\Delta t_{phys} = \Delta t_{raw}(i,j) + \Delta t_0(i,j), \tag{3}$$

where Δ represents a difference (ie $\Delta t(i,j) = t_i - t_j$). Therefore, $\Delta t_{phys}(i,j)$ should be the expected time of flight mentioned above. Assuming that the positions of both PMTs $\boldsymbol{r_i}$ and $\boldsymbol{r_j}$ are well determined, it can be computed as the time that light needs to propagate from $\boldsymbol{r_i}$ to $\boldsymbol{r_j}$ in the sea water,

$$\Delta t_{phys} = d(\boldsymbol{r_i}, \boldsymbol{r_j})/c_w. \tag{4}$$

c_w is the speed of light in water. Therefore, equation 3 allows to compute the quantity $\Delta t_0(i,j)$ from the PMT positions and from their respective measurements of the nanobeacon pulse.

If the intra-DOM time calibration has been performed and the t_0s of the PMTs within each DOM are consistent with each other, then the quantity $\Delta t_0(i,j)$ can be used to appropriately shift the t_0s of all the PMTs in one of the DOMs. If this process is repeated iteratively with all the DOM pairs in the DU, then the DU will be completely calibrated. The following example can help to visualize this process.

2.1 A practical example

- Assume that both the acoustic positioning and the intra-DOM time calibration have been performed in a DU, and that a detector file with the positions and t_0s of all the PMTs is available.

- Assume that the nanobeacon of the DOM in floor 1 is flashed, and its pulse is detected by the PMT 0 of DOM in floor 1, and by the PMT 22 of the DOM in floor 2.

- The observation of the nanobeacon pulse by PMT0-F1 and PMT22-F2, the positions from the detector file and equation 3 are used to compute the quantity

  $$\Delta(t_0)_{NB} \equiv \left[\Delta t_0(\mathrm{PMT}-22\mathrm{F2}, \mathrm{PMT0}-\mathrm{F1})\right]_{NB}.$$

  This means that in order for the physical times of the hits recorded by these two PMTs to be consistent, the t_0 of PMT22-F2 must be larger than the t_0 of PMT0-F1 by an amount $\Delta(t_0)_{NB}$.

- One can use the detector file to calculate the value of this same quantity from the t_0s before the nanobeacon calibration, $\Delta(t_0)_{DF}$. If $\Delta(t_0)_{DF} == \Delta(t_0)_{NB}$, this means that the positions and the values of the t_0s for both PMTs in the detector file would allow to calculate correctly the time of flight of the nanobeacon pulse from one PMT to the other. Since the intra-DOM calibration has already been performed, then nothing needs to be done because the PMTs in each of the DOMS are correctly synchronized with each other.

- If $\Delta(t_0)_{DF} != \Delta(t_0)_{NB}$, then the t_0s in the detector file should be corrected. This correction is done by leaving the t_0s for the PMTs in DOM 1 as they are in the detector file, and by shifting the t_0s of DOM2 by $\Delta(t_0)_{NB} - \Delta(t_0)_{DF}$. Keep in mind that in order to keep the results of the intra-DOM time calibration, the t_0s of ALL the PMTs in DOM 2 must be corrected by the same factor. At this point, the t_0s of all the PMTs in DOMs 1 and 2 are consistent.

- The process is repeated by flashing the nanobeacon in floor 3 and detecting its signal with a PMT in DOM 4, and so on.

Figure 3 shows the comparison of three different detector files for the ORCA detector. The three of them are identical except for the t_0s of the PMTs in DU2. One contains the t_0s estimated from calibrations in the dark room, other contains the values after the inter-DOM calibration with nanobeacons as described above, and the third one contains the t_0s found by an alternative inter-DOM time calibration performed in the sea from the reconstruction of atmospheric muons. The X axis represents couples of consecutive DOMS. The Y axis in the upper pannel represents $\Delta(t_0)_{NB}$, $\Delta(t_0)_{DR}$ and $\Delta(t_0)_{\mu}$. The lower pannel represents the diference between the points in the upper pannel, and the straight lines show the mean difference for all the pairs in the DU. Following the description of the calibration method given above, this plot shows that both inter-DOM time calibration methods found differences with respect to the dark room calibration that needed to be corrected. The comparison of the detector files obtained after performing both the nanobeacons and the atmospheric muon calibrations, show that their results are consistent with each other.

3 Optimization of the nanobeacon voltages

For a better accuracy in the inter-DOM calibration with nanobeacons, the nanobeacons of each DOM in a DU should be appropriately tuned. The ideal voltage for a nanobeacon is that voltage that

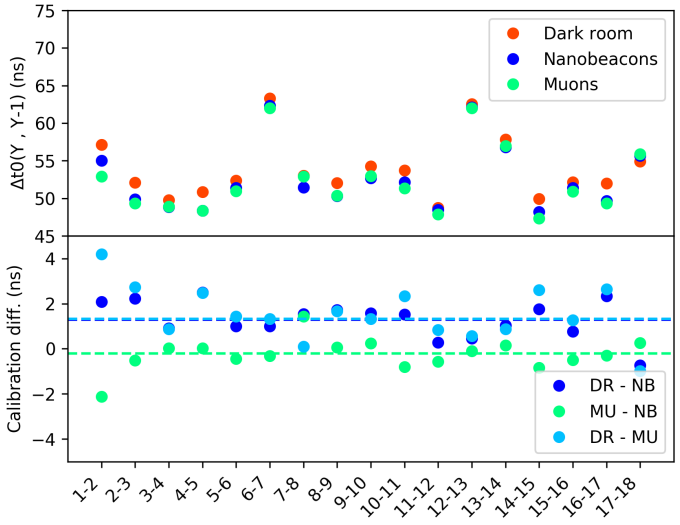


Figure 3: Comparison of detector files after different calibrations for ORCA DU2.

produces the maximum amount of hits in the PMTs around it with an average ToT value that doesn't exceed the 1 photo electron level (which, as suggested by some laboratory studies REF corresponds to 26.4 ns).

The optimization of the nanobeacon voltages in a DU is performed by taking multiple runs, each of them characterized by a voltage at which all the nanobeacons in the DU are tuned. Currently 9 runs are taken where the corresponding voltage values are 5.5V , 6V , 6.5V , 7V , 7.5V , 8V , 8.5V , 10V and 15V. To optimize the voltage of a nanobeacon, one can look at the ToT distribution of the hits that it produces in the closest PMT from the same DOM for each of the different runs (ie, for each of the voltage values). Figure 4 shows these results for the nanobeacons in floors 3 and 10 of the ORCA DU2. Each of the curves shows the result of performing this operation different weeks. These results tell that the behaviour of the nanobeacons is very stable with time and that in principle, it suffices with performing the voltage optimization only once.

4 The Code

A set of tools has been developed and included in Jpp that allows to perform the inter-DOM time calibration and to find the optimal nanobeacon voltage, as well as other studies related with the nanobeacon pulses, such as studies related to the measurement of water properties.

Figure 5 show the flow diagrams for the voltage optimization and for the calibration analyses, and the different programs involved. Automating these analyses would be rather simple for an ideal

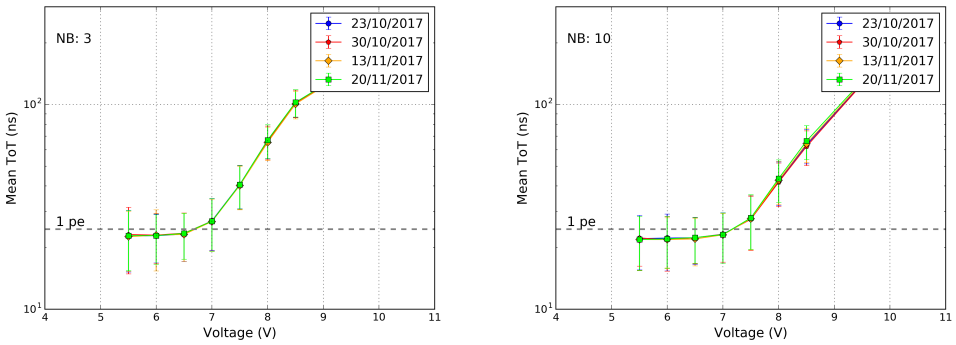


Figure 4: Results of performing the nanobeacon optimization procedure four different times for two nanobeacons in ORCA DU2. These curves show the mean ToT value of the hits produced by the nanobeacon in the closest by PMT as a function of the nanobeacon voltage. The horizontal line shows the 1 pe level. In this case, the nanobeacon voltages are tuned to 6.5V and 7V respectively.

detector where every PMT and nanobeacon worked well and stably, and where the nanobeacon signal was observed as a nicely defined pulse by the different PMTs. But it may happen that certain pmts or nanobeacons cease to work, that certain DOMs are switched off for whatever reason, or that the nanobeacon signal is not very well detected by the PMTs. The programs depicted in figures 5 use some common classes designed to make the code as compact as possible while accounting for different eventualities and problems. The main difficulty is to decide at each time what PMTs and nanobeacons to use for an analysis.

4.1 The common classes

The common classes mentioned above are four: the `NBRun` class, the `SUPERMODULE` class the `SUPERPMT` class, and the `NBPulse` class. In the following, a brief description of each of them is given.

4.1.1 The `NBPulse` class

This class is devoted to the analysis and classification of the nanobeacon signal as observed by a given PMT. All of its methods and attributes are realted to a 2D histogram with the ToT and time distribution for the hits recorded by the PMT during a time window wide enough to include the hits

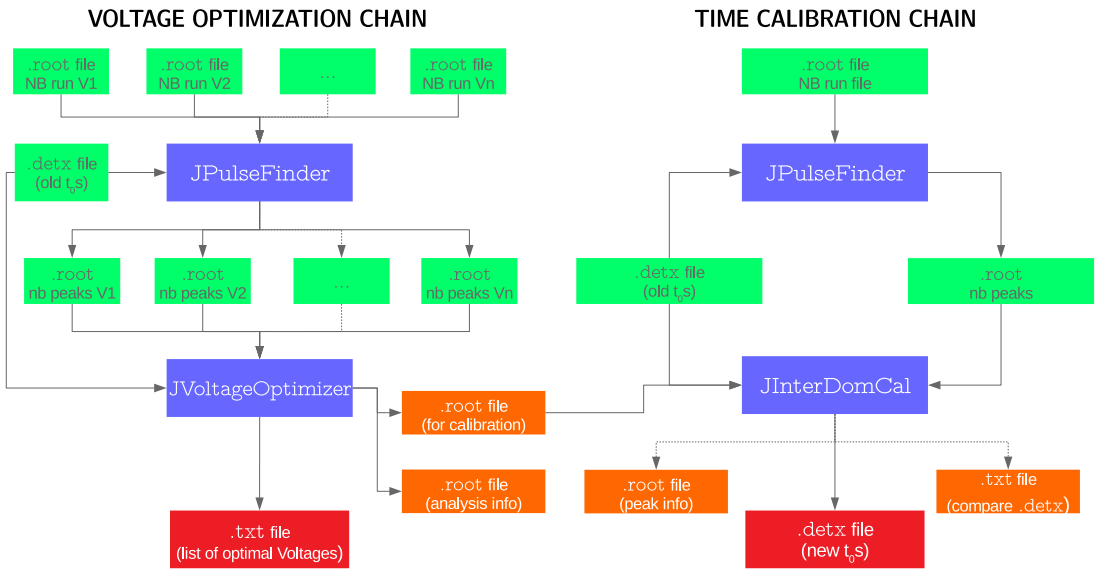
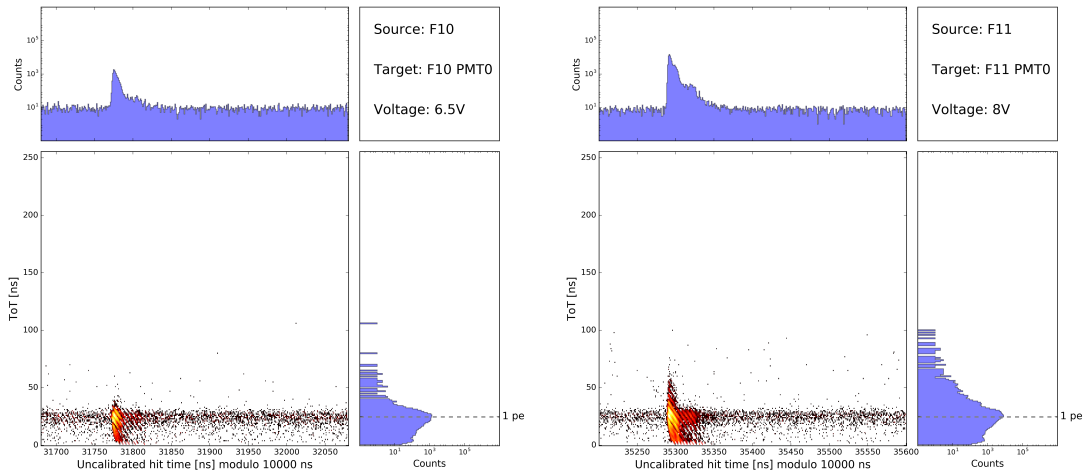


Figure 5: Flow diagrams containing the different programs that perform the inter-DOM time calibration and the nanobeacon voltage optimization. The red boxes indicate the output that these chains are designed to produce. The orange boxes represent extra optional output. The green boxes represent input files, and the blue boxes represent the programs.

produced by the nanobeacon. Some examples of such histograms and their projections onto the X and Y axes are shown in figure 6. These plots also illustrate the three different categories into which the pulses are classified. Their classification is done according to their time distribution (the projection into the X axis)[1]. At the time of performing an analysis, pulses classified as good will have the highest priority, and weak or saturated pulses will be discarded.

The `NBPulse` class has several attributes related to the pulse properties and several analysis methods. The two most important attributes of the peak at the time of performing the time calibration and the nanobeacon voltage optimization are the arrival time of the pulse and its mean ToT. Currently there are two analysis methods implemented in the class. The first analysis method is a fast analysis which consists on projecting the 2D distribution on the X axis, and selecting a window of about 10 bins around the bin with maximum content of the resulting distribution. The pulse arrival time in this case is defined as the bin center of the bin with maximum content, and the mean ToT distribution is estimated as the mean of the distribution obtained by projecting the selected window of the 2D distribution onto the Y axis. Figure **??** left shows the result of applying this method to the histogram in figure 6d. This analysis method is very fast and is useful for the tuning of the nanobeacon voltages, which is based only on the mean ToT and does not use the pulse arrival time. A more sophisticated

[1]This classification is currently done in a quite artisanal way, but so far it works quite well



(a) good pulse

(b) good pulse

(c) weak pulse

(d) saturated pulse

Figure 6: Examples of nanobeacon pulse histograms produced by `JPulseFinder` and analyzed by the `NBPulse` class.

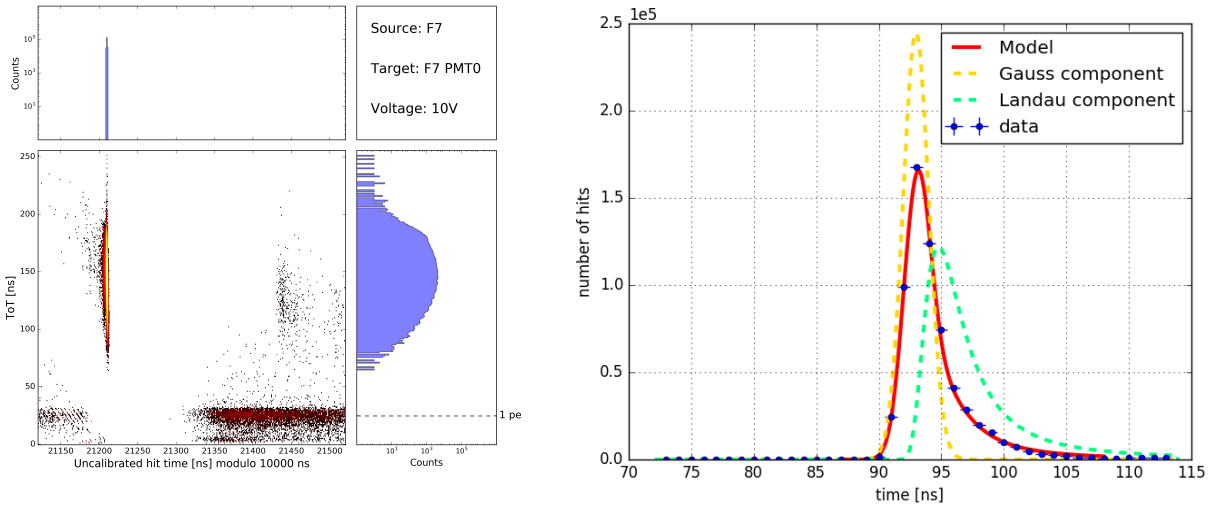


Figure 7: Examples of the two analysis methods implemented in the `NBPulse` class to estimate the pulse arrival time. The left plot shows the result of the fast analysis. In this case, the pulse arrival time is estimated from the bin with maximum content of the time distribution. The right plot shows the result of the more sophisticated method. In this case the pulse arrival time is the time at which the fitted model has its maximum.

way of estimating the pulse arrival time consists on fitting the hit time distribution to a mathematical model around the bin with maximum content, and to get the time at which the fitted model has its maximum value. Currently the following mathematical model is implemented:

$$M(\mu_g, \sigma_g, \mu_l, \sigma_l, \alpha) = G(\mu_g, \sigma_g) + \alpha \cdot L(\mu_l, \sigma_l). \tag{5}$$

It is a two component model where $G(\mu_g, \sigma_g)$ represents a Gauss law , $L(\mu_l, \sigma_l)$ represents a Landau law and α is a parameter that represents the mixing fraction of both components. An example of this fit is shown in figure 7. This simple model is implemented using the `ROOFIT` libraries REF. This libraries were chosen because they allow to implement easily more sophisticated models that could be used to perform studies of water properties. Besides, the `RooWorkspace` class allows to easily store all the information related to the fit and can be directly stored into a `.root` file.

4.1.2 The SUPERPMT class

The SUPERPMT class is very simple. It contains a JPMT and a NBPulse. It allows to associate the analyzed pulses to a particular PMT in the detector.

4.1.3 The SUPERMODULE class

A particular characteristic of the different analyses involving nanobeacons is that they usually involve at least two DOMs. A **source DOM**, and one or more **target DOMs**. The source DOM is the DOM hosting the triggered nanobeacon. Usually, the emission time of the nanobeacon is estimated as the arrival time of the pulse to one of the closest PMTs in the emitting DOM. This PMTs are also called the **reference PMTs**. A target DOM is any DOM that contains PMTs sensitive to the nanobeacon pulse.

A given DOM can act both as source and as target in the same analysis. When a DOM acts as a source it can point to multiple target DOMs in the DU and similarly, it can be pointed to by multiple source DOMs when it is acting as target. The SUPERMODULE class has been designed to implement these relationships between a DOM and the rest of the DOMs in the DU.

A SUPERMODULE contains a JModule object and has a vector of reference SUPERPMTS (ie, those that contain the NBPulses from its own nanobeacon detected by its own PMTs). It also contains a vector of **targets** and a vector of **sources**. A target is a pair made up of a pointer to another SUPERMODULE and a list of references to the corresponding target SUPERPMTS. Similarly, a source consists on a pair formed by a pointer to a source SUPERMODULE a list of references to the corresponding target SUPERPMTS. To clarify this, consider the SUPERMODULE corresponding to DOM in floor number 2. A possible target of this SUPERMODULE is the SUPERMODULE in floor 3. The corresponding pair will contain a pointer to the SUPERMODULE in floor 3, and an std::vector of references to the SUPERPMTs corresponding to the pulses emitted by the nanobeacon in floor 2, and detected by the PMTs in floor 3. On the other hand, a possible source for the SUPERMODULE in floor 2 is the nanobeacon from floor 1. The corresponding pair will consist on a pointer to the SUPERMODULE in floor 1, and an std::vector containing references to the SUPERPMTs with the pulses emitted by the nanobeacon in floor 1, and detected by the PMTs in floor 2.

The SUPERMODULE class contains methods to filter sources, targets and reference SUPERPMTs according to the quality of their NBPulses.

The the initialization and a coherent setting of references between the different SUPERMODULE objects in the DU are implemented in the NBRun class.

4.1.4 The NBRun class

The NBRun class intended to perform the analysis of a nanobeacon run for a given DU in a detector. Therefore, a nanobeacon run is instantiated with a JDetector, a DU number and the path to a file containing the nanobeacon pulses in the different PMTs from the detector, such as the ones shown in figure 6. When the NBRun class is instantiated, it creates a vector of references to the 18 SUPERMODULEs in the DU. For each of them, it sets the vectors of possible sources and targets accordingly. A program was included in the Tests directory to show that these references are set coherently.

The NBRun class also contains the implementation of the different functions that read and analyze the file containing the nanobeacon pulses.

4.2 JPulseFinder

4.2.1 What does it do?

JPulsefinder is the first program in the analysis chain. It uses a .detx file and a .root file from a KM3NeT nano beacon run. It produces 2D histograms as the ones in figure 6 for all the possible combinations of source nanobeacon and target PMTs in the DU and store them in a new .root file for its posterior analysis.

The complete list of options that JInterDomCal can receive through the command line are the following

```
-a <input detector file>

-f <path to input .root file>

-s <number of the DU to calibrate>

-o <name of out .root file>

-V <voltage of the nanobeacons>
```

`-p <pulse_period_16ns>`

`-d <overall_delay_16ns>`

`-stagger <stagger_16ns>`

The arguments `-a` , `-f` and `-s` are mandatory. Without specifying them, the program won't run. The option `-o` is not mandatory. In case that a name for the output file is not provided, a file named `out.root` will be produced in the same path where the binary `JPulsefinder` is. The same situation holds for the rest of the arguments, which are set to 0 by default. The last arguments are related to how the nanobeacon runs are set, and are discussed below.

4.2.2 How does it work?

The settings of a nanobeacon run include several parameters that can be modified. In order prevent the simultaneous arrival of the signal emitted by different nanobeacons at the same PMT, the nanobeacons are not flashed at the same time. Instead, the nanobeacons are flashed in ascending order from the bottom to the top floor of the string. Each pair of consecutive nanobeacons is flashed with a time difference called stagger time. This iterative procedure is known as the staggering method.

These concepts are depicted in figure **??**. This figure shows a hypothetical time distribution of the hits detected by a PMT, where the contribution of two nanobeacons that are staggered by 100 ns is visible. In this example, each nanobeacon emits pulses with a period of 200 ns. If the stagger time and the pulse period are chosen appropriately, the the contribution of each nanobeacon to the hits recorded by the PMT can be isolated. The main difference of this hypothetical example with respect to reality is that with the used settings, the PMTs detect only about one hit per nanobeacon pulse, and the single pulses are not visible as they are represended in figure **??**. This means that in practice it is impossible to work with the signal produced by the single nanobeacon pulses. But because the nanobeacons emit pulses periodically, this issue can be easily solved. If instead of plotting the distribution of hit times one represents the distribution of the reminder of the hit time divided by the pulse period (ie the modulo function), then the contribution of all the nanobeacon pulses would be cumulated together in a peak with much more statistics. This is shown in figure **??**, where the contribution of the 4 pulses from figure **??** are gathered in a single peak for each nanobeacon, with 4 times more statistics.

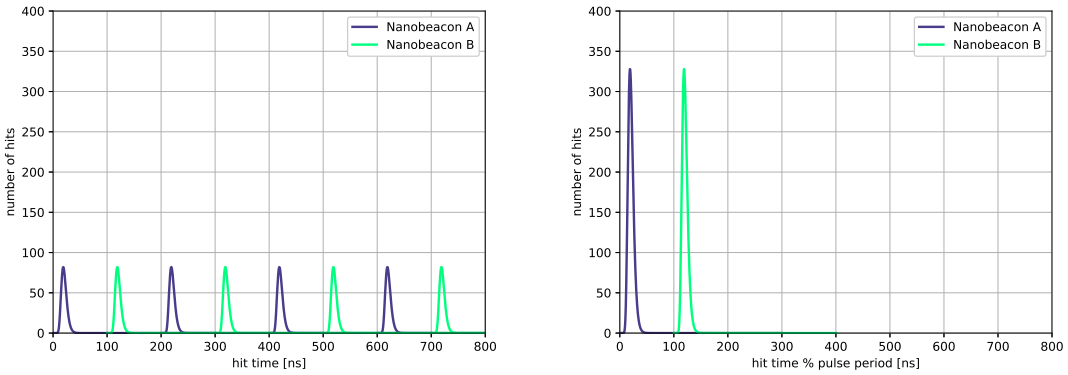


Figure 8: Hypothetical representation of the staggering technique, and how to gather the contribution of all the nanobeacon pulses into a single peak with larger statistics. See the text.

The argument `-stagger` specifies the stagger time in units of 16 ns. Additionally there can be a overall delay with respect to the beginning of a timeslice, which corresponds to the `-d` option. When they are triggered, nanobeacons emit pulses with a period corresponding to the `-p` option (should be given in units of 16 ns). With the current settings, `-stagger = 3520` , `-d = 0` and `-p = 6520`.

`JPulseFinder` is a modified version of the `MBeaconLite` program REF, which is suited for the current framework. What it does is to use the stagger time and the pulse period to isolate the contribution of each nanobeacon to each PMT as described in the hypothetical example above. It saves each individual contribution to a separate 2D histogram where the distribution of the ToT distribution of the hits is also represented in addition to the reminder of the hit time divided by the pulse period. For each histogram, the time range can be defined by using the stagger and overall delay times together with the positions of the nanobeacons and the PMTs in the detector. To fill these histograms the program loops over all the time slices of the run. For each hit it reads its time and ToT values and uses this information to fill the histograms for all the nanobeacons and the PMT that recorded the hit. If the time range of the different histograms has been correctly defined, the contribution of each hit will be visible only in the histogram for the nanobeacon that produced it. For the rest of the histograms, the hit will be allocated in the underflow or overflow bins.

4.3 JInterDomCal

4.3.1 What does it do?

As figure **??** shows, the JInterDomCal program needs a .detx file as well as a .root file as inputs. The detector file contains the positions of the PMTs and their t_0s, and the .root file contains the 2D histograms produced by JPulseFinder with the peaks observed by each PMT from each nanobeacon. This program performs the inter-DOM time calibration and produces a new .detx file which is a copy of the input .detx, but with the t_0s of the PMTs modified following the procedure described in section 2. Additionally, it can produce a .txt file with a comparison of the new and old detector files, and a .root file containing extra information about the peaks used for the calibration.

The complete list of options that JInterDomCal can receive through the command line are the following

```
-a <input detector file>

-f <path to input .root file>

-s <number of the DU to calibrate>

-x <name of out .detx>

-t <name of out .txt file>

-r <name of out .root file>

-u <up hemisphere pmts option>

-d <down hemisphere pmts option>

-n <maximum number of neighbors>

-l <peak analysis level>
```

The options -a , -f , -s are mandatory. Without these arguments the program won't run. The option -x is not mandatory. In case that a name for the output detector file is not provided, a file named out.detx will be produced in the same path where the binary JInterDomCal is. The

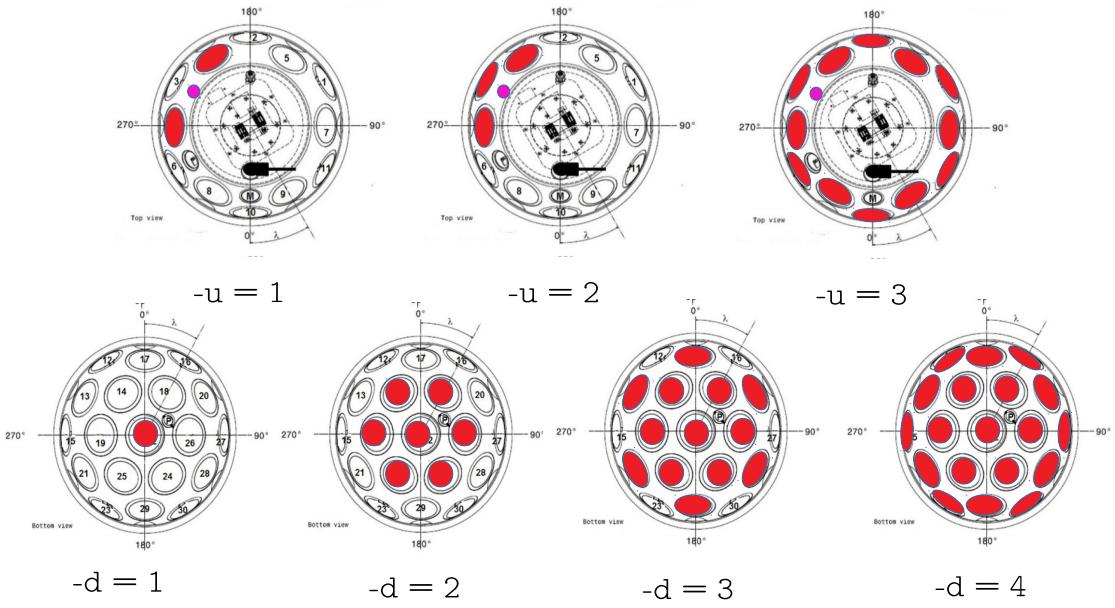


Figure 9: Different choices for the arguments `-u` and `-d`

arguments `-t` , `-r` are optional. If they are not provided the corresponding files won't be produced[2]. The arguments `-u` , `-d` represent options to indicate `JInterDomCal` which PMTs (or which not) to use in the upper and lower hemisphere to perform the calibration. The possible choices for these arguments are given by different integer values and are summarized in figure 9. The argument `-n` allows to limit the maximum distance between source and target DOMs when performing the calibration. The default value is set to 2. Finally, the argument `-l` allows to choose how the nanobeacon pulses are analyzed. It's value is also an integer number, and currently two possibilities are implemented: `-l = 0` and `-l = 1`, which correspond respectively to the fast and more sophisticated analyses described in section 4.1.1. The default option is set to 0.

4.3.2 How does it work?

The program is rather short. It's main stages are the following:

1. Read input options

[2]The function that produces the optional `.root` file with the analysis info. is not yet implemented

2. Instantiation of a NBRun with the user options

3. Analysis of the NBRun according to the user options

4. Generate a new detector according to the results of the previous step

5. Write output

During the analysis of the run, the code selects pairs of DOMS that are as close as possible. For each pair of DOMs, the code analyzes each possible combination of reference and target SUPERPMTs that are classified as good. A correction factor is computed for each pair as described in section 2, and the t_0s of the detector are updated by using the median correction. Note that if one gets too conservative with respect to the options `-n` , `-u` and `-d` some DOMs may not be calibrated.

4.4 JVoltageOptimizer

4.4.1 What does it do?

As figure **??** shows, `JVoltageOptimizer` uses the result of multiple nanobeacon runs analyzed by `JPulseFinder` to find the optimal voltage at which each nanobeacon in the DU should be tuned. It returns a `.txt` file with a list of nanobeacons and voltages. Optionally, it produces a `.root` file containing some extra information related to the optimization process, and a second `.root` file containing the pulses emitted by the nanobeacons tuned at the optimal voltage that can be used by `JInterDomCal` to calibrate the DU. The command line arguments for `JVoltageOptimizer` are the following:

```
-a <input detector file>

-f <list of paths to .root files>

-s <number of the DU to analyze>

-t <name of out .txt file>

-o <name of out .root file with extra info>

-c <name of out .root file for calibration>
```

```
-u <up hemisphere pmts option>

-d <down hemisphere pmts option>

-l <peak analysis level>

-n <maximum DOM distance>

-S <saturation ToT value>
```

The arguments `-a` , `-f` , `-s` , `-u` , `-d` , `-l` and `-n` have a similar meaning that those for `JInterDomCal`. `-t` is for the name of the output file with the optimal nanobeacon voltages. By default a file named `out.txt` is produced in the path where the binary `JVoltageOptimizer` is. `-o` and `-c` are respectively the names of the optional output files with extra information about the optimization process and for the calibration. If those arguments are not provided, the files won't be produced. The argument `-S` defines the nanobeacon saturation voltage, which by default is set to 26.4 ns. The optional `.root` file contains a tree with a branch for each nanobeacon.

4.4.2 How does it work?

The results of `JVoltageOptimizer` are also based on the analysis of the nanobeacon signal implemented in the `NBRun` class. For each nanobeacon, the program finds the mean of the ToT distribution for the hits produced in each of its reference PMTs, and the average of these values is stored to find the optimal voltage. If the `-o` option is chosen, a tree is created which contains a branch for each nanobeacon. Each event in the tree corresponds to a different run (voltage), and contains the average ToT, its standard deviation and the voltage of the run. This tree allows to make plots as the ones in figure 4.

5 Scripts

Two scripts are included in `$JNanoBeacons/Scripts`. The script `Calibrate.sh` runs the different programs involved in the calibration chain, and the script `Optimize_Voltage.sh` does the same for the voltage optimization. In each case the user needs to specify the run number of the runs involved in the analysis, as well as the different command line arguments. Both scripts download from `irods`

the `.root` files corresponding to the needed runs, and delete them after they have been analyzed. The idea is to use these scripts to regularly perform the calibration and the voltage optimization every time that a DU is deployed in the sea water.