Detector calibration

Maarten de Jong, BoukeJisse Jung

2026-04-17 16:31:54

Abstract

The calibration of the detector as seen from a Jpp perspective is presented.

1 Introduction

The science output of KM3NeT will depend to a large extent on the availability and accuracy of the calibration of the detector. In view of the envisaged life time of KM3NeT, the calibration of the detector should be based on an analysis of *in situ* data. In the following, a description of the Jpp software needed for the calibration of the detector is presented. Although acoustic position calibration should also be considered an analysis, it is not (yet) considered part of this discourse.

As a reminder, the data-acquisition (DAQ) system is based on the "all-data-to-shore" concept. In this, the analogue output of a PMT is processed via an amplifier (with pulse shaping) and a time-over-threshold discriminator. The typical threshold of the discriminator corresponds to 0.3 photo-electron equivalent (0.3 p.e.). Each analogue pulse that passes this threshold is digitised. The digital data contain the time stamp of the leading edge and the pulse length of the time-over-threshold signal from the discriminator. The digitisation is implemented inside an FPGA that is located inside the optical module. Hence, the name digital optical module (DOM). The least significant bits of the time stamp and the pulse length correspond to 1 ns. The dynamic ranges of the time stamp and pulse length are 2^{32} and 2^8, respectively. To efficiently process the data on shore, the continuous data streams from the optical modules are sliced in time. The duration of the time slice is 100 ms. Events are filtered on-shore using custom software running on a farm of servers. All data corresponding to the same time slice are sent to the same server. The different servers provide the capacity to process the incoming data in real time. An event is triggered by a cluster of a minimal number of causally related hits. Normally, all filtered data are written to disk. In addition, level-0 (L0), level-1 (L1), level-2 (L2), Supernova (SN), and summary data can be written to disk according a preset sampling (e.g. every n^{th} time slice or no data when $n = 0$). The summary data contain the rate and status of all PMTs in the detector per time slice. To limit the volume of the summary data, the measured rate is compressed to a single byte. The L0 data contain all the digital data (i.e. time stamp and time-over-threshold) from all PMTs in the detector. These data are thus totally unbiased. The L1 data contain local coincidences between two (or more) L0 hits in one optical module within a predefined time window. The typical time window is 20 ns. The L2 and Supernova data are a subset of the L1 data subject to additional constraints.
In the following, the positions of the PMTs are relative to the position of the detector as defined in the UTM coordinate system and the time calibration of the PMTs applies to the UTC time of the data. The concept of calibration simply refers to the addition of a (small) offsets to previous values.

2 PMT time calibration

The PMT time calibration (also known as "intra-DOM calibration") refers to the calibration of the relative time offsets of the PMTs inside the same optical module. For this, level-0 (L0) or level-1 (L1) hits can be used. The L0 and L1 data primarily contain background signals from ^{40}K decays in the water. Such a decay produces a sizeable number of Cherenkov photons. For those decays occurring in the vicinity of an optical module (typically within few metres), the Cherenkov photons can produce a time coincidence of two (or more) hits in the optical module. The PMT time calibration exploits the time correlation between these hits.

The L0 or L1 data should sequentially be processed with the following applications.

- `JCalibrateK40`
- `JMergeCalibrateK40`
- `JFitK40`

`JCalibrateK40` is used to process the L0 or L1 input data. In this, a 2D-histogram is produced for each optical module containing the time differences between hits from each *unique* pair of PMTs in that optical module. For an optical module with 31 PMTs, there are $31 \times 30/2 = 465$ unique pairs. At the same time, the random background and live time of each PMT pair are monitored using separate 1D-histograms. By default, the calculation of the background is based on the probability of random coincidences using the measured singles rates obtained from the summary data. The selection of hits is such that the data from a PMT with a high-rate veto or FIFO (almost) full of the off-shore DAQ or UPD packet losses are discarded. This is taken into account in the determination of the live time.

`JMergeCalibrateK40` can be used to convert the output of `JCalibrateK40` into a single histogram for each optical module with the normalised coincidence rates of each PMT pair. It is possible to convert multiple outputs of `JCalibrateK40` in one go. Hence, one can apply `JCalibrateK40` on a run-by-run basis and combine different data taking runs *a posteriori*.

`JFitK40` can be used to fit a model to the output histograms of `JMergeCalibrateK40`. The model is based on a parametrisation of the coincidence rate as a function of the opening angle between the axes of the two PMTs. The model parameters have been determined from detailed GEANT4 simulation (using option `JFitK40 -M`). The list of parameters in the fit includes the t_0, QE and TTS of each PMT. In this, t_0 refers to the offset of the global time calibration of a PMT, QE to the quantum efficiency and TTS to the transition-time spread.

In general, the probability to convert a photon to an electron depends on the wavelength of the light. This probability is commonly referred to as quantum efficiency (QE). Furthermore, there is a finite probability that the electron passes through the subsequent amplification stages in the dynode of the PMT and produces an analogue pulse. This is referred to as the collection efficiency. In the following, the collection efficiency is considered part of the quantum efficiency. The angular acceptance of the PMT and in-homogeneity of the photo-cathode area are considered part of the fit model. The QE may significantly vary for the different PMTs. The observed coincidence rate thus depends on the QE of the PMTs in the corresponding pair. A renormalisation of the QE is taken into account in the fit. The PMT QE calibration is described below.

The time between a photon hitting the photo-cathode of the PMT and the time of the corresponding analogue pulse is referred to as the transition time. Due to the different paths of the photo-electron and the electrons in the subsequent amplification stages in the dynode of the PMT, the transition times show a spread. This spread is modelled in the fit by a Gaussian function. Note that in the simulation of the PMT

response in application JTriggerEfficiency, the transition time is reproduced from measurements in the laboratory.

By default, the average time offset of the PMTs (i.e. average of the fitted t_0's) in the same optical module is constrained to 0. The number of free parameters is then reduced by 1. In this mode, the PMT time calibration can independently be made of other time calibrations, including the string time calibration (see below). In other cases such as tuning of the high voltages, the time offsets of specific PMTs should be constrained to maintain the results of previous calibrations.

The PMT time calibration procedure is based on the following steps.

1. L0 or L1 data taking;
2. `JCalibrateK40 -a <detector calibration file> -C <selector> -f KM3NeT_XXXXXXXX_YYYYYYYY.root -o <workdir>/calibrate.root`
3. `JMergeCalibrateK40 -f <workdir>/calibrate.root -o <workdir>/merge.root`
4. `JFitK40 -a <detector calibration file> -f <workdir>/merge.root -w -o fit.root`
5. `JFitK40 -a <detector calibration file> -f <workdir>/merge.root -w -o fit.root -A`

In this, <detector calibration file> refers to the standard ASCII formatted file with detector calibration data ("detx" file) [1] and <workdir> to some working directory. For the first step, data corresponding to a live time of at least 15 minutes should be taken to accumulate sufficient statistics. In the second step, the option to select the data type should be `-C JDAQTimesliceL0` and `-C JDAQTimesliceL1` for L0 and L1 data, respectively. After verification of the output of the fourth step, the detector calibration file could be updated, as indicated in the fifth step (option `-A`).

3 String time calibration

Before the deployment of a string, the time offsets of some reference PMT(s) in each optical module have been calibrated using a designated laser calibration set-up. The corresponding detector calibration file should be archived for future use (e.g. `detx` file). It is also mandatory to archive the list of reference PMTs (this information is not explicitly stored in the detector calibration file). This archive should contain the optical module identifier and the PMT readout channel of each reference (i.e. calibrated) PMT. The list of optical module identifiers and PMT readout channels of the reference PMTs can conveniently be stored in an ASCII formatted file. After the deployment of a string, the time offsets of the PMTs with respect to the reference PMT(s) should be determined as well as the time offset of the whole string with respect to the previously deployed strings. The latter is normally done using the White Rabbit (WR) system. Considering the lengths of the optical fibres, the various asymmetries in the fibre-optic system and the different delays due to dispersion, it may be required to verify (and possibly to correct) the time calibration of the string. Alternatively, the time calibration of a string can be determined by maximisation of the efficiency to trigger an event as a function of the time offset of the string. The detection of atmospheric muons provide for sufficient count rate and coverage of the detector. For such a procedure, the auxiliary script `JCalibrateStrings.sh` can be used. In this, the settings of the trigger parameters can be tuned to cover a large range of time offsets or to achieve high accuracy of the time calibration. The quoted Supernova data (see above) provide an unbiased input with high-purity for this procedure.

3.1 Laser calibration

The time offsets of some reference PMT(s) in each module should be calibrated before the deployment of the string. To this end, a designated set-up with a laser is used. In this, the light from a laser is distributed to the photo-cathode of each reference PMT via optical fibres with equal lengths. The laser should be synchronised with the clock system of the detector and produce single pulses at a fixed frequency. For a laser frequency of $10\ \mathrm{kHz}$, 5 minutes (or more) of L0 data should be taken. The application `JPulsar` can then be used to determine the relative time offsets of the reference PMTs. The reference PMTs should be specified on the command line of `JPulsar` as follows

```
JPulsar -a <detector calibration file> -f KM3NeT_XXXXXXXX_YYYYYYYY.root -!  "<module
identifier> <PMT readout channel> ..." -A
```

In this, multiple pairs of `"<module identifier> <PMT readout channel>"` can be specified. The quotes are needed to correctly assign all values to the corresponding option. Depending on the shell in use, it may be necessary to precede the option with a backslash (i.e. `-\!`). Each pair should correspond to one of the reference PMTs. A wild card (`-1`) can be specified for the module identifier. By default, the specified PMT readout channel then applies to all modules in the detector. This default can be modified for a specific module by an additional specification, e.g. `"-1 1 12345678 10"` will define the PMT readout channel `10` as the reference for module `12345678` and otherwise `1`. If the list of optical module identifiers and PMT readout channels of the reference PMTs is stored in an ASCII formatted file, the corresponding option can simply read `-!  <file name>`. The time offsets of the other PMTs are set to that of the reference PMT in the same optical module appearing first in the list. The resulting detector calibration is then input to an *in situ* PMT time calibration, as described next.

3.1.1 Transition-time distribution

The transition-time of the signal in the PMTs is simulated in `JTriggerEfficiency`. For this, the combined distribution of the measured hit times of the reference PMTs is also stored in the output of `JPulsar`. After the standard laser calibration, the underlying distributions will be centred at zero. As a result, the combined distribution of the measured hit times represents an average transition-time distribution. The probability density function (PDF) and the cumulative density function (CDF) thereof can subsequently be determined using application `JLegolas`. To reduce the impact of statistical fluctuations, the data are smoothed. The auxiliary script `JLegolas.sh` can be used for the complete procedure of producing the PDF and CDF of the transit-time distribution. In this, two ASCII formatted files are produced for inclusion in `JPMTTransitTimeProbability.hh` and `JPMTTransitTimeGenerator.hh`, respectively. The latter is included in `JTriggerEfficiency`. A correction for time slewing exists which is implemented in `JTimeSlewing.hh` as a time offset as a function of the time-over-threshold. The applications `JFrodo` and `JPerth` can be used to determine and improve the time-slewing correction, respectively. An updated transition-time distribution becomes thus effective after the standard recompilation.

3.2 PMT time calibration

A time calibration of the PMTs with respect to the reference PMTs from the laser calibration should be made as soon as possible after the deployment of the string. This procedure is based on the standard PMT time calibration (see above) but with corresponding constraints applied. These constraints can be specified on the command line of `JFitK40` as follows

```
JFitK40 -!  "<module identifier> <PMT readout channel> ..."
```

In this, multiple pairs of "<module identifier> <PMT readout channel>" can be specified. Each pair should correspond to one of the reference PMTs of the laser calibration. So, the value at option JFitK40 -! should be identical to the value at option JPulsar -! (see above). The resulting detector calibration is then input to the following string time calibration.

3.3 String time calibration

Following the *in situ* time calibration of the PMTs with respect to the reference PMTs from the laser calibration, a verification (or correction) of the time calibration of a string should be made after its deployment. To this end, event data should be analysed. The event data are primarily triggered by muons produced by interactions of cosmic rays in the atmosphere above the detector, commonly referred to as atmospheric muons. These muons can produce time correlated signals in two (or more) strings. The string time calibration exploits the time correlation between the hits from atmospheric muons.

The string time calibration procedure could be based on the following steps.

1. Modifications of detector calibration file;
2. Repeated reconstruction of muon trajectories using different detector calibrations;
3. Determination of the optimal detector calibration;

A modification of the detector calibration file can conveniently be made with application JEditDetector. In principle, any modification is possible but the CPU time of the second step may limit number of modifications. The reconstruction of muon trajectories can be realised by subsequent processing of the event data with the applications JMuonPrefit, JMuonSimplex and JMuonGandalf. The script JCalibrateTime.sh serves as an example to do the first two steps. In this, a common time offset is applied to the PMTs of a given string. This is achieved using option JEditDetector -S "<string identifier> add <time offset [ns]>". The determination of the optimal detector calibration is based on the maximum of the total quality of the reconstructed muon trajectories. The script plot-time.sh serves as an example to do the last step. In this, the optimal calibration is determined by a fit of an M-Estimator function to the total quality as a function of the applied time offset. The scripts JCalibrateTime.sh and plot-time.sh are located in sub-directory examples/JCalibrate/ in the Jpp software repository.

During nominal operation (e.g. with a complete detector), a similar procedure can be applied to continually monitor the time calibration of the strings and possibly to calibrate the position and orientation of the anchors of the string.

4 Optical module time calibration

The time calibration of the optical modules is *a priori* defined by the laser calibration (see above). During normal data taking, the time calibration of the optical modules can be verified and corrected. This is commonly referred to as "inter-DOM calibration". For this, the standard reconstruction of muon trajectories (i.e. JMuonPrefit, JMuonSimplex and JMuonGandalf) can be used. For the time calibration of optical modules, the application JCalibrateMuon can be applied to the output of the standard reconstruction. In this, the time residuals between the hits and the muon trajectories are recorded for each optical module. To avoid a possible bias, the data from the optical module in question are excluded from the event data when refitting the muon trajectory. For further analysis, a 2D-histogram with the corresponding distributions is stored in the output file of JCalibrateMuon. The application JHobbit can be used to process the 2D-histograms from a set of output files. In this, the contents of the 2D-histograms from the different files

are added and independent fits are made to the time residual distributions of each module. Various fit functions are possible (see option `JHobbit -h!` for the actual list of possible fit functions). Each fit function includes a free parameter for the time offset. The option `JHobbit -A` can be used to correct the detector calibration file specified via option `JHobbit -a <detector calibration file>`. In this, the average of the time offsets of all optical modules is constrained to 0.

5 Nano-beacon time calibration

The time calibration of the optical modules can be verified (or corrected) using the nano-beacon which is located in each optical module [2]. It is interesting to note that the light from the nano-beacon travels mainly upwards whereas the light from atmospheric muons travels mainly downwards. A comparison between the time calibration with nano-beacons and atmospheric muons can thus be used to disambiguate the time offset and the distance between neighbouring optical modules.

6 HV tuning

A regular tuning of the high voltage (HV) supplied to the PMTs is needed to ensure that the analogue pulses of the PMTs always match the preset threshold of the discriminator. The procedure consists of three consecutive steps applied to data acquired for this purpose. For this, designated calibration runs are taken in which the HV of a subset of the PMTs are varied in steps between typically -125 V and $+125$ V with respect to the actual HV or the vendor HV (i.e. the voltage recommended by the manufacturer). The offset between the vendor high-voltage and the applied high-voltage is commonly expressed in terms of an index. Incrementing this index by a value of 1 approximately corresponds to increasing the absolute high-voltage setting by about 3.14 V:

$$\Delta HV = -3.14 \times i. \tag{1}$$

When taking calibration runs for the purpose of high-voltage tuning, L0 data should be acquired with a corresponding live time of at least 5 minutes. This minimizes possible biases and ensures that sufficient statistics are acquired for each PMT.

The first step of the HV tuning procedure consists of collecting the time-over-threshold values from each calibration run. This can be done using the application `JCalibrateToT`. The data are stored in separate histograms for each PMT.

In the second step, the gain and gain spread of a PMT are determined from a fit of a model function to the data (see [3]). This is achieved using the application `JFitToT`.

The third step consists of associating the measured gains with the corresponding high-voltage settings for every PMT. This can be achieved using the application `JGetInputTuneHV`.

The fourth and final step consists of an interpolation of the high-voltage setting which optimizes the PMT gain. For this purpose, the application `JTuneHV` can be used. The interpolation is based on the fact that the gain and high-voltage are related according to a power law:

$$G = A \times HV^{kN}, \tag{2}$$

where A, k and N are constants related to the dynode system of the PMT. To determine the high-voltage corresponding to a nominal gain of 1.0, the data are linearised and subsequently interpolated.

The eponymous bash script JTuneHV.sh can be used to run all four steps in one go. This script can be run via the command line as follows:

<Jpp>/examples/JCalibrate/JTuneHV.sh <detector calibration file> <input files> [output directory] [PMT parameter file].

In this line, <input files> corresponds to the input data files, formatted as an array. The optional argument [output directory] can be used to specify where the final output is saved (default is in /tmp), whilst the argument [PMT parameter file] can be given to specify the default PMT parameters. Furthermore, the type of time data (e.g. JDAQTimesliceL0 or JDAQTimesliceL1), the database API version, the database test type, the default output file name and the ROOT fit range and fit options for JFitToT can be specified by respectively setting the shell variables: TUNEHV_TIMESLICE_SELECTOR, TUNEHV_DB_APIVERSION, TUNEHV_DB_TESTTYPE, TUNEHV_OUTPUT_FILE, TUNEHV_GAINFIT_RANGE, TUNEHV_GAINFIT_OPTIONS to the desired value, if needed. The available database API versions are listed in the class JDATABASE::JDBAPIVersion, whilst the available database test types are listed in the class JDATABASE::JDBTestTypesTuneHV.

The script automatically generates three sub-directories, namely "caldata", "PMTfiles" and "fitdata", where the output of each application is stored. The final output consists of a JSON file containing the calibration data and a ROOT file with the HV-gain diagrams used for evaluating the nominal high-voltage setting for each PMT. This final output is stored in the main directory (i.e. [output directory]). A summary of the calibration, including a list of any failed evaluations, can be printed on the screen using the standalone application JPrintTuneHV.

Failed evaluations are flagged by a FAIL label in the JSON output. There are three scenarios which can give rise to a failure.

The first type of failure occurs when none of the time-over-threshold distributions for a given PMT contain sufficient data for fitting (the minimum required histogram entries are set by the option -@Wmin in JFitToT). This either means that the absolute high-voltage settings for the different calibration runs were too low or simply that the PMT is unresponsive (dead or contained in a dead module).

The second case of failure occurs in situations where the optimal HV-setting is not contained within the scanned HV range. As long as the projected optimal setting is separated from the outermost data points no further than twice the HV-difference between the individual calibration runs, JTuneHV will perform an extrapolation instead of an interpolation and write the resulting high-voltage to the JSON output. However, if the optimal value lies further away, the resulting entry is labelled FAIL.

The final situation where failures may occur, is when one or more of a PMT's time-over-threshold distributions have an anomalous shape. This will cause the HV-gain data points to deviate from the behaviour described by equation 2. Whenever the closest points above and below the nominal gain value are not strictly increasing as a function of the HV, the resulting output will be marked FAIL.

All individual cases of failure can be listed and written to an individual JSON file using the application JPrintTuneHV. They can be inspected visually by plotting the corresponding HV-gain diagrams using e.g.: JPlot1D -f <ROOT file>:<module identifier>.<PMT readout channel>,

where ROOT file denotes the ROOT output file of JTuneHV.

The application JEditTuneHV can be used to update any failures listed within a JSON file, either according to a given vendor or run-specific high-voltage table, or a according to manual specifications. The corresponding high-voltage tables can be created using the application JAsciiDB. A bash script (i.e. JEditTuneHV.sh) is available, to produce a desired input high-voltage table and edit a JSON-output file accordingly. This script can be run via the command line as follows:

```
<Jpp>/examples/JCalibrate/JEditTuneHV.sh <input JSON file> <output JSON file>
[detector identifier] [run identifier] [selection of PMT UPIs],
```

If both a detector identifier and run identifier are specified in this line, the JSON-file will be edited according to the high-voltages of the corresponding DAQ run. Instead, if only a detector identifier is specified, the JSON-file will be edited according to the corresponding vendor high-voltages. If neither a detector nor run identifier are specified, the program will ask for manual specifications. A list of PMT UPIs can be specified optionally, to adjust only the results for a given subset of PMTs.

Individual JSON-output files created using the applications JTuneHV and JEditTuneHV can be merged into one file using the application JMergeTuneHV. If a PMT is listed among multiple files, only the result corresponding to the first given file is maintained.

When all steps in the HV tuning procedure have been completed and once the final output has been inspected and verified, it is possible to upload the derived optimal HV settings to the database. This can be done by converting the JSON output to a "product test file" (see [4]), and sending it to the database, either via the database web page [1] or, alternatively, via wget. For more information on how to do this, please see the corresponding wiki page.

Once the new high-voltage calibration data has been uploaded, the database coordinator should be notified, such that he or she can make the data available to the wider collaboration. All tuned high-voltage data sets used in the past can be found in the file "pmt_available_hvtuned_sets.txt". This file can be accessed via the database.

6.1 Time Recalibration

Because the transition time of the PMT depends on the HV ($t \propto \frac{1}{\sqrt{V}}$), a procedure to maintain the time calibration of PMTs should always follow the tuning of the HVs of the PMTs. This procedure is based on the following steps.

1. PMT time calibration with previous HVs on all PMTs in each optical module;
2. PMT time calibration with tuned HVs on a subset of PMTs in each optical module.
3. PMT time calibration with tuned HVs on all PMTs in each optical module;

The first step is meant to ensure that the PMTs are well time calibrated *before* the changing the HVs. In case some PMTs are not well time calibrated, the fitted time offsets should be applied to the current detector calibration file. The result of this step is referred to as <detector calibration - 1.>.

For the second step, a convenient subset of PMTs should be defined of which the HV is not changed. For example, by selecting the odd readout channels of the PMTs of all optical modules. As a result, the PMTs corresponding to odd readout channels have an optimal HV. These PMTs should now be time calibrated.

[1] See the *Product Tests* tab on https://km3netdbweb.in2p3.fr/default.htm.

In the third step, all PMTs have an optimal HV. The PMT time calibration should now be done whilst constraining the time offsets of the other PMTs (i.e. those PMTs of which the tuned HV was already used in the second step and therefore have the same value in the third step).

These constraints can be specified on the command line of `JFitK40` as follows.

```
JFitK40 -! "-1 1 -1 3 ... -1 29"
```

In this, the values between quotes refer to "<module identifier> <PMT readout channel>". The value `-1` corresponds to *any* optical module identifier. If these values are stored in an ASCII formatted file, the corresponding option can simply read `-! <file name>`. Such a file can be generated with application `JTDC`. In this, the actual HVs are extracted from table "pmt_hv_run_settings" in the database for the two corresponding data taking runs and used to determine the constraints. As a result, the constraints are consistent by construction.

```
JTDC -r "<first run> <second run>" -o TDC.txt
```

The detector calibration file from step 1. should first be updated. This could be done as follows.

```
JConvertDetectorFormat -a <detector calibration - 1.> -o <detector calibration - 2.>

JFitK40 -a <detector calibration - 2.> -A -! TDC.txt ...
```

In the third step, the constraints can be reverted as follows

```
JFitK40 -! "-1 1 -1 3 ... -1 29" -r
```

In this, option `-r` will make `JFitK40` reverse the specified constraints. Alternatively, the application `JTDC` can be used again.

```
JTDC -r "<second run> <third run>" -o TDC.txt
```

The detector calibration file from step 2. should now be updated. This could be done as follows.

```
JConvertDetectorFormat -a <detector calibration - 2.> -o <detector calibration - 3.>

JFitK40 -a <detector calibration - 3.> -A -! TDC.txt ...
```

The final detector calibration is the result of 3.

7 PMT gain calibration

The PMT gain calibration refers to the determination of the gain and gain spread of the PMTs (see [3]). For this, level-0 (L0) or level-1 (L1) hits can be used. Genuine coincidences between different PMTs are filtered out to avoid a bias due to multiple photons in a single hit. The PMT gain calibration exploits the time-over-threshold distribution of single photo-electron hits. Therefore, one should effectively cut the contribution of multiple photons in a hit for L1 data.

The L0 or L1 data should sequentially be processed with the following applications.

- JCalibrateToT
- JMergeCalibrateToT
- JFitToT

JCalibrateToT is used to process the L0 or L1 input data. In this, a 2D-histogram is produced for each optical module containing the distribution of the time-over-threshold from each PMT in that optical module. Note the command line options -T <T_ns> and -c <ctMax> which should be used to preferably select single photo-electrons hits from L1 data (i.e. reject genuine coincidences from ^{40}K decays).

JMergeCalibrateToT can be used to convert the output of JCalibrateToT. It is possible to convert multiple outputs of JCalibrateToT in one go to combine statistics. Hence, one can apply JCalibrateToT on a run-by-run basis and combine different data taking runs *a posteriori*.

JFitToT can be used to fit a model to the output histograms of JMergeCalibrateToT. The model is based on a mathematical description of the analogue pulse from the PMT. In this, the leading edge is described by a Gaussian and the trailing edge by an exponential. These are matched at one point where the two function values and the derivatives thereof are equal. The specification of the peak of the time-over-threshold distribution further relates the sigma of the Gauss to the exponential slope.

The probability that the analogue pulse passes the threshold of the discriminator is referred to as the "survival" probability. The survival probability depends on the gain (and the gain spread) and thereby on the HV. For a correct determination of the QE of the PMT (see below) and a correct simulation of the PMT response, the gain and gain spread should be determined before the QE calibration. The survival probability is then accounted for.

8 PMT QE calibration

The probability to detect a single photon is referred as the quantum efficiency (QE) of the PMT. The QE of the PMTs can be determined with the same procedure as used for the PMT time calibration (see above). To store the measured QEs, option JFitK40 -P <PMT parameters file> can be used. The <PMT parameters file> is an ASCII formatted file which contains not only the values of the fitted QEs but also other PMT parameters. This file can directly be used in the simulation of the PMT response using option JTriggerEfficiency -P <PMT parameters file>.

The probability that the analogue pulse from the PMT passes the threshold of the discriminator depends on the gain and the gain spread of the PMT. The gain depends on the high voltage (HV) applied to the PMT and typically amounts to 10^6. The typical gain spread is 0.4–0.5. The simulation of the processing of the analogue pulse through the discriminator is based on a designated model[5]. To ensure consistency, the measured QE is corrected for the calculated probability that the analogue pulse from the PMT passes the threshold of the discriminator. This correction compensates the corresponding probability that the analogue pulse of a hit does not pass the discriminator threshold in the simulation of the PMT response in application JTriggerEfficiency.

9 PMT calibration revisited

The threshold of the PMT is considered fixed (i.e. it is not determined by a calibration procedure) but it is part of the PMT parameters file. So, if the threshold of a PMT was set during data taking to a different

value than the nominal one, the PMT parameters file should be updated before the calibration of the gain, gain spread and QE. This can manually be done using the application JEditPMTParameters with option -M "<module identifier> <PMT channel> set threshold <value>". To automatically set all PMT thresholds to those of the DAQ configuration, the application JPMTTreshold can be used. This involves a transfer function from the DAQ value to a value in units of number of photo-electrons. A default function is implemented in application JPMTTreshold which can be redefined with option -F formula (see option -h! for the possible syntax).

In addition, the transit-time distribution of each PMT can be set. This can manually be done using the application JEditPMTParameters with option -M "<module identifier> <PMT channel> set TTS_ns <value>". In this, a positive value corresponds to the width of a Gaussian distribution and a negative integer value to a function identifier. To automatically set all transit-time distributions to the measured ones, the application JPMTTTS can be used.

Following the calibration of the gain, gain spread and QE, several corrections should be applied to the measured QE before use (e.g. in JTriggerEfficiency). Due to the finite probability that one hit is caused by two (or more) Cherenkov photons, there is a difference between the probability to detect a hit and the actual QE. A correction for this difference can be made using the application JEditPMTParameters. For this, the option -E <mu> is available. In this, mu corresponds to the expectation value of the Poisson probability distribution to detect a given number of Cherenkov photons, given that there is a coincidence between two (or more) hits due to a ^{40}K decay. The typical value for mu is in the range 0.25–0.40. A value of mu = 0 corresponds to a QE equal to the hit probability. Another correction should be made due to the cut on the time-over-threshold values applied in application JCalibrateK40. This cut is applied to reject noise hits which typically yield (very) small time-over-threshold values. As a consequence, also a small fraction of signal hits is rejected. This can be corrected for using option -T "Tmin_ns Tmax_ns". In addition, options -M and -A can be used to change the value of a specific parameter of a specific PMT or a specific physical address (i.e. ring and position), respectively.

References

[1] km3net_soft_2016_003.

[2] <jpp>/documentation/jnanobeacon/jnanobeacons.pdf.

[3] Bouke Jung. Pmt signal processing. https://common.pages.km3net.de/jpp/PMTmodeling.PDF. last accessed on 2020-07-18.

[4] Cristiano Bozza. Database/product tests and parameters. https://wiki.km3net.de/index.php/Database/Product_Tests_and_Parameters. last accessed on 2020-07-18.

[5] <jpp>/documentation/jcalibrate/jpmtanaloguesignalprocessor.pptx.